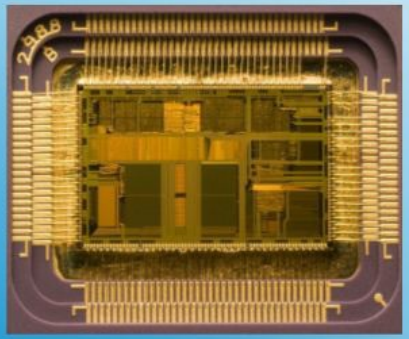
C# 程式設計

- *基礎程式*
- *物件導向*
- *視窗程式*
- *網路程式*
- *遊戲程式*



作者：陳鍾誠

C# 程式設計

陳鍾誠

2013 年 8 月

C# 程式設計

1. C# 程式基礎：簡介與開發環境
    1. Visual Studio 安裝
    2. 命令列工具
    3. 視窗程式設計
    4. 資料庫
    5. 網路
    6. 網頁
    7. 手機
    8. 遊戲
    9. 跨平台的考量
    10. 最新的發展
    11. 結語
    12. 參考文獻
2. C# 程式基礎：變數與運算式
    1. 簡介
    2. 變數與型態
    3. 測驗
    4. 結語

    5. 參考文獻
3. C# 程式基礎：流程控制
    1. if 語法
    2. for 語法
    3. while 語法
    4. 結語
    5. 練習：if 範例
    6. 練習：while 範例
4. C# 程式基礎：陣列
    1. 一維陣列
    2. 二維陣列
    3. 參考文獻
5. C# 程式基礎：函數
    1. 簡介
    2. 函數的參數
    3. 練習
6. C# 程式基礎：物件
    1. 封裝
    2. 繼承
    3. 多型
    4. 參考文獻

7. C# 程式基礎：例外處理
    1. 範例：引發例外
    2. 範例：捕捉例外
    3. 視窗版的例外處理
8. C# 程式基礎：資料結構
    1. 容器函式庫
    2. 容器的使用
9. C# 程式基礎：檔案處理
    1. 讀取檔案
    2. 寫入檔案
    3. 物件導向的寫法
    4. 更物件導向的寫法
10. C# 程式進階：正規表達式
    1. 程式範例
11. C# 程式進階：作業系統與 Thread
    1. Process 與 Thread
    2. C# 中的 Thread 概念
    3. 以 C# 體驗死結 (Deadlock)
    4. 競爭情況 (Race Condition)
    5. 號誌 (Semaphore) 與生產者/消費者問題
    6. 參考文獻

12. C# 視窗程式：簡介
    1. 視窗與 C# 程式
    2. 按鈕測試
    3. 文字型計算機
    4. 小字典
13. C# 視窗程式：時間驅動
    1. 碼表
    2. 小時鐘
    3. 用 Timer 控制動畫
14. C# 視窗程式：設計文字編輯器
15. C# 視窗程式：DataGridView 與 ListView 元件
    1. DataGridView 的使用
    2. 賣泡沫紅茶的系統 (Point of Sale, POS)
    3. 參考文獻
16. C# 視窗程式：繪圖功能
    1. 畫圖功能示範
    2. 小畫板
17. C# 網路程式：簡介、IP 與 URL
    1. 簡介
    2. IP 層的程式設計
    3. URL 、DNS 與網址剖析

    4. 結語
18. C# 網路程式：UDP 程式設計
    1. UDP 簡介
    2. 單向 UDP 訊息傳遞程式
    3. 參考文獻
19. C# 網路程式：TCP 程式設計
    1. TCP 簡介
    2. TCP 的程式架構
    3. TCP 單向的訊息傳遞程式
    4. TCP 雙向聊天程式
    5. TCP 多人聊天室 (視窗版)
20. C# 網路程式：HTTP 程式設計 (1) -- Server
    1. HTTP 協定簡介
    2. HelloServer: 永遠傳回 Hello 的 WebServer
    3. 完整的 Web Server
    4. 參考文獻
21. C# 網路程式：HTTP 程式設計 (2) -- Client
    1. HTTP Client 端程式簡介
    2. 單一網頁下載
    3. 網路爬蟲 WebCrawler 的設計
    4. Browser 的控制

22. C# 遊戲程式：XNA 遊戲引擎
    1. XNA 遊戲程式架構
    2. XNA 的 2D 打飛碟遊戲
    3. XNA 的 3D 遊戲架構
    4. XNA 的 3D 打飛碟遊戲
    5. 從 XNA 到 Unity3D
    6. 參考文獻
23. C# 程式設計：結語
24. 附錄：教學錄影

C# 程式基礎：簡介與開發環境

C# 是微軟所設計的一種物件導向語言，其設計理念受到 C 與 Java 語言的影響，採用類似 C 語言的語法，並使用類似 Java 語言的虛擬機架構，具備物件導向的能力，是微軟在其主力平台 .NET 上最重要的開發語言。

Visual Studio 安裝

要使用 C# 開發程式，必須安裝 Visual C# Express 或是 Visual Studio，其中 Visual C# Express 是免費的，您可以下列網址當中下載其安裝程式。

- Visual C# express 中文版 -- http://www.microsoft.com/visualstudio/cht/downloads

Visual C# Express 的安裝相當簡單，筆者在此不加以說明，但若您希望能在安裝前先預習一遍，可以參考下列網頁的安裝過程。

- 佳新的部落格 - 下載並安裝Visual C# 2008 Express中文版 -- http://jarsing.blogspot.com/2009/01/visual-c-2008-express.html

Visual Studio 是一個功能很強大的整合開發環境，包含視覺化設計模式、整合除錯環境、方便的程式編輯環境等等。因此，大部分人都會使用 Visual Studio 作為 C# 的開發工具。

目前 Visual Studio 可分為三個版本，企業版 (Enterprise Edition)、專業版 (Professional Edition) 以及簡潔版

(Express Edition)，其中企業版與專業版是需要付費購買的，而簡潔版則是免費的。通常一般的學習者使用簡潔版就綽綽有餘了，簡潔版的功能其實已經相當強大，除了缺少一些系統分析、團隊合作、以及企業報表等工具之外，其他的部分，像是視覺化設計模式、整合除錯環境、方便的程式編輯環境等，都已經包括在內，因此相當適合學生學習使用。

- Visual Studio 2012 Express for Windows Desktop -- http://www.microsoft.com/visualstudio/cht/downloads

當您下載並安裝完畢後，可以直接從微軟視窗系統中的「開始/所有程式/Microsoft Visual 2010 Express」這個路徑啟動簡潔版的開發環境。在這個環境中，使用編輯程式、除錯或使用視覺化設計模式，這是相當方便的一個工具。

命令列工具

微軟的 C# 語言的編譯器是 csc.exe 這個執行檔，這個檔案通常被安裝在 .NET的路徑下。舉例而言，在我的電腦中，就可以在下列路徑中找到這個檔案。

```
C:\WINDOWS\Microsoft.NET\Framework\v3.5
```

如果您安裝的是專業版，您可以在「開始/所有程式/Microsoft Visual Studio 2008/Visual Studio Tools/Visual Studio 2008 命令提示字元」這個功能表選項中，啟動 Visual Studio 的命令列工具。如此您可以使用命令列的方式，對程式進行編譯與執行的動作。

舉例而言，您可以使用「csc 檔案名稱」的指令，編譯特定的 C# 程式，像是下列指令就可以將 Hello.cs 編

譯為 Hello.exe。

```
csc Hello.cs
```

在某些時候，特別是開發網路程式的時候，命令列會比整合開發環境更方便使用，因此您可以自行決定何時應該用哪一種方式進行編譯。

由於微軟是目前軟體界最大的公司，並且極力推廣 C# 與 .NET 平台，這使得 C# 所支援的程式領域特別廣泛，除了命令列程式之外，C# 還常用在視窗、資料庫、網路、網頁、手機、遊戲等領域，其應用的廣泛性是其他語言所難以趕上的。

以下，我們分別就這些領域，逐一進行介紹。

視窗程式設計

C# 可用來撰寫 MS. Windows 作業系統上的視窗程式。微軟目前的視窗開發套件有兩種，比較舊但卻很成熟的一種稱為 Window Forms，比較新但卻較少人用的一種稱為 WPF (Windows Presentation Foundation)。

Window Forms 採用的是物件導向的視覺化設計元件，您可以用拖拉的方式，輕易的設計出視窗介面，然後利用事件驅動的方式，撰寫該事件的處理程式，像是滑鼠被按下，鍵盤被按下等都會觸發視窗系統中的事件。

WPF 的設計雖然也是物件導向式的，但是為了網路化的考量，微軟創造了一個稱為 XAML 的 XML 規

格，讓使用者可以撰寫 XAML 語法以創建使用者介面。這種做法與 Google 在 Android 平台上的做法有點類似，但是直到目前為止，這些規格仍然沒有受到瀏覽器的支援，因此用 XAML 設計使用者介面的必要性並不強烈，筆者仍建議採用 Window Forms 撰寫程式。

資料庫

微軟設計的開發工具，通常都會極力支援自家的產品，因此在 C# 當中最容易使用的是MS. SQL 與 Access 資料庫。但是由於這兩個資料庫都是要收費的，因此對於經費有限的個人而言，並不適當。但是如果您有 MS. SQL 或 Access 等軟體，就會感覺到微軟在資料庫上的用心，因為這是微軟主要的獲利來源。

網路

微軟的 .NET 平台除了支援傳統的 TCP/IP 網路基礎函式庫 Socket 之外，還設計了許多新的網路物件，像是 HTTP 的 WebRequest 等，這些物件可以讓程式設計者更省力的設計出網路程式。但是以筆者觀點，Socket 函式庫仍然是最重要的，因為使用 Socket 函式庫可以讓你清楚的理解網路程式的運作原理，直接透過 TCP/IP 掌握通訊程式的精隨。

網頁

微軟的網頁伺服器 IIS (Internet Information Server) 當中，所使用的開發環境稱為 ASP.NET，這是從過去的 ASP (Active Server Pages) 所延伸而來的。在 ASP.NET 當中支援了 C# 與 VB 等兩種開發語言，您可以輕易的使用 Visual Studio 進行 ASP.NET 的程式開發。

手機

微軟的手機從 Smart Phone 開始，經過 Windows Mobile、Windows Phone 一直到現在的 Windows 8，一開始帶出了智慧型手機的概念，但是卻受到 iPhone 與 Android 的夾殺，目前仍然無法成為智慧型手機與平版的主流。

不過微軟最強的部份是在桌上型電腦，如果能將手機、平板、遊戲機 XBox、筆電與桌上型電腦的生態系建構好，或許還有機會在市場上與 Apple, Google 等公司一爭長短。

遊戲

微軟在 2007 年推出了 XNA 遊戲開發平台，讓程式設計者可以利用 C# 語言開發遊戲程式，並且可以將這些遊戲放到 PC、XBOX 與 Zune 等裝置上執行，這對想要學習遊戲程式設計的人而言，是一個很好的開發平台。在 XNA 出現之前，遊戲公司都必須購買昂貴的遊戲設計軟體，以便開發遊戲程式。因此遊戲程式成了遊戲公司人員的商業秘密，但是在 XNA 出現之後，個人或者工作室都可以利用 C# 語言，直接開發出遊戲程式，而不需要購買那些昂貴的設計軟體。這對想學習遊戲程式設計的人而言是一個很好的消息，遊戲的開發因 XNA 而變得普及了。

要撰寫 XNA 遊戲程式，您必須安裝 XNA Game Studio 套件於 Visual C# Express 當中，您可以從下列 MSDN 網址中取得該套件。

- http://www.microsoft.com/downloads/details.aspx?FamilyID=80782277-d584-42d2-8024-893fcd9d3e82&displaylang=en

跨平台的考量

假如您希望讓 C# 程式在 UNIX/Linux/FreeBSD/MAC OS X 等平台上執行，也可以採用 Novell 公司所主導的 Mono 計畫，該計畫已經發展出一套跨平台的函式庫，讓您可以輕易的將 C# 程式放到非微軟的平台執行，Mono 計畫的網址如下。

- http://www.mono-project.com/

最新的發展

C# 語言在 3.0 版當中，加入了許多方便的新語法，像是匿名函數、資料查詢語言 Linq 等等，這些新功能讓 C# 語言超越了 Java ，成為簡單又強大的語言，有興趣的讀者可以觀看下列網頁當中的說明，該文章對 C# 3.0 的功能有簡單且扼要的介紹。

- 搖擺天秤的程式開發日誌： http://richielin-programer.blogspot.com/2008/02/visual-c-30.html

結語

雖然我並不是微軟的擁護者，甚至還有點反微軟的傾向，但是我仍然選擇了用 C# 為主要的開發語言。原因是 C# 的用途相當廣泛，支援的體系很完整，Visual C# Express 也很好用，而且我是個實用主義者。

我需要撰寫 Windows 當中的視窗程式、網路程式與遊戲程式，因此我使用 C# 與 Visual C# Express。

在本文中，我們簡單介紹了 C# 程式語言開發環境 - Visual C# Express 與 Visual Studio 等工具的安裝與使用方式。當然，我們無法完整的介紹這些工具的所有功能，只能將最常用的幾種方式簡要的說明一下，以幫助初學者能順利的進入 C# 程式開發的領域。

參考文獻

- 賴榮樞 的軟體資訊誌: Hello C# 編譯執行 console 程式, 2008/01 -- http://www.goodman-lai.idv.tw/2008/01/hello-c-console_3469.html

C# 程式基礎：變數與運算式

簡介

以下是一個完整的 C# 程式 (Hello.cs)，可以印出 Hello !。

```
using System;                       //引用 System 函式庫

public class Hello                  //宣告類別 Hello
{
  public static void Main()         //類別 Hello 的主程式
  {
    Console.WriteLine("Hello !");   //印出 Hello !
  }
}
```

如果您將上述程式加入到 Visual C# Express 的空專案中，就可以直接執行之。如果您會使用命令列的 csc 編譯器，就可以用下列指令將該程式編譯成執行檔 Hello.exe，然後您就可以在命令列中輸入 Hello 指令執行該程式。

```
C:\>csc Hello.cs
C:\>Hello
Hello!
```

對於初學程式設計的人而言，我們還需要很長的一段時間才能完全理解上述程式中每個指令的意義，先讓我們從最基本的變數與型態開始熟悉 C# 語言。

變數與型態

變數是程式用來儲存資料的地方，程式透過變數存取資料，並透過指定敘述 (Assign) 搬移資料。舉例而言，假如我們希望設定變數 X 為 3，可以用 X=3 的運算式，若我們希望將 Y 的內容複製給 X ，則可以用 X=Y 這個指定敘述。

C# 是一個強型態的語言，每個變數都必須要指定型態，C# 中常用的基本型態包含字元 (char)、整數 (int)、實數 (float)、字串(string) 、布林 (bool) 等，以下是這些基本型態的一些範例。

```
int a = 3, b = a;         // 整數
double pi = 3.14;         // 浮點數
char c = 'H';             // 字元
string msg = "Hello!";    // 字串
bool x = true, y = false; // 布林 (邏輯)
string str = "a=" + a + " b=" + b;
```

```
Console.Write(str);
```

- 運算式 程式語言當中的指定運算式，通常用 = 符號代表，其意義與數學當中的等號不同，請看下列範例。

```
int x = 3, y = 5;
x = y;
```

在上述範例中，x, y 分別被設定為 3 與 5，因此若以數學式的看法, x=y 是不成立的，但是在 C# 當中，若我們想判斷 X 與 Y 是否相等，應該用兩個等號的 x==y 表達。與此不同的是，上述範例中的 x=y 是指定的意思，也就是將 y 當中的 5 複製一份給 x，於是 x 的值也就變成 5 了。

C# 語言中的加減乘除等運算，基本上與數學的定義類似，因此，您可以寫出下列的運算式，該運算是依照先乘除後加減的順序，結果會設定 x 的值為 19。

```
int x = 3 * 5 + 8 / 2;
```

C# 語言承襲了 C 語言的許多語法，遞增運算 (x++), 移位運算 (x << 2), 邏輯運算 (&& , | |, !) 等，其語法都與 C 語言一致。

遞增運算 ++ 的意義，是將該變數加上 1，而遞減運算 -- 的意義，則是將該變數減去 1，所以 x++ 代表將變數 x 加上 1，等同於 x=x+1 的結果，而 x-- 代表將變數 x 減去 1，等同於 x = x - 1 的結果。

移位運算 < < 的意義，代表左移，而 > > 則是右移的意思。舉例而言，x < < 2 代表將 x 視為位元串，並且向左移兩位。因此若 x 是一個值為 2 的整數 (int) ，由於其二進位表達式為 00000010。所以當我們執行運算式 x = x< <2 後，x 將會變成 00001000，也就是十進位的 8，請看下列範例。

```
int x = 2;    // x = 00000010 = 2
x = x << 2;   // x = 00001000 = 8
```

邏輯運算 ! 是邏輯反閘 (not) 的意思，此運算會將真假值顛倒，也就是 !true = false, !false = true。

邏輯運算 && 是邏輯且 (AND) 的意思，只有當 X 與 Y 兩者均為真 (true) 時， X && Y 才會為真，否則 X&&Y 將會是假 (false) 值。

邏輯運算 | | 是邏輯或 (OR) 的意思，只要 X 與 Y 兩者當中有一個為真 (true) 時， X || Y 就會為真。

下列範例顯示了 C# 當中的邏輯運算式用法。

```
bool x = true, y = false;
bool a = x && y;   // a = true and false = false
bool b = x || y;   // b = true or false = true
bool c = !x;       // c = !true = false
```

C# 由於繼承了 C 語言的運算式語法，因此在位元邏輯上也採用 & 代表逐位元的 AND 運算，而 | 代表 逐

位元的 OR 運算。舉例而言，下列範例中的 x & y 之結果為 1， x | y 的結果為 7。

```
int x = 3, y = 5;
int a = x & y;      // a=00000011 AND 00000101 = 00000001 = 1
int b = x | y;      // b=00000011 OR 00000101 = 00000111 = 7
```

測驗

請寫出以下程式的輸出：

```
int x = 3, y = 5;
x = y;
x = x << 2;

bool t = true, u = false;
bool a = t && u;   // a = true and false = false
bool b = t || u;   // b = true or false = true
bool c = !u;       // c = !false = true

int d = x & y;      // d=00010100 AND 00000101 = 00000100 = 4
int e = x | y;      // e=00010100 OR  00000101 = 00010101 = 21
```

結語

萬事起頭難，學習程式設計尤其是如此。雖然到目前為止我們只學習了 C# 當中的基本運算與變數宣告，但這兩個主題卻是所有程式語言都必須要具備的。有了這些基礎，我們才能開始學習 if 判斷語句和 for, while 等迴圈語法。

參考文獻

- C# 程式設計人員參考
  - C# 教學課程：http://msdn.microsoft.com/zh-tw/library/aa288436(VS.71).aspx
  - Hello World 教學課程：http://msdn.microsoft.com/zh-tw/library/aa288463(VS.71).aspx
  - 命令列參數教學課程：http://msdn.microsoft.com/zh-tw/library/aa288457(VS.71).aspx

C# 程式基礎：流程控制

在結構化的程式語言中，流程控制是以判斷 (if, switch) 與迴圈 (for, while) 為主的。C# 也不例外，我們將在本文中介紹 C# 的流程控制語法，包含如何利用條件判斷語法控制程式的分支情況，以及用迴圈語法重複運行某些程式碼。

if 語法

C# 的條件判斷以 if 為主，語法完全繼承 C 語言的語法，其語法如下：

```
if EXP 1
  BLOCK 1
else if EXP 2
  BLOCK 2
...
else EXP K
  BLOCK K
```

舉例而言，假如我們想要判斷成績變數 score 是否及格，也就是 score 是否到達 60 分以上，則可以用下列語法。

```
if (score >= 60)
  Console.WriteLine("及格");
else
  Console.WriteLine("不及格");
```

更進一步的，假如我們想用程式判斷等第，其中 90 分以上為 A，80-90 之間為 B，70-80 之間為 C，70 以下為 D，那麼就可以用下列語法進行判斷。

```
if (score >=90)
  degree = "A";
else if (score >= 80)
  degree = "B";
else if (score >= 70)
  degree = "C";
else
  degree = "D";
```

如果我們將上述範例與 if 語句的語法對照起來，可以很清楚的看到 if 語句的語法結構，如以下範例所示。

```
if (score >=90)          // if EXP 1
```

```
  degree = "A";          //  BLOCK 1
else if (score >= 80)    // else if EXP 2
  degree = "B";          //  BLOCK 2
else if (score >= 70)    // else if (EXP 3
  degree = "C";          //  BLOCK 3
else                   // else EXP 4
  degree = "D";          //  BLOCK 4
```

for 語法

C# 當中的迴圈語法，包含 for, while, foreach 等，其中的 for 與 while 是由 C 繼承而來的，語法與 C 語言一致。而 foreach 的語法則是新創造的，其使用上比 for 語法更方便。

for 迴圈的語法如下所示，其中的 EXP1 是指定敘述，可以用來設定索引變數的初值，EXP2 是一個判斷條件，用來判斷是否應跳出迴圈，EXP3 則是累加條件，通常用來對索引變數進行累加 (++) 的動作。

```
for (EXP1; EXP2; EXP3)
  BLOCK;
```

舉例而言，假如我們想計算從 1 加到 100 的結果，就可以利用下列程式，不斷的將索引變數值 i 加入到總和變數 sum 當中，最後 sum 當中所儲存的就會是 1+2+...+100 的結果 5050。

```
int sum = 0;
for (int i=1; i<=100; i++)
  sum += i;
```

while 語法

while 迴圈的語法比 for 迴圈更簡單，其語法如下範例所示，其中的 EXP 是一個邏輯判斷式，用來判斷是否應該離開迴圈。在還沒離開之前，會不斷的重複執行 BLOCK 區塊。

```
while (EXP)
  BLOCK
```

同樣的，我們也可以利用 while 迴圈計算從 1 加到 100 的結果，其程式如以下範例所示。

```
int sum=0;
int i = 1;
while (i<=100)
{
  sum = sum + i;
  i++;
}
```

至於 foreach 迴圈，則是針對某個容器結構 (例如陣列) 當中的每個元素都巡迴執行一次，其語法我們將留待未來討論陣列的主題時再行說明。

結語

判斷與迴圈是結構化程式設計的兩大流程控制方法，有效的結合判斷與迴圈，就能產生變化無窮的程式，這正是程式設計精妙的地方，也是程式設計師必須要會的基本能力。

練習：if 範例

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

namespace ConsoleApplication1
{
   class Program
   {
      static void Main(string[] args)
      {
```

```
        // int score = 55;
        Console.Write("請輸入分數：");
        String scoreStr = Console.ReadLine();
        int score = int.Parse(scoreStr);
        if (score >= 60)
            Console.WriteLine("及格");
        else
            Console.WriteLine("不及格");
    }
  }
}
```

練習：while 範例

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

namespace ConsoleApplication1
```

```
{
    class Program
    {
        static void Main(string[] args)
        {
            int score = 55;
            if (score >= 60)
                Console.WriteLine("及格！");
            else
                Console.WriteLine("不及格！");

            while (score < 60)
            {
                Console.WriteLine("score=" + score + "=> 不及格！");
                score++;
            }
            Console.WriteLine("score=" + score + "=> 恭喜你！及格了！");
/*
            int i = 1;
            while (i <= 10)
            {
```

```
            Console.WriteLine("i=" + i);
            i++; // i = i + 1; // i++
        }
*/
        for (int i = 1; i <= 10; i+=2) // i+=2 => i=i+2
            Console.WriteLine("i=" + i);



    }
  }
}
```

C# 程式基礎：陣列

陣列是傳統程式設計上用來儲存很多個元素的資料結構。有了陣列，我們就可以將無數的資料放入程式當中以供存取、排序或搜尋。雖然在現代的語言當中都會有其他更方便的容器存在，像是字典 (Dictionary)，但這些容器也通常是利用陣列所實作出來的。

一維陣列

C# 當中的陣列是用中括號的方式宣告的。舉例而言，假如我們想宣告一個陣列可以儲存一年當中每個月的天數，我們可以用下列程式表示。

```
int[] days;
days = new int[12];
days[0]=31; days[1]=28; days[2] =31; days[3]=30;
days[4]=31; days[5]=30; days[6] =31; days[7]=31;
days[8]=30; days[9]=31; days[10]=30; days[11]=31;
```

請注意，由於 C# 繼承了 C 語言的習慣，陣列都是從 0 開始算起的。所以在上述程式中，我們用 days[0] 代表 1 月的天數，days[1] 代表 2 月的天數，以此類推，...。

然而，僅僅宣告每個月的天數，就需要這莫多指令，未免也太複雜了。因此，C# 語言允許我們直接設定

陣列的初值，以下是我們用很簡潔的語法做到與上述範例相同的功能，但卻只要用一行就夠了。

```
int[] days = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
```

有了陣列之後，我們就可以透過迴圈的方式，逐一存取陣列的內容。舉例而言，我們可以利用下列程式計算一年當中所有月份的天數總和。

```
int count = 0;
for (int i=0; i<12; i++)
  count += days[i];
```

如果您經常寫程式，就會覺得上述範例中的 for 迴圈語法仍然不夠簡潔，因為我們總是要寫 (i=0; i< ...; i++)。所以 C# 當中設計了一種更簡潔的迴圈語法，用來巡訪陣列或容器當中的每個元素，這個語法稱為 foreach。

我們可以用 foreach 寫出計算天數總和的程式如以下範例所示，在本範例中我們用 d 變數取出每個月的天數後，將之加入 count 變數以便計算總和。

```
int count = 0;
foreach (int d in days)
  count += d;
```

作業：請寫出內積的版本 int c = innerProduct(a,b)

解答：

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

namespace ArrayTest
{
   class Program
   {
      static void Main(string[] args)
      {
         int[,] a1 = { { 1, 2 }, { 3, 4 } };
         int[,] b1 = { { 3, 4 }, { 5, 6 } };
         int[,] c1 = new int[2, 2];

         int[] a = { 1, 2, 3, 4, 5 };
         int[] b = { 3, 3, 3, 3, 3 };
```

```
        int[] c = new int[5];

        for (int i = 0; i < a.Length; i++)
        {
            c[i] = a[i] + b[i];
        }

        int[] d = new int[5];
        arrayAdd2(a, b, d);

        int[] e = arrayAdd(a, b);

        printArray(a);
        printArray(b);
        printArray(c);
        printArray(d);
        printArray(e);
    }

    static void printArray(int[] x)
    {
```

```
    for (int i = 0; i < x.Length; i++)
        Console.Write(x[i] + " ");
    Console.WriteLine();
}

static void arrayAdd2(int[] x, int[] y, int[] z)
{
    for (int i = 0; i < x.Length; i++)
    {
        z[i] = x[i] + y[i];
    }
}

static int[] arrayAdd(int[] x, int[] y)
{
    int[] z= new int[x.Length];
    for (int i = 0; i < x.Length; i++)
    {
        z[i] = x[i] + y[i];
    }
    return z;
```

```
    }
  }
}
```

二維陣列

以下我們用「矩陣加法」作為範例，以說明 C# 二維陣列的用法。

範例：請寫出矩陣的加法 int[,] c = matrixAdd(a,b)

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

namespace ArrayTest
{
  class Program
  {
    static void Main(string[] args)
    {
```

```
        int[] a = { 1, 2, 3, 4, 5 };
        int[] b = { 3, 3, 3, 3, 3 };
        int[] c = new int[5];

        for (int i = 0; i < a.Length; i++)
        {
            c[i] = a[i] + b[i];
        }

        int[] d = new int[5];
        arrayAdd2(a, b, d);

        int[] e = arrayAdd(a, b);

        printArray(a);
        printArray(b);
        printArray(c);
        printArray(d);
        printArray(e);
    }
```

```
static void printArray(int[] x)
{
    for (int i = 0; i < x.Length; i++)
        Console.Write(x[i] + " ");
    Console.WriteLine();
}

static void arrayAdd2(int[] x, int[] y, int[] z)
{
    for (int i = 0; i < x.Length; i++)
    {
        z[i] = x[i] + y[i];
    }
}

static int[] arrayAdd(int[] x, int[] y)
{
    int[] z= new int[x.Length];
    for (int i = 0; i < x.Length; i++)
    {
        z[i] = x[i] + y[i];
```

```
        }
        return z;
      }
   }
}
```

進階題：矩陣的轉置

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

namespace ConsoleApplication1
{
   class Program
   {
      static void Main(string[] args)
      {
         double[,] a = { { 1, 2, 3 }, { 3, 4, 5 } };
         double[,] b = { { 3, 4, 2 }, { 5, 6, 1 } };
         double[,] at = transpose(a);
```

```
    Console.WriteLine("a.GetLength(0)=" + a.GetLength(0));
    Console.WriteLine("a.GetLength(1)=" + a.GetLength(1));
    printMatrix("a", a);
    printMatrix("b", b);
    printMatrix("at", at);
}

static void printMatrix(String name, double[,] x)
{
    Console.WriteLine("==========" + name + "==========");
    for (int i = 0; i < x.GetLength(0); i++)
    {
        for (int j = 0; j < x.GetLength(1); j++)
        {
            Console.Write(x[i,j]+" ");
        }
        Console.WriteLine();
    }
}
```

```
        static double[,] transpose(double [,] x)
        {
            double[,] t = new double[x.GetLength(1), x.GetLength(0)];
            for (int i = 0; i < x.GetLength(0); i++)
            {
                for (int j = 0; j < x.GetLength(1); j++)
                {
                    t[j, i] = x[i, j];
                }
            }
            return t;
        }

    }
}
```

加分題：請寫出矩陣的乘法 int[,] c = matrixMul(a,b)

參考文獻

- C# 教學課程
  - 陣列教學課程 - http://msdn.microsoft.com/zh-tw/library/aa288453(VS.71).aspx

- 索引子教學課程 - http://msdn.microsoft.com/zh-tw/library/aa288465(VS.71).aspx
- 索引屬性教學課程 - http://msdn.microsoft.com/zh-tw/library/aa288464(VS.71).aspx

C# 程式基礎：函數

簡介

C# 分為靜態函數與成員函數兩類，靜態函數附屬於類別，呼叫時可以直接指定類別名稱即可。成員函數附屬於物件，呼叫時必須透過物件變數進行呼叫。

通常函數會接收到一些呼叫端傳入的參數。C# 的參數有數種傳遞方式，包含傳值參數 (call by value)，傳址參數 (call by reference) 等，基本型態的參數，像是 int, double, char, ... 等，預設都使用傳值的方式，而物件形態的參數，像是 StringBuilder，陣列等，預設則是使用傳址的方式。

以下是一個 C# 的靜態函數範例，其中的 square 就是靜態函數，其功能是將傳入的數值進行平方動作後傳回，這是一個傳值參數的範例。

```
using System;

class Func1
{
   public static void Main(string[] args)
   {
      int x = square(5);
```

```
        Console.WriteLine("x="+x);
    }

    public static int square(int n)
    {
        return n*n;
    }
}
```

函數的參數

陣列也可以用來作為函數的參數，由於陣列的傳遞採用傳址的方式，因此在函數中對陣列的修改將會是永久性的修改，離開函數後並不會恢復成原先的數值。以下範例中的 add 函數用來將兩個二維陣列 (x,y) 相加，然後將結果放入 z 當中，print 函數則是將傳入的陣列 x 印出來。必須注意的是，對於二維陣列而言，要取得陣列的第一維元素個數 (列數)，可用 GetLength(0)，要取得陣列的第二維元素個數 (行數)，必須使用 GetLength(1)。

```
using System;

class Func2
{
```

```
public static void Main(string[] args)
{
    int[,] a = {{1,2}, {3,4}};
    int[,] b = {{5,6}, {7,8}};
    int[,] c = new int[2,2];
    add(a, b, c);
    Console.WriteLine("=======a=======");
    print(a);
    Console.WriteLine("=======b=======");
    print(b);
    Console.WriteLine("=======c=======");
    print(c);
}

public static void add(int[,] x, int[,] y, int[,] z)
{
    for (int i=0; i<z.GetLength(0); i++)
    for (int j=0; j<z.GetLength(1); j++)
    {
     z[i,j] = x[i,j] + y[i,j];
    }
```

```
    }

    public static void print(int[,] x)
    {
        for (int i=0; i<x.GetLength(0); i++)
        {
        for (int j=0; j<x.GetLength(1); j++)
         Console.Write(x[i,j]+" ");
        Console.WriteLine();
    }
    }
}
```

練習

```
using System;

namespace ConsoleApplication1
{
    class Program
    {
```

```
        static void Main(string[] args)
        {
            sum(10);
            sum(20);
            sum(30);
        }

        static void sum(int n)
        {
            int s = 0;
            for (int i = 1; i <= n; i++)
            {
                s = s + i;
            }
            Console.WriteLine("s = " + s);
        }
    }
}
```

C# 程式基礎：物件

C# 是很好的物件導向語言，而且微軟的 .NET Framework 函式庫設計得相當優美，這使得 C# 的魅力相當大。

傳統的程式設計會將資料與程式分開，但是在物件導向的概念當中，資料與程式被合併成一個結構，這個結構就稱為物件。

一個物件可以包含資料部分 (資料成員) 與函數部分 (函數成員)，函數成員可以對資料成員進行操作，以下是一個 C# 的物件範例，該範例中定義了一個人員 (Person1) 的結構，該結構包含兩個資料成員 (name, weight) 與一個成員函數 (checkWeight)，該函數會檢查人員結構的體重 (weight) 看看是重還是輕。另外，還包含了一個建構函數 Person1() ，這個建構函數可以讓使用者在建立物件時順便將參數傳入，這是物件導向的一種常見手法。

```
using System;

class Object1 {
   public static void Main(String[] args) {
      Person1 p1, p2;
      p1 = new Person1("大雄", 50);
      p2 = new Person1("胖虎", 80);
```

```
        p1.checkWeight();
        p2.checkWeight();
        p2.weight = 68;
        p1.checkWeight();
        p2.checkWeight();
    }
}

class Person1 {
    public string name;
    public int weight;

    public Person1(string pName, int pWeight) {
        name   = pName;
        weight = pWeight;
    }

    public void checkWeight()
    {
        Console.Write(name+"體重 "+weight+" 公斤,");
        if (weight < 70)
```

```
        Console.WriteLine("很苗條!");
      else
        Console.WriteLine("很穩重!");
    }
}
```

執行結果

```
D:\myweb\teach\CSharpProgramming>csc Object1.cs
Microsoft (R) Visual C# 2008 Compiler version 3.5.30729.1
for Microsoft (R) .NET Framework version 3.5
Copyright (C) Microsoft Corporation. All rights reserved.


D:\myweb\teach\CSharpProgramming>Object1
大雄體重 50 公斤,很苗條!
胖虎體重 80 公斤,很穩重!
大雄體重 50 公斤,很苗條!
胖虎體重 68 公斤,很苗條!
```

封裝

在傳統的結構化程式設計 (像是 C, Fortran, Pascal) 當中，我們用函數來處理資料，但是函數與資料是完全區隔開來的兩個世界。然而，在物件導向當中，函數與資料被合為一體，形成一種稱為物件的結構，我們稱這種將函數與資料放在一起的特性為「封裝」。

以下我們將以矩形物件為範例，說明 C# 當中的封裝語法，以及物件導向中的封裝概念。

範例一：封裝 -- 將函數與資料裝入物件當中

```
using System;

class Rectangle
{
   double width, height;

   double area()
   {
      return width * height;
   }

   public static void Main(string[] args)
   {
      Rectangle r = new Rectangle();
```

```
        r.width = 5.0;
        r.height = 8.0;
        Console.WriteLine("r.area() = " + r.area());
    }
}
```

執行結果

```
r.area() = 40
```

修改進化版

由於上述物件與主程式混在一起，可能容易造成誤解，因此我們將主程式獨立出來，並且新增了一個 r2 的矩形物件，此時程式如下所示。

```
using System;

class Rectangle
{
    public double width, height;

    public double area()
```

```
    {
        return width * height;
    }
}

class Test
{
    public static void Main(string[] args)
    {
        Rectangle r = new Rectangle();
        r.width = 5.0;
        r.height = 8.0;
        Console.WriteLine("r.area() = " + r.area());

        Rectangle r2 = new Rectangle();
        r2.width = 7.0;
        r2.height = 4.0;
        Console.WriteLine("r2.area() = " + r2.area());
    }
}
```

執行結果：

```
r.area() = 40
r2.area() = 28
```

範例二：加上建構函數

```
using System;

class Rectangle
{
   double width, height;

   public Rectangle(double w, double h)
   {
      width = w;
      height = h;
   }

   public double area()
   {
      return width * height;
```

```
    }

    public static void Main(string[] args)
    {
        Rectangle r = new Rectangle(5.0, 8.0);
        Console.WriteLine("r.area() = " + r.area());
    }
}
```

執行結果

```
r.area() = 40
```

範例二：加上建構函數 -- 同時有 0 個與兩個引數

```
using System;

class Rectangle
{
    public double width, height;

    public Rectangle() { }
```

```
    public Rectangle(double w, double h)
    {
        width = w;
        height = h;
    }

    public double area()
    {
        return width * height;
    }
}

class Test
{
    public static void Main(string[] args)
    {
        Rectangle r = new Rectangle(5.0, 8.0);
        Console.WriteLine("r.area() = " + r.area());

        Rectangle r2 = new Rectangle();
```

```
        r2.width = 7.0;
        r2.height = 4.0;
        Console.WriteLine("r2.area() = " + r2.area());
    }
}
```

習題：向量物件

```
using System;

class Vector
{
    double[] a;

    public Vector(double[] array)
    {
        a = new double[array.GetLength(0)];
        for (int i = 0; i < a.GetLength(0); i++)
        {
            a[i] = array[i];
        }
    }
```

```
    public Vector add(Vector v2)
    {
        Vector rv = new Vector(v2.a);
        for (int i = 0; i < rv.a.GetLength(0); i++)
        {
            rv.a[i] = this.a[i] + v2.a[i];
        }
        return rv;
    }

    public void print()
    {
        for (int i = 0; i < a.GetLength(0); i++)
        {
            Console.Write(a[i] + " ");
        }
        Console.WriteLine();
    }
}
```

```
class Test
{
   public static void Main(string[] args)
   {
      Vector v1 = new Vector(new double[] { 1.0, 2.0, 3.0 });
      Vector v2 = new Vector(new double[] { 4.0, 5.0, 6.0 });
      Vector v3 = v1.add(v2);

      v1.print();
      v2.print();
      v3.print();
   }
}
```

進階練習 1：加上內積的函數，並寫出呼叫範例。 進階練習 2：寫出矩陣物件 (有加法、乘法) (方法一：用二維陣列、方法二：用 Vector)。

習題：矩陣物件

繼承

範例一：矩形、圓形繼承形狀（Shape）

```
using System;

class Shape
{
    public virtual double area() { return 0.0; }
}

class Rectangle : Shape
{
    double width, height;

    public Rectangle(double w, double h)
    {
        width = w;
        height = h;
    }

    public override double area()
    {
        return width * height;
```

```
    }
}

class Circle : Shape
{
    double r;

    public Circle(double r)
    {
        this.r = r;
    }

    public override double area()
    {
        return 3.1416 * r * r;
    }
}

class Test
{
    public static void Main(string[] args)
```

```
    {
        Shape s = new Shape();
        Console.WriteLine("s.area() = " + s.area());

        Rectangle r = new Rectangle(5.0, 8.0);
        Console.WriteLine("r.area() = " + r.area());

        Circle c = new Circle(2);
        Console.WriteLine("c.area() = " + c.area());
    }
}
```

執行結果：

```
s.area() = 0
r.area() = 40
c.area() = 12.5664
```

範例二：加入厚度與體積函數（在子物件使用父物件的欄位）

執行結果

```
s.area() = 0
r.area() = 40
r.volume() = 80
c.area() = 12.5664
c.volume() = 37.6992
c.volume() = 62.832
```

程式碼

```
using System;

class Shape
{
    public double thick;
    public virtual double area() { return 0.0; }
}

class Rectangle : Shape
{
    double width, height;
```

```
    public Rectangle(double w, double h)
    {
        width = w;
        height = h;
        thick = 2.0;
    }

    public override double area()
    {
        return width * height;
    }

    public double volume()
    {
        return width * height * thick;
    }
}

class Circle : Shape
{
    double r;
```

```
    public Circle(double r)
    {
        this.r = r;
        thick = 3.0;
    }

    public override double area()
    {
        return 3.1416 * r * r;
    }

    public double volume()
    {
        return area() * thick;
    }
}

class Test
{
    public static void Main(string[] args)
```

```
    {
        Shape s = new Shape();
        Console.WriteLine("s.area() = " + s.area());

        Rectangle r = new Rectangle(5.0, 8.0);
        Console.WriteLine("r.area() = " + r.area());
        Console.WriteLine("r.volume() = " + r.volume());

        Circle c = new Circle(2);
        Console.WriteLine("c.area() = " + c.area());
        Console.WriteLine("c.volume() = " + c.volume());
        c.thick = 5;
        Console.WriteLine("c.volume() = " + c.volume());
    }
}
```

範例三：將 volume() 函數提到 Shape 物件中

```
using System;

class Shape
{
```

```
    public double thick;
    public virtual double area() { return 0.0; }
    public double volume()
    {
        return area() * thick;
    }
}

class Rectangle : Shape
{
    double width, height;

    public Rectangle(double w, double h)
    {
        width = w;
        height = h;
        thick = 2.0;
    }

    public override double area()
    {
```

```
        return width * height;
    }
}

class Circle : Shape
{
    double r;

    public Circle(double r)
    {
        this.r = r;
        thick = 3.0;
    }

    public override double area()
    {
        return 3.1416 * r * r;
    }
}

class Test
```

```
{
    public static void Main(string[] args)
    {
        Shape s = new Shape();
        Console.WriteLine("s.area() = " + s.area());

        Rectangle r = new Rectangle(5.0, 8.0);
        Console.WriteLine("r.area() = " + r.area());
        Console.WriteLine("r.volume() = " + r.volume());

        Circle c = new Circle(2);
        Console.WriteLine("c.area() = " + c.area());
        Console.WriteLine("c.volume() = " + c.volume());
        c.thick = 5;
        Console.WriteLine("c.volume() = " + c.volume());
    }
}
```

習題：人、學生與老師

請定義「人、學生與老師」等三個類別，其中的人有「姓名、性別、年齡」三個欄位，學生另外有「學號、年級」等兩個欄位，老師另外有「職等」的欄位，所有物件都有 print() 函數，可以將該物件的所有欄

位印出。

請建立一個具有 3 個學生與兩個老師的陣列，利用多型機制，呼叫 print 函數以便印出這 5 個人的基本資料，如下所示。

```
學生 -- 姓名：王小明，性別：男，年齡：20，學號：R773122456，年級：一年級
學生 -- 姓名：李小華，性別：女，年齡：19，學號：R773122432，年級：一年級
教師 -- 姓名：陳福氣，性別：男，年齡：40，職等：教授
學生 -- 姓名：黃大虎，性別：男，年齡：22，學號：R773122721，年級：四年級
教師 -- 姓名：李美女，性別：女，年齡：35，職等：助理教授
```

解答：

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

namespace Person
{
/*
 * 請定義「人、學生與老師」等三個類別，其中的人有「姓名、性別、年齡」三個欄位，
 * 學生另外有「學號、年級」等兩個欄位，老師另外有「職等」的欄位，所有物件都有
```

```
 * print() 函數，可以將該物件的所有欄位印出。
 *
 * 請建立一個具有 3 個學生與兩個老師的陣列，利用多型機制，呼叫 print 函數以便
 * 印出這 5 個人的基本資料，如下所示。
 *
 * 學生 -- 姓名：王小明，性別：男，年齡：20，學號：R773122456，年級：一年級
 * 學生 -- 姓名：李小華，性別：女，年齡：19，學號：R773122432，年級：一年級
 * 教師 -- 姓名：陳福氣，性別：男，年齡：40，職等：教授
 * 學生 -- 姓名：黃大虎，性別：男，年齡：22，學號：R773122721，年級：四年級
 * 教師 -- 姓名：李美女，性別：女，年齡：35，職等：助理教授
*/
    class Program
    {
        static void Main(string[] args)
        {
            Student s1 = new Student("王小明", "男", 20, "R773122456", 1);
            Student s2 = new Student("李小華", "女", 19, "R773122432", 1);
            Student s3 = new Student("黃大虎", "男", 22, "R773122721", 4);
            Person t1 = new Person("陳福氣", "男", 40);
            Person t2 = new Person("李美女", "女", 35);
```

```
        Person[] list = { s1, s2, t1, s3, t2 };
        foreach (Person p in list)
        {
            p.print();
            Console.WriteLine();
        }
    }
}

class Person
{
    String name;
    String sex;
    int age;

    public Person(String name, String sex, int age)
    {
        this.name = name;
        this.sex = sex;
        this.age = age;
    }
```

```
    public virtual Person print()
    {
        Console.Write("姓名："+name+" 性别："+sex+" 年龄："+age);
        return this;
    }
}

class Student : Person
{
    String id;
    int degree;

    public Student(String name, String sex, int age, String id, int degree)
        : base(name, sex, age)
    {
        this.id = id;
        this.degree = degree;
    }

    public override Person print()
```

```
        {
            Console.Write("學生 -- ");
            base.print();
            Console.Write(" 學號：" + id + " 年級：" + degree);
            return this;
        }
    }
}
```

多型

物件導向的多型機制，是指當兩個以上的類別繼承同一種父類別時，我們可以用父類別型態容納子類別的物件，真正進行函數呼叫時會呼叫到子類別的函數，此種特性稱之為多型。

以下是我們用 C# 實作的一個多型範例，在範例中，我們宣告了一個形狀類別，該類別具有一個 area() 函數可以計算該形狀的面積，然後我們又宣告了兩個子類別 Rectangle (矩形) 與 Circle (圓形)。我們將兩者放入到 shapes 陣列中，以便展示多型技巧，用父類別容器呼叫子類別的實體。

範例：形狀、矩形與圓形

```
using System;
```

```
class Shape
{
    public virtual double area() { return 0.0; }

    public static void Main(string[] args)
    {
        Rectangle r = new Rectangle(5.0, 8.0);
        Console.WriteLine("r.area() = " + r.area());

        Circle c = new Circle(3.0);
        Console.WriteLine("c.area() = " + c.area());

        Shape[] shapes = { r, c };
        for (int i = 0; i < shapes.Length; i++)
            Console.WriteLine("shapes[" + i + "].area()=" + shapes[i].area());
    }
}

class Rectangle : Shape
{
    double width, height;
```

```
    public Rectangle(double w, double h)
    {
        width = w;
        height = h;
    }

    public override double area()
    {
        return width * height;
    }
}

class Circle : Shape
{
    double r;

    public Circle(double r)
    {
        this.r = r;
    }
```

```
    public override double area()
    {
        return 3.1416 * r * r;
    }
}
```

執行結果

```
r.area() = 40
c.area() = 28.2744
shapes[0].area()=40
shapes[1].area()=28.2744
```

範例：使用抽象父型態

```
using System;

class ShapeTest
{
    public static void Main(string[] args)
    {
```

```
        Rectangle r = new Rectangle(5.0, 8.0);
        Console.WriteLine("r.area() = " + r.area());

        Circle c = new Circle(3.0);
        Console.WriteLine("c.area() = " + c.area());

        Shape[] shapes = { r, c };
        for (int i = 0; i < shapes.Length; i++)
            Console.WriteLine("shapes[" + i + "].area()=" + shapes[i].area());
    }
}

abstract class Shape
{
    public abstract double area();

}

class Rectangle : Shape
{
    double width, height;
```

```
    public Rectangle(double w, double h)
    {
        width = w;
        height = h;
    }

    public override double area()
    {
        return width * height;
    }
}

class Circle : Shape
{
    double r;

    public Circle(double r)
    {
        this.r = r;
    }
```

```
    public override double area()
    {
        return 3.1416 * r * r;
    }
}
```

執行結果： r.area() = 40 c.area() = 28.2744 shapes[0].area()=40 shapes[1].area()=28.2744

範例：使用介面

```
using System;

class ShapeTest
{
    public static void Main(string[] args)
    {
        Rectangle r = new Rectangle(5.0, 8.0);
        Console.WriteLine("r.area() = " + r.area());

        Circle c = new Circle(3.0);
        Console.WriteLine("c.area() = " + c.area());
```

```
        Shape[] shapes = { r, c };
        for (int i = 0; i < shapes.Length; i++)
            Console.WriteLine("shapes[" + i + "].area()=" + shapes[i].area());
    }
}

interface Shape
{
    double area();

}

class Rectangle : Shape
{
    double width, height;

    public Rectangle(double w, double h)
    {
        width = w;
        height = h;
```

```
    }

    public double area()
    {
        return width * height;
    }
}

class Circle : Shape
{
    double r;

    public Circle(double r)
    {
        this.r = r;
    }

    public double area()
    {
        return 3.1416 * r * r;
    }
```

```
}
```

執行結果：

```
r.area() = 40
c.area() = 28.2744
shapes[0].area()=40
shapes[1].area()=28.2744
```

範例：較完整複雜的版本

```
using System;

class Shape
{
    public double thick = 0.0;
    public virtual double area() { return 0.0; }
    public double volume()
    {
        return area() * thick;
    }
```

```
    public override String ToString()
    {
        return "Shape: thick=" + thick+" area="+area()+" volume="+volume();
    }
}

class Rectangle : Shape
{
    double width, height;

    public Rectangle(double w, double h)
    {
        width = w;
        height = h;
        thick = 2.0;
    }

    public override String ToString()
    {
        return base.ToString()+" Rectangle:width=" + width + " height=" + height;
    }
```

```
    public override double area()
    {
        return width * height;
    }
}

class Circle : Shape
{
    double r;

    public Circle(double r)
    {
        this.r = r;
        thick = 3.0;
    }

    public override String ToString()
    {
        return base.ToString() + " Circle:r=" + r + " thick=" + thick;
    }
```

```
    public override double area()
    {
        return 3.1416 * r * r;
    }
}

class Test
{
    public static void Main(string[] args)
    {
        Shape s = new Shape();
        Console.WriteLine("s.area() = " + s.area());

        Rectangle r = new Rectangle(5.0, 8.0);
        Console.WriteLine("r.area() = " + r.area());
        Console.WriteLine("r.volume() = " + r.volume());

        Circle c = new Circle(2);
        Console.WriteLine("c.area() = " + c.area());
        Console.WriteLine("c.volume() = " + c.volume());
```

```
        c.thick = 5;
        Console.WriteLine("c.volume() = " + c.volume());

        Shape[] array = new Shape[] { s, r, c, r };
        foreach (Shape o in array)
        {
//          Console.WriteLine("o.area()=" + o.area() +" o.volume()="+o.volume());
            Console.WriteLine(o.ToString());
        }
    }
}
```

參考文獻

- C# 教學課程
  - 結構教學課程 - http://msdn.microsoft.com/zh-tw/library/aa288471(VS.71).aspx
  - 使用者定義轉換教學課程 - http://msdn.microsoft.com/zh-tw/library/aa288476(VS.71).aspx
  - 運算子多載化教學課程 - http://msdn.microsoft.com/zh-tw/library/aa288467(VS.71).aspx

C# 程式基礎：例外處理

C# 支援例外處理機制，當有任何的例外錯誤發生時，程式會立刻中斷，然後跳出到外層。此時，如果有任何例外處理的程式 (try ... catch) 位於外層，就會接到這個例外，並可以即時處理之。否則，該例外會一直被往外丟，假如都沒有被處理，則程式將被迫中斷，系統會自行輸出例外訊息。

範例：引發例外

以下是一個會引發例外的程式，由於 a/b = 3/0 會導致嘗試以零除 (System.DivideByZeroException) 的例外，但這個例外又沒有被任何的 try ... catch 段落所處理，因此整個程式會中斷並輸出錯誤訊息。

```
using System;

class Try1
{
   public static void Main(string[] args)
   {
      int a = 3, b = 0;
      Console.WriteLine("a/b=" + a/b);
   }
}
```

執行結果

```
D:\myweb\teach\CSharpProgramming>csc Try1.cs
Microsoft (R) Visual C# 2008 Compiler version 3.5.30729.1
for Microsoft (R) .NET Framework version 3.5
Copyright (C) Microsoft Corporation. All rights reserved.



D:\myweb\teach\CSharpProgramming>Try1

未處理的例外狀況: System.DivideByZeroException: 嘗試以零除。
   於 Try1.Main(String[] args)
```

範例：捕捉例外

要處理例外可以用 try...catch 語句，以下範例就利用 try { ... } catch (DivideByZeroException ex) 捕捉了上述的除以零之例外，您可以在 catch 段落中進行例外處理後，再決定要如何繼續執行程式。(本範例中只單純的提示被除數不可為零)。

```
using System;
```

```
class Try2
{
   public static void Main(string[] args)
   {
      try
      {
       int a = 3, b = 0;
       Console.WriteLine("a/b=" + a / b);
      }
      catch (DivideByZeroException ex)
      {
       Console.WriteLine("被除數不可為 0 !\n"+ex);
      }
   }
}
```

執行結果

```
D:\myweb\teach\CSharpProgramming>csc Try2.cs
Microsoft (R) Visual C# 2008 Compiler version 3.5.30729.1
```

```
for Microsoft (R) .NET Framework version 3.5
Copyright (C) Microsoft Corporation. All rights reserved.



D:\myweb\teach\CSharpProgramming>Try2
被除數不可為 0 !
System.DivideByZeroException: 嘗試以零除。
   於 Try2.Main(String[] args)
```

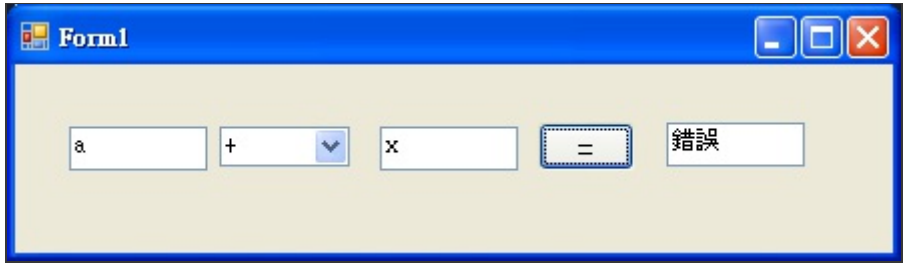
視窗版的例外處理

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace WError
{
```

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        try
        {
            double x = double.Parse(textBox1.Text);
            double y = double.Parse(textBox2.Text);
            String op = comboBox1.Text;
            double result = 0.0;
            switch (op)
            {
                case "+": result = x + y; break;
                case "-": result = x - y; break;
                case "*": result = x * y; break;
                case "/": result = x / y; break;
```

```
                default: throw new Exception("出現錯誤!");
            }
            textBox3.Text = result.ToString();
        }
        catch
        {
            textBox3.Text = "錯誤";
        }
    }
  }
}
```

執行結果：



圖、視窗版的例外處理

C# 程式基礎：資料結構

容器函式庫

一般型別	泛型型別	說明
ArrayList	List< >	串列，以陣列實作的串列結構
Queue	Queue< >	佇列，先進先出的結構
Stack	Stack< >	堆疊，後進先出的結構
Hashtable	Dictionary< >	雜湊表格，快速用 key 查找 value
StoredList	StoredList< >	排序串列，使用排序與二分搜尋法的結構
ListDictionary	Dictionary< >	字典，快速用 key 查找 value
HybridDictionary	Dictionary< >	小集合採用 ListDictionary，集合變大時，會自動改用 Hashtable 的一種字典

OrderedDictionary	Dictionary< >	比SortedList類別多了一些功能的類別
SortedDictionary	StoredDictionary< >	鍵值一定是字串，用法跟 Hashtable 相似
NameValueCollection	Dictionary< >	NameValueCollection可以單一索引鍵對應多重值
DictionaryEntry	KeyValPair< >	字典中的一個項目，(Key, Value) 的結構
StringCollection	List<String>	用法跟ArrayList相似
StringDictionary	Dictionary<String>	字串字典

- 上層結構：Collection
- 容器物件：Array, ArrayList, HashTable, SortedList
- 位元容器：BitArray, BitVector32,
- 泛型容器：NameValueCollection, Dictionary

容器的使用

範例：

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

namespace ConsoleApplication1
{
   class Program
   {
      static void Main(string[] args)
      {
         ArrayList a = new ArrayList();
         a.Add("John");
         a.Add(1);
         a.Add(3.1416);
         a.RemoveAt(2);

         Object[] array = a.ToArray();
         Console.WriteLine(array);

         List<Object> list = new List<Object>();
```

```
        list.Add("John");
        list.Add("Mary");
        list.Add("George");
        list.Add("3.14159");
        list.Add(3.14159);

        foreach (Object o in list)
            Console.Write(o+" ");
        Console.WriteLine();
        char c = 'A';
//          Console.Write("{0:X}", (int) c);

        Hashtable h = new Hashtable();
        h.Add("John", "0977332415");
        h.Add("Mary", "0977342415");
        h.Add("George", "0977372416");
        h.Add("Peter", "0977332425");
        String gtel = (String) h["George"];
        Console.WriteLine("George Tel : "+gtel);

        Dictionary<String, String> d = new Dictionary<string, string>();
```

```
            d.Add("John", "0977332415");
            d.Add("Mary", "0977342415");
            d.Add("George", "0977372416");
            d.Add("Peter", "0977332425");
            gtel = d["George"];
            Console.WriteLine("George Tel : " + gtel);

        }

    }
}
```

C# 程式基礎：檔案處理

C# 當中的檔案主要以串流 (Stream) 的形式呈現，串流讀取器 (StreamReader) 與串流寫入器 (StreamWriter) 是兩個主要的檔案處理類別。另外像 File，FileInfo, DirectoryInfo 等，則是用來存取檔案屬性與資料夾的類別。而BufferedStream、FileStream、MemoryStream、NetworkStream 則是分別對應到緩衝、檔案、記憶體、網路等類型的串流物件。因此串流可以說是檔案與網路的共同介面。

讀取檔案

以下範例中的 fileToText() 函數，會將一個文字檔讀入放到字串中傳回，其方法是利用 StreamReader 物件，指定所要讀取的檔案，然後利用 readToEnd() 函數讀取整個文字檔，再用 Close() 函數關閉該檔案。

```
using System;
using System.IO;

class FileTest
{
   public static void Main(String[] args)
   {
      String text = fileToText(args[0]);
      Console.WriteLine(text);
```

```
    }

    public static String fileToText(String filePath)
    {
        StreamReader file = new StreamReader(filePath);
        String text = file.ReadToEnd();
        file.Close();
        return text;
    }
}
```

上述範例的執行結果如下所示，必須注意的是，Hello.txt 檔案必須存在，而且儲存成 Unicode 的 UTF8 格式，這是因為 C# 內部預設使用 Unicode 的編碼格式。如果希望讀取 Big5 (或 GB2312) 格式的檔案，必須在 StreamReader() 建構函數當中，指定StreamReader file = new StreamReader(filePath, System.Text.Encoding.GetEncoding("Big5")) ，如此 StreamReader 才會以 Big5 的編碼方式對檔案進行讀取，結果才不會變成亂碼。

```
D:\>csc FileTest.cs
Microsoft (R) Visual C# 2008 Compiler version 3.5.30729.1
for Microsoft (R) .NET Framework version 3.5
Copyright (C) Microsoft Corporation. All rights reserved.
```

```
D:\>FileTest Hello.txt
Hello !
你好 !
```

寫入檔案

```
using System;
using System.IO;

class FileTest
{
   public static void Main(String[] args)
   {
      String text1 = "Hello, C#!";
      String file = "Hello.txt";
      textToFile(file, text1);
      String text2 = fileToText(file);
      Console.WriteLine(text2);
   }
```

```
    public static String fileToText(String filePath)
    {
        StreamReader file = new StreamReader(filePath);
        String text = file.ReadToEnd();
        file.Close();
        return text;
    }

    public static void textToFile(String filePath, String text)
    {
        StreamWriter file = new StreamWriter(filePath);
        file.Write(text);
        file.Close();
    }
}
```

執行結果

```
C:\ccc>csc File.cs
Microsoft (R) Visual C# 2005 Compiler version 8.00.50727.3053
for Microsoft (R) Windows (R) 2005 Framework version 2.0.50727
```

```
Copyright (C) Microsoft Corporation 2001-2005. All rights reserved.



C:\ccc>File Hello.txt
Hello, C#!
```

物件導向的寫法

```
using System;
using System.IO;

class File
{
   String path;

   public static void Main(String[] args)
   {
      File file = new File("Hello.txt");
      file.save("Hello, C#!");
      String text2 = file.load();
      Console.WriteLine(text2);
```

```
}

public File(String path)
{
    this.path = path;
}

public String load()
{
    StreamReader file = new StreamReader(path);
    String text = file.ReadToEnd();
    file.Close();
    return text;
}

public void save(String text)
{
    StreamWriter file = new StreamWriter(path);
    file.Write(text);
    file.Close();
}
```

```
}
```

執行結果

```
C:\ccc>csc FileTest.cs
Microsoft (R) Visual C# 2005 Compiler version 8.00.50727.3053
for Microsoft (R) Windows (R) 2005 Framework version 2.0.50727
Copyright (C) Microsoft Corporation 2001-2005. All rights reserved.


C:\ccc>FileTest Hello.txt
Hello, C#!
```

更物件導向的寫法

```
using System;
using System.IO;

class TextFile
{
   String path;
```

```
String text;

public static void Main(String[] args)
{
    TextFile file = new TextFile();
    file.load(args[0]).print();
}

public TextFile() {}

public TextFile load(String path)
{
    this.path = path;
    StreamReader file = new StreamReader(path);
    text = file.ReadToEnd();
    file.Close();
    return this;
}

public TextFile save()
{
```

```
        StreamWriter file = new StreamWriter(path);
        file.Write(text);
        file.Close();
        return this;
    }

    public TextFile print()
    {
        Console.WriteLine("file: path="+path);
        Console.WriteLine(text);
        return this;
    }
}
```

執行結果

```
C:\ccc>csc TextFile.cs
Microsoft (R) Visual C# 2005 Compiler version 8.00.50727.3053
for Microsoft (R) Windows (R) 2005 Framework version 2.0.50727
Copyright (C) Microsoft Corporation 2001-2005. All rights reserved.
```

```
C:\ccc>TextFile Hello.txt
file: path=Hello.txt
Hello, C#!
C# 真有趣!
I love C#.
```

C# 程式進階：正規表達式

正規表達式是現代程式設計的重要工具，在 C# 當中，對正規表達式的支援相當的完整。您可以用正規表達式抽取出文件中的電話、地址、超連結、email 等欄位，因此正規表達式在文字型資料的處理上是相當方便的。

程式範例

在以下的範例中，我們利用正規表達式 "[0-9]+號"，抽取出字串當中的號碼，像是 32號，45號等。其中的 matches 函數是正規表達式的主要部分，我們透過 Regex 物件中的 Match(pText) 與 NextMatch() 函數，不斷取得比對的結果 (Match m)，然後再利用Match 結構取出 m.Groups[pGroupId].Value 這個比對的值，其中若 pGroudId 為 0 ，代表所要取得的是比對結果的全部。而 m.Success 可以用來判斷下一個比對是否成功，這可以做為回圈節數的條件。

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

public class Regexp
{
```

```
    public static void Main(String[] args)
    {
        String text = "王小明:32號，李小華：45號";
        foreach (String s in matches("[0-9]+號", text, 0))
         Console.WriteLine(s);
    }

    public static IEnumerable matches(String pPattern, String pText, int pGroupId)
    {
        Regex r = new Regex(pPattern, RegexOptions.IgnoreCase | RegexOptions.Compiled);
        for (Match m = r.Match(pText); m.Success; m = m.NextMatch())
         yield return m.Groups[pGroupId].Value;
    }
}
```

上述範例的執行結果如下所示，您可以看到字串 "王小明:32號，李小華：45號" 當中的 32號與 45 號被抽出來了，這正是正規表達式 "[0-9]+號" 所指定的樣式阿。

```
D:\myweb\teach\CSharpNetworkProgramming>csc RegexpTest1.cs
Microsoft (R) Visual C# 2008 Compiler version 3.5.30729.1
for Microsoft (R) .NET Framework version 3.5
```

```
Copyright (C) Microsoft Corporation. All rights reserved.



D:\myweb\teach\CSharpNetworkProgramming>RegexpTest1
32號
45號
```

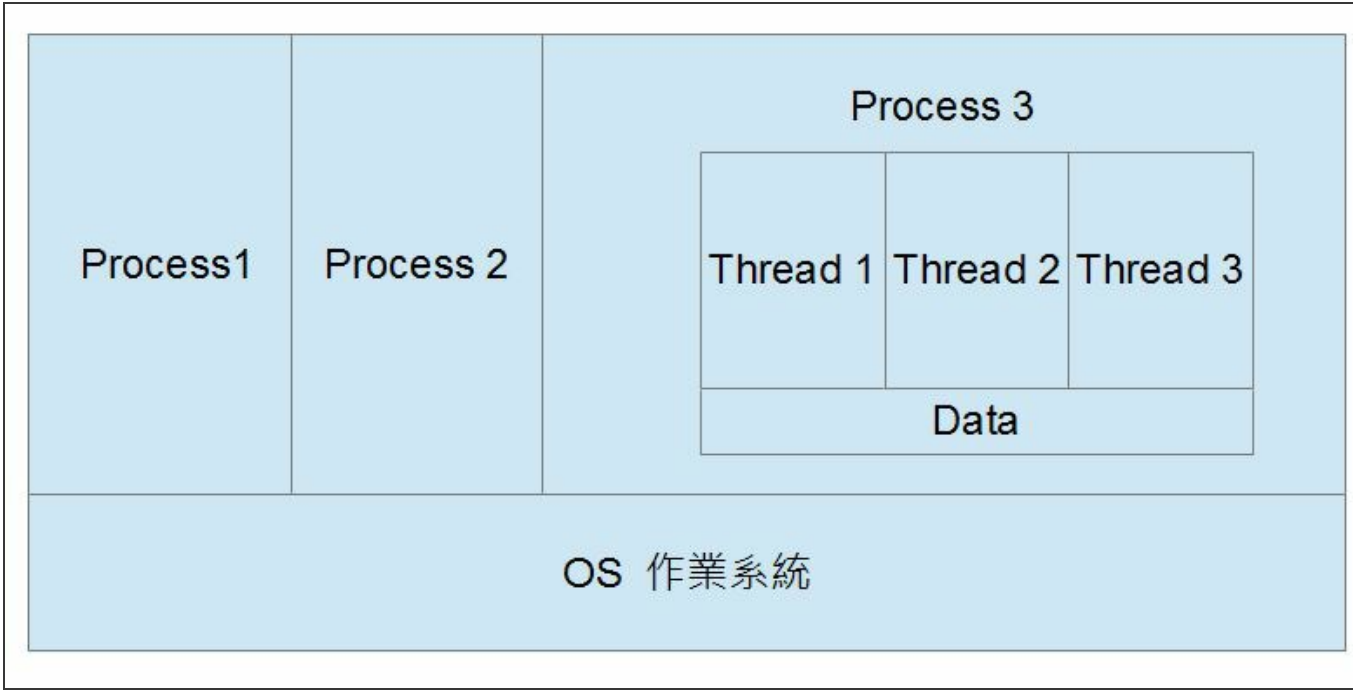
C# 程式進階：作業系統與 Thread

很多資工系的學生都上過作業系統這門課，但是通常老師只有講理論，沒有說明如何實作，這讓很多同學都無法清楚的理解其中的概念。在本系列文章中，我們將使用 C# 語言來說明作業系統當中的一些關鍵概念，像是 Process (進程、行程、Task)、Thread (執行緒、線程)、Deadlock (死結)、Race Condition (競爭情況)、Semaphore (號誌、信號量) 等等。

Process 與 Thread

Thread 在台灣被稱為『執行緒』，但是在中國被稱為『線程』，作業系統教科書中通常會定義 Process 為：執行中的程式。因此假如您開了一個 Word 檔案，那就是有一個 Word 行程在執行，如果您又開了個命令列，那就是又有一個命令列行程在執行，如果又開第二個命令列，那就有兩個命令列行程在執行。

Thread 在作業系統中通常被定義為輕量級行程 (Light Weight Process)，一個 Process 可以包含很多個 Thread，如下圖所示：

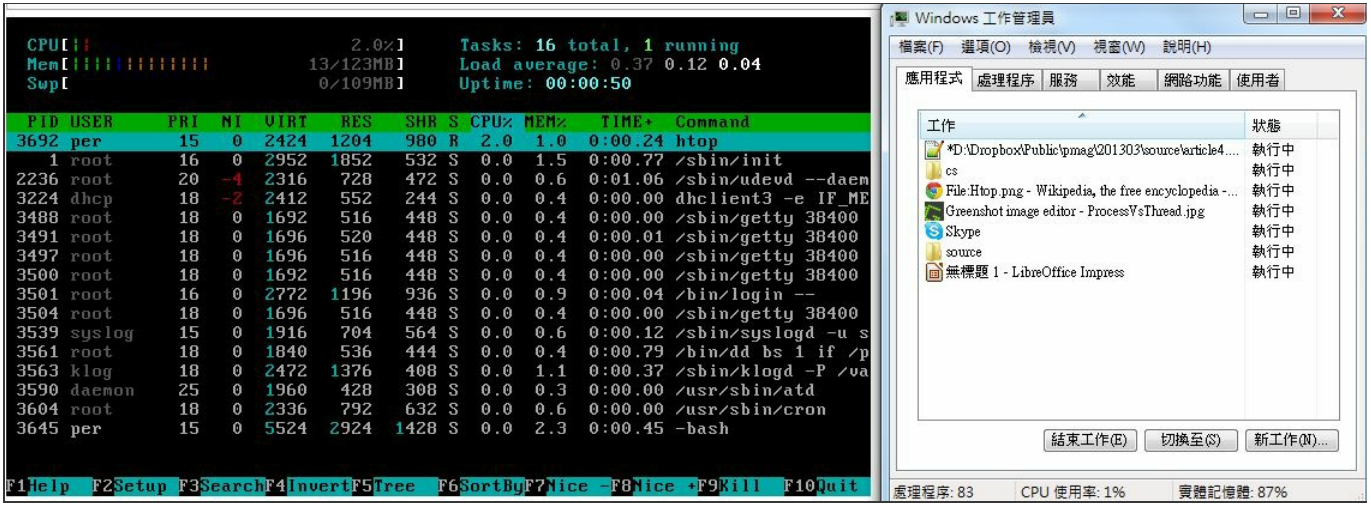


圖、Process 與 Thread 的關係

每個 Process 與 Thread 都會執行，而且執行到一半很可能就會因為進行輸出入或佔用 CPU 過久而被作業系統切換出去，改換另一個 Process 或 Thread 執行，這種概念稱為 Multitasking (多工)。

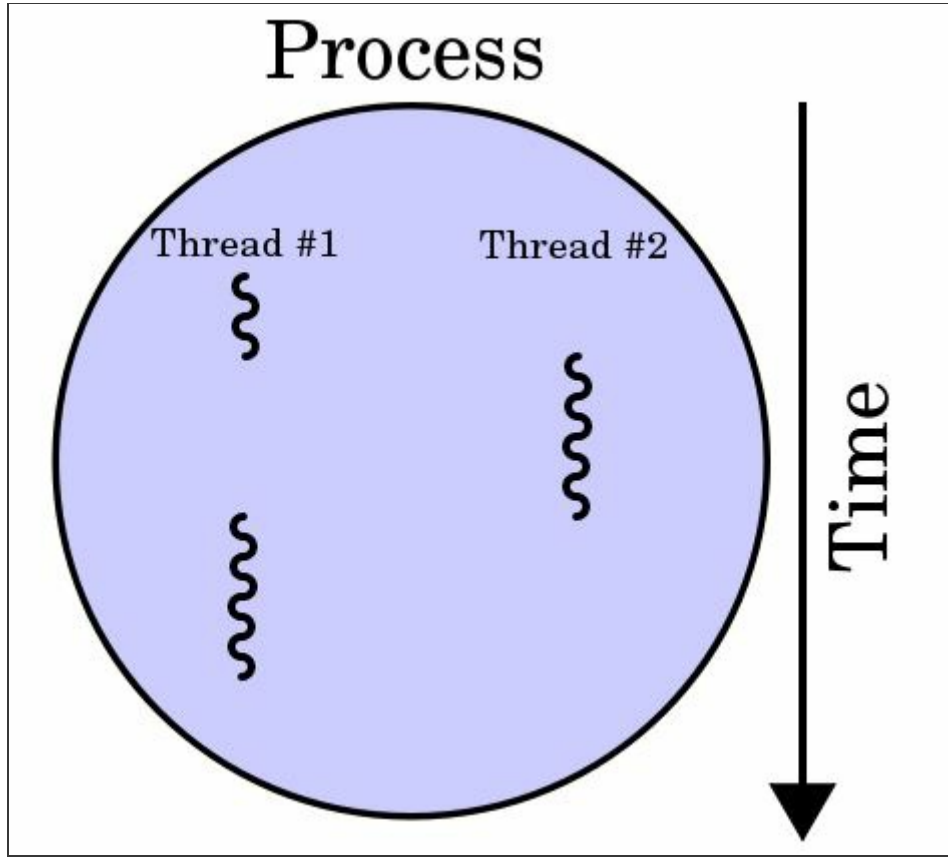
在 Windows 當中，我們可以按下 Ctrl-Alt-Del 鍵以顯示出系統的行程資訊，而在 Linux 中則可以用 ps

(Process Status) 這個指令顯示行程資訊，以下是這兩個作業系統中的行程資訊範例。



圖、Linux 與 Windows 中的 Process

Thread 交替執行的這種概念可以用下圖表示。(Proces 也是如此，只是將圖中的 Thread 改為 Process 而已)



圖、Thread 的概念

C# 中的 Thread 概念

在現代的作業系統當中，如果我們將一個程式重複執行兩次，將會產生兩個 Process ，那麼這兩個程式將是毫不相關的。任何一個程式都不需要知道另一個程式是否存在，通常也不會與另一個程式進行溝通。

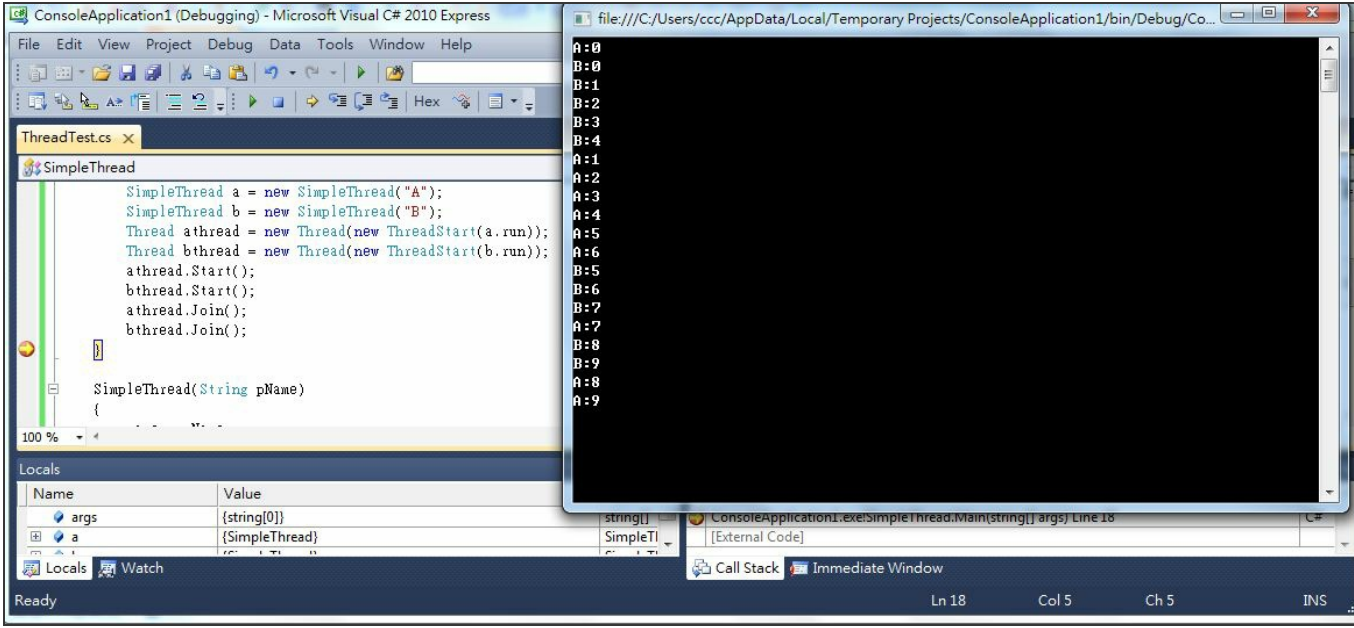
但是，如果我們希望兩個程式能夠互相分享某些變數，但是卻又同時執行，此時就可以利用 Thread 的機制。對於程式設計師而言，Thread 就像一個可以單獨執行的函數，這個函數與其他程式 (包含主程式) 同時執行，感覺上好像互相獨立，但是又可以共用某些變數。以下是一個 C# 的 Thread 範例：

```
using System;
using System.Threading;

class SimpleThread
{
   String name;

   public static void Main(String[] args)
   {
      SimpleThread a = new SimpleThread("A");
      SimpleThread b = new SimpleThread("B");
      Thread athread = new Thread(a.run);
```

```
    Thread bthread = new Thread(b.run);
    athread.Start();
    bthread.Start();
    athread.Join();
    bthread.Join();
}

SimpleThread(String pName)
{
    name = pName;
}

public void run()
{
    for (int i = 0; i < 10; i++)
    {
        String line = name + ":" + i;
        Console.WriteLine(line);
        // Thread.Sleep(10);
    }
}
```

```
}
```

其執行結果如下圖所示：



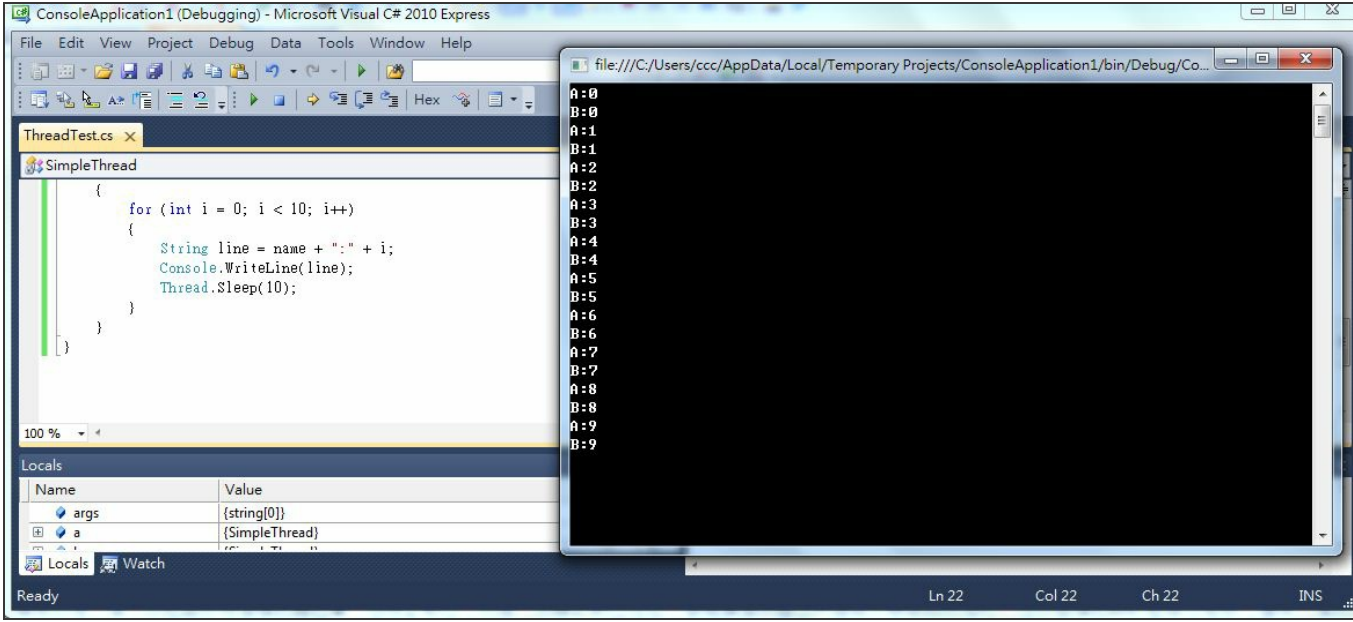
圖、Thread 的執行結果 -- 沒有 Sleep 的情況

對於剛開始接觸 Thread 的程式人員而言，會感覺到相當的詭異。因為『兩個 Thread 同時執行』是一個相當難以理解的概念。事實上，對於只有一個 CPU 的程式而言，並非兩個程式真的會「同時」執行，而只不過是「交錯」執行而已。但是這個交錯方式是由作業系統決定的，而非由程式設計師自行安排。而對於多 CPU 或多核心的處理器而言，就真的會「同時」執行，而不是只有「交錯」執行而已。

通常，程式人員對於這種不能由自己操控決定的情況會有不安的感覺，但是當您多寫幾個程式之後，這種

疑慮就會消除了，畢竟，程式人員本來就相當依賴作業系統，只是自己通常感覺不到而已。

當然，如果我們想要稍微控制一下 Thread 的執行順序，那麼就可以要求目前的 Thread 去休息睡覺，像是上述程式中的 Thread.Sleep(10) 這行程是本來是被註解掉的，但是如果我們將這行程式的註解拿掉，那麼將得到下列執行結果。

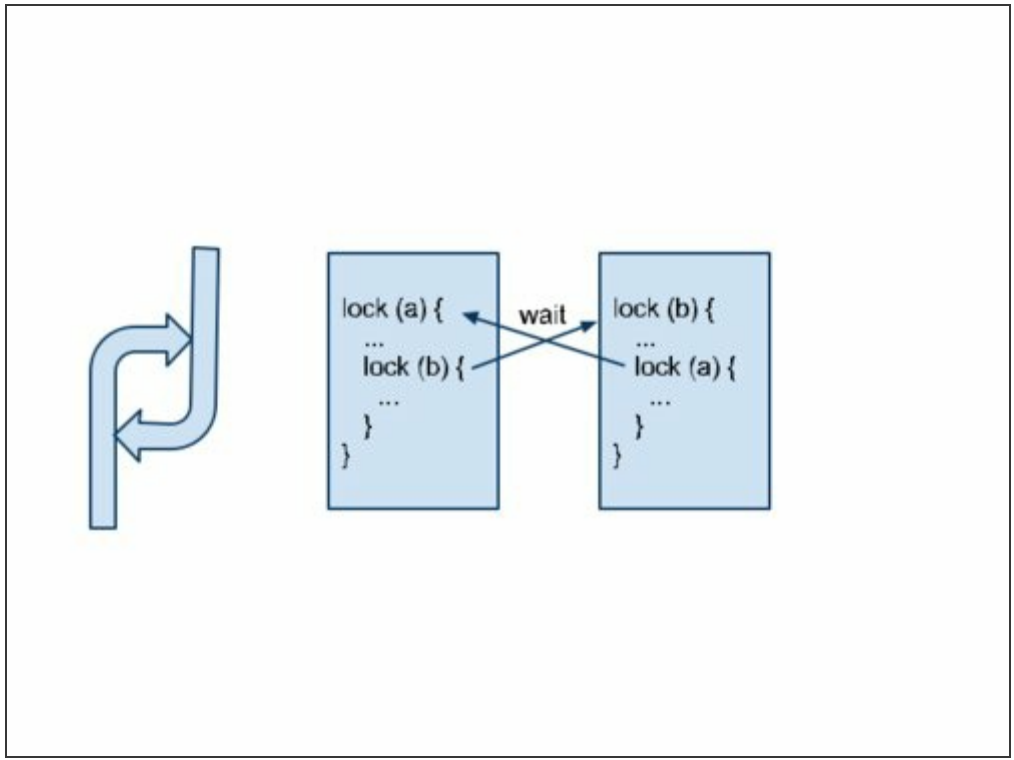


圖、Thread 的執行結果 2 -- 有 Sleep 的情況

從上面兩個圖中，您可以看到還沒加入 Thread.Sleep(10) 之前，兩個 Thread 的交錯方是很隨興，基本上是由作業系統任意安排的，但是在加入 Thread.Sleep(10) 之後，因為兩個 Thread 在印一次後就會禮讓給對方，所以就成了嚴格交互的 A, B, A, B .... 之情形了。

以 C# 體驗死結（Deadlock）

在作業系統的課程當中我們會學到『死結』這個問題，當程式 1 抓住資源 A，卻又在等程式 2 釋放資源 B，而程式 2 則抓住資源 B ，卻又在等程式 1 釋放資源 A 的時候，就會進入死結狀態。這就像兩台很長的火車，互相卡住對方一般，下圖顯示了死結情況的示意圖。



圖、死結的示意圖

在程式設計中我們真的會遇到死結嗎？如果真的有死結，能否寫一個會造成死結的程式呢？

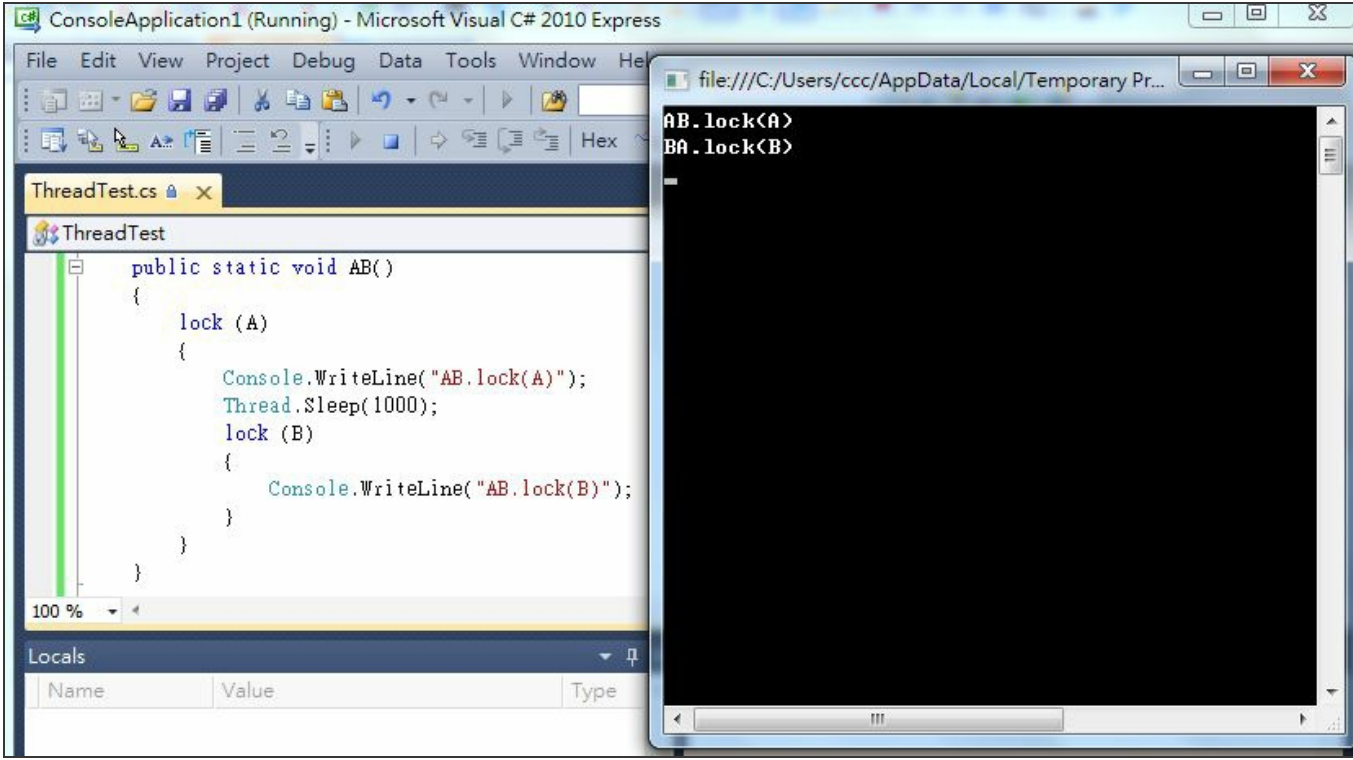
這並不難，只要用執行緒 (Thread) 與鎖定 (lock) 機制，我們很容易就可以造出會導致死結的程式，以下是我們用 C# 撰寫的一段死結程式，請參考。

```
using System;
using System.Threading;
using System.Text;

class ThreadTest
{
   static String A = "A";
   static String B = "B";

   public static void Main(String[] args)
   {
      Thread thread1 = new Thread(AB);
      Thread thread2 = new Thread(BA);
      thread1.Start();
      thread2.Start();
```

```
    thread1.Join();
    thread2.Join();
}

public static void AB()
{
    lock (A)
    {
        Console.WriteLine("AB.lock(A)");
        Thread.Sleep(1000);
        lock (B)
        {
            Console.WriteLine("AB.lock(B)");
        }
    }
}

public static void BA()
{
    lock (B)
    {
```

```
            Console.WriteLine("BA.lock(B)");
            Thread.Sleep(1000);
            lock (A)
            {
                Console.WriteLine("BA.lock(A)");
            }
        }
    }
}
```

上述程式的執行結果如下圖所示，當程式跑到 BA.lock(B) 之後就進入了死結，再也無法跑下去了，因此我們不會看到 BA.lock(A) 與 AB.lock(B) 這兩行輸出的結果，程式已經進入了死結狀態，再也出不來了。



圖、Deadlock.cs 程式的執行結果

競爭情況（Race Condition）

至此，我們已經用 C# 實作了作業系統中的 Thread 與 Deadlock 這兩種概念，但事實上、這兩個概念之間是有關係的，要理解 Thread 與死結之間的關係，就必須從 Race Condition (競爭情況) 這個問題談起。

在多 Thread (或多 CPU) 的情況之下，兩個 thread 可以共用某些變數，但是共用變數可能造成一個嚴重的問題，那就是當兩個 thread 同時修改一個變數時，這種修改會造成變數的值可能錯誤的情況，以下是一個範例。

Thread 1	Thread 2	Thread1+2 (第 1 種情況)	Thread1+2 (第 2 種情況)
counter = 0			
...			
R1 = counter	R1 = counter	T1:R1 = counter	T1:R1 = counter
R1 = R1 + 1	R1 = R1 - 1	T2:R1 = counter	T1:R1 = R1+1
counter = R1	counter = R1	T2:R1 = R1-1	T2:R1 = counter
		T2:counter=R1	T2:R1 = R1-1
		T1:R1 = R1+1	T2:counter = R1

		T1:counter = R1	T1:counter = R1
		...	...
		執行結果 : counter = 0	執行結果 : counter = -1

這種情況並非只在多 CPU 的系統裡會發生，在單 CPU 的多線程系統裡也會發生，因為一個高階語言指令在翻譯成機器碼時，通常會變成很多個指令，這讓修改的動作無法在單一指令內完成，如果這些修改動作執行到一半的時候，線程被切換了，就會造成上述的競爭情況。

高階語言	組合語言	對應動作
counter ++	LD R1, counter	R1 = counter
	ADD R1, R1, 1	R1 = R1 + 1
	ST R1, counter	counter = R1

這種競爭情況對程式設計者而言是無法接受的，如果程式的執行結果無法確保，那所有的程式都將陷入一團混亂，連 counter++ 這樣的動作都有問題的話，那任何多線程的程式都將無法正確運作。

為了避免這樣的問題產生，一個可能的解決方法是採用鎖定 (lock) 的方式，當我們執行共用變數的修改時，先進行鎖定，讓其他的線程無法同時修改該變數，等到修改完畢後解索後，其他的線程才能修改該變數，這樣就能避免掉競爭情況的問題了。

但是、一但我們能夠進行鎖定的動作，那就可能會造成上述的死結情況，這也正是 Thread 與死結之間的關係，讓我們用一句話總結如下：

> 因為多線程的程式會有競爭情況，為了避免該情況而引入了鎖定機制，
> 但是鎖定機制用得不好就會造成死結。

讓我們先用程式來驗證競爭情況的存在，以下是一個 C# 的競爭情況範例 (當然這種競爭情況是我們刻意造成的)。

檔案：RaceConditon.cs

```
using System;
using System.Threading;
using System.Collections.Generic;

class RaceCondition
{
   static int counter = 0, R1 = 0;
```

```
static void Main(string[] args)
{
    Thread thread1 = new Thread(inc);
    Thread thread2 = new Thread(dec);
    thread1.Start();
    thread2.Start();
    thread1.Join();
    thread2.Join();
}

static Random random = new Random();

static void waitAndSee(String msg)
{
    Thread.Sleep(random.Next(0, 10));
    Console.WriteLine(msg+"     R1:"+R1+" counter:"+counter);
}

static void inc()
{
    R1 = counter;
```

```
        waitAndSee("inc:R1 = counter");
        R1 = R1 + 1;
        waitAndSee("inc:R1 = R1 + 1 ");
        counter = R1;
        waitAndSee("inc:counter = R1");
    }

    static void dec()
    {
        R1 = counter;
        waitAndSee("dec:R1 = counter");
        R1 = R1 - 1;
        waitAndSee("dec:R1 = R1 - 1 ");
        counter = R1;
        waitAndSee("dec:counter = R1");
    }
}
```

執行結果

```
D:\Dropbox\Public\cs\code>RaceCondition
```

```
inc:R1 = counter     R1:0 counter:0
dec:R1 = counter     R1:0 counter:0
inc:R1 = R1 + 1      R1:0 counter:0
inc:counter = R1     R1:0 counter:0
dec:R1 = R1 - 1      R1:0 counter:0
dec:counter = R1     R1:0 counter:0

D:\Dropbox\Public\cs\code>RaceCondition
inc:R1 = counter     R1:0 counter:0
inc:R1 = R1 + 1      R1:0 counter:0
dec:R1 = counter     R1:0 counter:0
inc:counter = R1     R1:-1 counter:0
dec:R1 = R1 - 1      R1:-1 counter:0
dec:counter = R1     R1:-1 counter:-1
```

要解決以上的競爭情況，必須採用一些協調 (Cooperation) 方法，C# 當中所提供的主要協調方法是 lock 這個語句。簡單來說，C# 的 lock 的實作方式就是採用作業系統教科書中所說的 Monitor 之方法，只是在 lock 的開始加入 Monitor.Enter() 語句，然後在 lock 的結束加入 Monitor.Exit() 語句而已，其方法如下所示。

C# lock	Monitor 語句
lock (_locker) {	Monitor.Enter(_locker);

...	...
critical();	critical();
...	...
}	Monitor.Exit(_locker);

使用 lock 的方式，我們可以很輕易的解決上述程式的競爭情況，以下是該程式加入 lock 機制後的程式碼與執行結果。

檔案：RaceConditonLock.cs

```
using System;
using System.Threading;
using System.Collections.Generic;

class RaceConditionLock
{
   static int counter = 0, R1 = 0;
```

```
static String counterLocker = "counterLocker";

static void Main(string[] args)
{
    Thread thread1 = new Thread(inc);
    Thread thread2 = new Thread(dec);
    thread1.Start();
    thread2.Start();
    thread1.Join();
    thread2.Join();
}

static Random random = new Random();

static void waitAndSee(String msg)
{
    Thread.Sleep(random.Next(0, 10));
    Console.WriteLine(msg+"     R1:"+R1+" counter:"+counter);
}

static void inc()
```

```
{
    lock (counterLocker) {
        R1 = counter;
        waitAndSee("inc:R1 = counter");
        R1 = R1 + 1;
        waitAndSee("inc:R1 = R1 + 1 ");
        counter = R1;
        waitAndSee("inc:counter = R1");
    }
}

static void dec()
{
    lock (counterLocker) {
        R1 = counter;
        waitAndSee("dec:R1 = counter");
        R1 = R1 - 1;
        waitAndSee("dec:R1 = R1 - 1 ");
        counter = R1;
        waitAndSee("dec:counter = R1");
    }
```

```
    }
}
```

執行結果

```
D:\Dropbox\Public\cs\code>csc RaceConditionLock.cs
適用於 Microsoft (R) .NET Framework 4.5 的
Microsoft (R) Visual C# 編譯器版本 4.0.30319.17929
Copyright (C) Microsoft Corporation. 著作權所有，並保留一切權利。


D:\Dropbox\Public\cs\code>RaceConditionLock
inc:R1 = counter      R1:0 counter:0
inc:R1 = R1 + 1       R1:1 counter:0
inc:counter = R1      R1:1 counter:1
dec:R1 = counter      R1:1 counter:1
dec:R1 = R1 - 1       R1:0 counter:1
dec:counter = R1      R1:0 counter:0

D:\Dropbox\Public\cs\code>RaceConditionLock
inc:R1 = counter      R1:0 counter:0
```

```
inc:R1 = R1 + 1      R1:1 counter:0
inc:counter = R1     R1:1 counter:1
dec:R1 = counter     R1:1 counter:1
dec:R1 = R1 - 1      R1:0 counter:1
dec:counter = R1     R1:0 counter:0

D:\Dropbox\Public\cs\code>RaceConditionLock
inc:R1 = counter     R1:0 counter:0
inc:R1 = R1 + 1      R1:1 counter:0
inc:counter = R1     R1:1 counter:1
dec:R1 = counter     R1:1 counter:1
dec:R1 = R1 - 1      R1:0 counter:1
dec:counter = R1     R1:0 counter:0
```

號誌（Semaphore）與生產者/消費者問題

```
using System;
using System.Threading;
using System.Collections.Generic;

class ProducerConsumer
```

```
{
    static readonly int BUFFER_SIZE = 4;
    static Queue<int> circularBuffer = new Queue<int>(BUFFER_SIZE);
    static Semaphore semaFilledSlots = new Semaphore(0, BUFFER_SIZE);
    static Semaphore semaUnfilledSlots = new Semaphore(BUFFER_SIZE, BUFFER_SIZE);
    static Mutex mutex = new Mutex(false);

    static void Main(string[] args)
    {
        Thread producer = new Thread(new ThreadStart(produce));
        Thread consumer = new Thread(new ThreadStart(consume));
        producer.Start();
        consumer.Start();
        producer.Join();
        consumer.Join();
    }

    static Random random = new Random();

    private static void produce()
    {
```

```
    while (true)
    {
        Thread.Sleep(random.Next(0, 500));
        int produceNumber = random.Next(0, 20);
        Console.WriteLine("Produce: {0}", produceNumber);

        semaUnfilledSlots.WaitOne();    // wait(semaUnfilledSlots)
        mutex.WaitOne();                // wait(mutex)

        queue.Enqueue(produceNumber);

        mutex.ReleaseMutex();           // signal(mutex)
        semaFilledSlots.Release();      // signal(semaFilledSlots)

        if (produceNumber == 0)
            break;
    }
}

private static void consume()
{
```

```
        while (true)
        {
            semaFilledSlots.WaitOne();      // wait(semaFilledSlots)
            mutex.WaitOne();                // wait(mutex)
            int number = circularBuffer.Dequeue();
            Console.WriteLine("Consume: {0}", number);

            mutex.ReleaseMutex();           // signal(mutex)
            semaUnfilledSlots.Release();    // signal(semaUnfilledSlots)
            if (number == 0)
                break;
            Thread.Sleep(random.Next(0, 1000));
        }
    }
}
```

執行結果

```
D:\Dropbox\Public\cs\code>csc ProducerConsumer.cs
適用於 Microsoft (R) .NET Framework 4.5 的
Microsoft (R) Visual C# 編譯器版本 4.0.30319.17929
```

```
Copyright (C) Microsoft Corporation. 著作權所有，並保留一切權利。


D:\Dropbox\Public\cs\code>ProducerConsumer
Produce: 9
Consume: 9
Produce: 8
Consume: 8
Produce: 2
Consume: 2
Produce: 6
Produce: 0
Consume: 6
Consume: 0
```

另外、在作業系統中有個多行程的經典問題稱為 Dining Philospher (哲學家用餐) 問題，也可以採用 lock 的方法解決，由於這個問題實務上比較不那麼常用，本文中就不再詳細探討此一問題，有興趣的朋友可以看看網路上的解決方法，像是以下這篇 java2s 當中的程式就用 C# 實作解決了此一問題。

- http://www.java2s.com/Tutorial/CSharp/0420__Thread/DiningPhilosopher.htm

在本章中，我們購過透過實作的方式，讓讀者感受作業系統當中的 Thread、Deadlock、Race Condition、與

Semaphore 等概念，希望這樣的說明方式對讀者會有所幫助。

參考文獻

- Threading in C#, Joseph Albahari (超讚！)
    - http://www.albahari.com/threading/

C# 視窗程式：簡介

視窗與 C# 程式

本書所採用的視窗技術，是較舊型的 Windows Forms 技術，而非新型的 WPF 技術，雖然這種技術較為傳統，但是卻比較容易學習，對初學者而言是比較好上手的。

要學習 C# 的視窗程式設計，只看程式碼往往是不夠的，因為程式碼沒有辦法傳達有關視覺化介面設計過程的實況訊息。因此、最好搭配教學影片同時學習。筆者在金門大學上課的時候，幾乎都有用 CamStudio 將上課過程進行螢幕錄影，讀者可以一邊觀看這些上課錄影，一邊參考本書的程式碼進行學習，以下是這些錄影在 YouTube 上的網址。

主題	教學影片
C# 視窗程式設計簡介 1	http://youtu.be/MtCx-T71_bg
C# 視窗程式設計簡介 2	http://youtu.be/_9rEEWE3lXQ
C# 視窗控制項巡禮	http://youtu.be/e5366D_ngAA
C# 文字型計算機	http://youtu.be/A1KVnrfVF7k

C# 實作字典查詢程式	http://youtu.be/yS3G-H_hrFU
C# Array 與 List 物件的使用	http://youtu.be/EHSGtKpRprI
C# HashTable 與 Dictionary 物件的使用	http://youtu.be/hYH-npRmmKM
C# 將小字典擴充為小翻譯系統	http://youtu.be/_sFsTo41PXs
C# Timer 與碼錶	http://youtu.be/UA0rizekLow
C# Timer 與小時鐘	http://youtu.be/NJ6B5-vO_88
C# Timer 與移動球	http://youtu.be/6Gs4MPzt6q4
C# 檔案處理	http://youtu.be/3EyPcAddd70
C# 文字編輯器 1	http://youtu.be/xymT54El53E
C# 文字編輯器 2	http://youtu.be/xz5sKvZjLZI
C# 畫圖功能示範	http://youtu.be/8aAql8R4WSg

C# 小畫板 (2)	http://youtu.be/HkOkWRQ_Ad4
C# 小畫板 (3)	http://youtu.be/VXaQu_yYu08
C# 瀏覽器控制 1/3	http://youtu.be/CIwYabPN7qA
C# 瀏覽器控制 2/3	http://youtu.be/sJ6cfuL3-ZA
C# 瀏覽器控制 3/3	http://youtu.be/YThlDxk-E7U
C# DataGridView 元件的使用	http://youtu.be/XzsZRqLAi30
C# 製作賣紅茶的 POS 系統	http://youtu.be/XX0wHhvkkiE

讀者可以先看「C# 視窗程式設計簡介1, 2 與 C# 視窗控制項巡禮」等三部影片，並且操作一次，有了基本概念之後，就可以繼續下看下列的內容了。

按鈕測試

專案下載：https://dl.dropbox.com/u/101584453/cs/code/ButtonTest.zip

```
using System;
```

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace ButtonTest
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            label1.Text = "你好！";
            textBox1.Text = "我很好！";
```

```
    }
  }
}
```

文字型計算機

教學錄影：C# 文字型計算機 -- http://youtu.be/A1KVnrfVF7k 專案下載：https://dl.dropbox.com/u/101584453/cs/code/TextCalculator.zip

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace TextCalculator
{
  public partial class Form1 : Form
```

```
{
    public Form1()
    {
        InitializeComponent();
    }

    private void buttonEqual_Click(object sender, EventArgs e)
    {
        try
        {
            double number1 = Double.Parse(textBoxNumber1.Text);
            double number2 = Double.Parse(textBoxNumber2.Text);
            string op = comboBoxOp.Text;
            double result = 0.0;
            // MessageBox.Show(number1 + op + number2 + "=");
            switch (op)
            {
                case "+": result = number1 + number2; break;
                case "-": result = number1 - number2; break;
                case "*": result = number1 * number2; break;
                case "/": result = number1 / number2; break;
```

```
                default: throw new Exception("op=" + op + ":Error!");
            }
            textBoxResult.Text = result.ToString();
        }
        catch (Exception exception)
        {
            MessageBox.Show(exception.ToString());
        }
    }
  }
}
```

小字典

教學錄影：

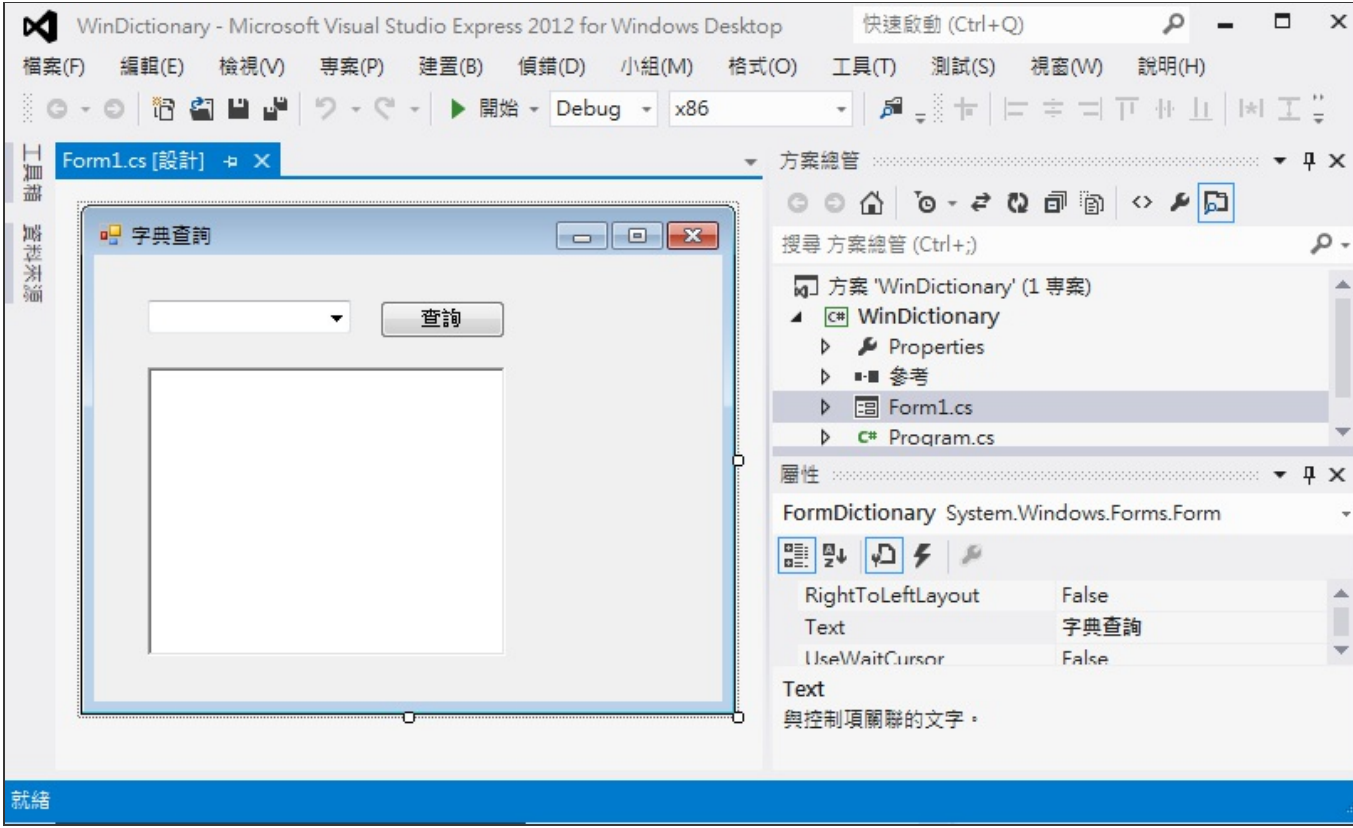
- C# 實作字典查詢程式 -- http://youtu.be/yS3G-H_hrFU
- C# Array 與 List 物件的使用 -- http://youtu.be/EHSGtKpRprI
- C# HashTable 與 Dictionary 物件的使用 -- http://youtu.be/hYH-npRmmKM
- C# 將小字典擴充為小翻譯系統 -- http://youtu.be/_sFsTo41PXs

專案下載： https://dl.dropbox.com/u/101584453/cs/code/WinDictionary.zip

執行畫面：

小字典執行畫面

介面設計


WinDictionary - Microsoft Visual Studio Express 2012 for Windows Desktop
快速啟動 (Ctrl+Q)
檔案(F) 編輯(E) 檢視(V) 專案(P) 建置(B) 偵錯(D) 小組(M) 格式(O) 工具(T) 測試(S) 視窗(W) 說明(H)
開始 Debug x86
Form1.cs [設計]
字典查詢
查詢
方案總管
搜尋 方案總管 (Ctrl+;)
方案 'WinDictionary' (1 專案)
WinDictionary
Properties
參考
Form1.cs
Program.cs
屬性
FormDictionary System.Windows.Forms.Form
RightToLeftLayout False
Text 字典查詢
UseWaitCursor False
Text
與控制項關聯的文字。
就緒

小字典介面設計

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class FormDictionary : Form
    {
        Dictionary<String, String> dict = new Dictionary<string, string>();

        public FormDictionary()
        {
            InitializeComponent();
        }
```

```
private void buttonQuery_Click(object sender, EventArgs e)
{
    try
    {
        String eword = comboBoxQuery.Text;
        String cword = dict[eword];
        richTextBox.Text = cword;
    }
    catch
    {
        MessageBox.Show("輸入錯誤，查不到！");
    }
}

private void FormDictionary_Load(object sender, EventArgs e)
{
    String[] eWords = new String[] { "dog", "cat", "eat", "chase", "run", "a",    "the" };
    String[] cWords = new String[] { "狗", "貓", "吃", "追",    "跑", "一隻", "這隻" };
    for (int i = 0; i < eWords.Length; i++)
    {
        dict.Add(eWords[i], cWords[i]);
```

```
            }
        }
    }
}
```

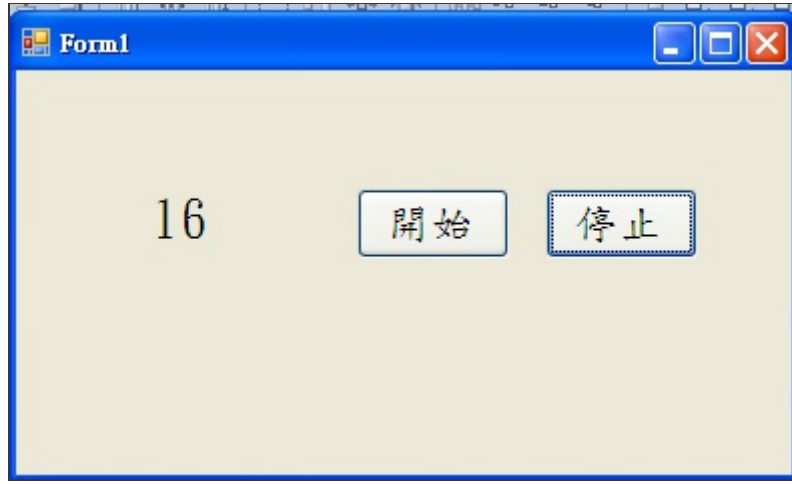
C# 視窗程式：時間驅動

碼表

教學錄影：C# Timer 與碼錶 -- http://youtu.be/UA0rizekLow

專案下載：

- https://dl.dropbox.com/u/101584453/cs/code/TimerDemo.zip
- https://dl.dropbox.com/u/101584453/cs/code/TimerClock.zip



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        int counter = 0;

        public Form1()
        {
            InitializeComponent();
```

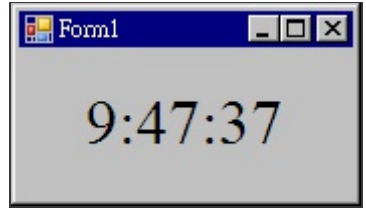
```
        }

        private void timer_Tick(object sender, EventArgs e)
        {
            counter++;
            labelTimer.Text = ""+counter;
        }

        private void buttonStart_Click(object sender, EventArgs e)
        {
            timer.Enabled = true;
        }

        private void buttonStop_Click(object sender, EventArgs e)
        {
            timer.Enabled = false;
        }
    }
}
```

小時鐘

教學錄影：C# Timer 與小時鐘 -- http://youtu.be/NJ6B5-vO_88

專案下載：https://dl.dropbox.com/u/101584453/cs/code/Clock.zip

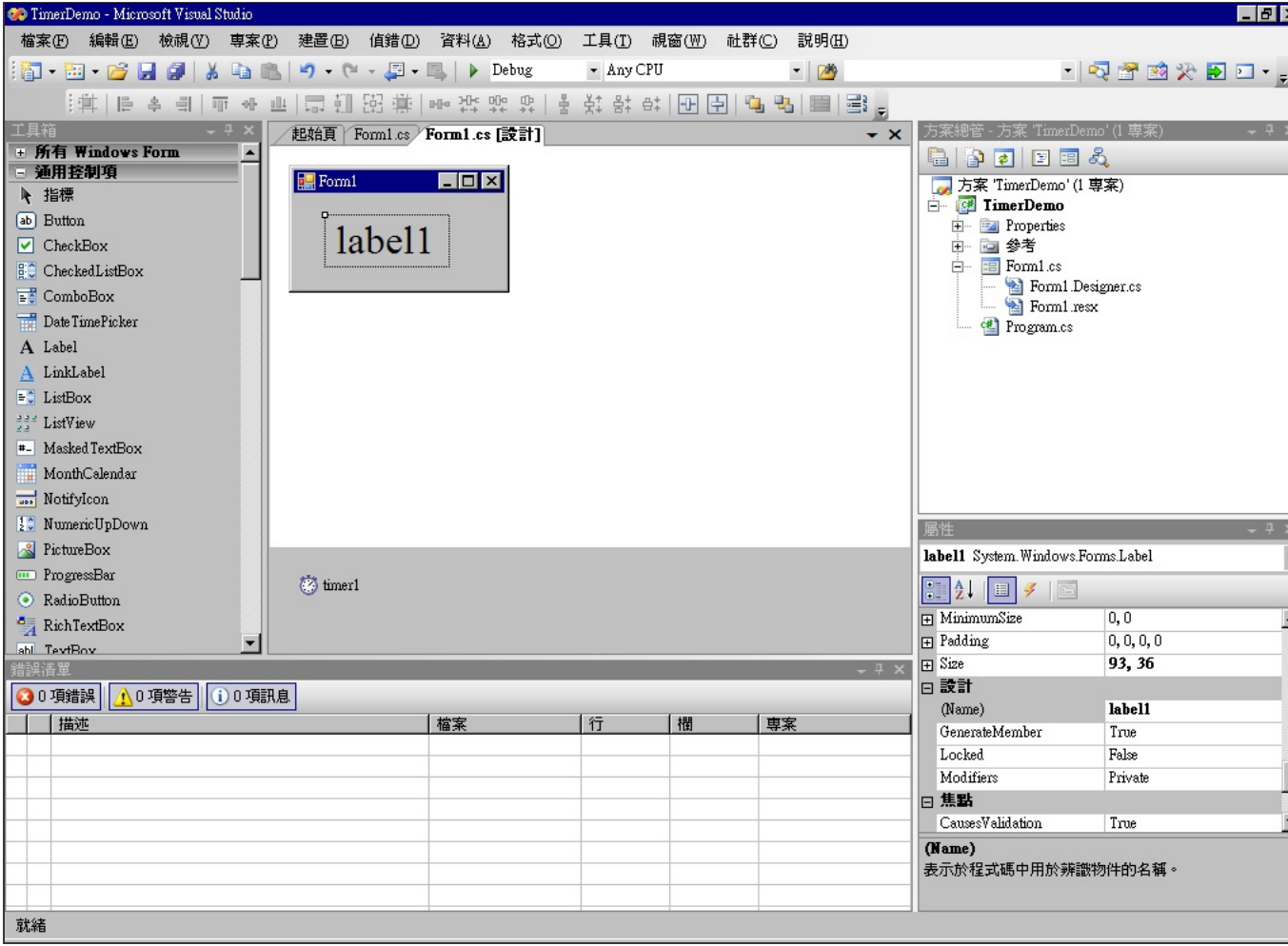
執行畫面：



小時鐘

介面設計

請從工具箱中拉出一個 label 與一個 Timer，如下圖所示。


TimerDemo - Microsoft Visual Studio
檔案(F) 編輯(E) 檢視(V) 專案(P) 建置(B) 偵錯(D) 資料(A) 格式(O) 工具(T) 視窗(W) 社群(C) 說明(H)
Debug
Any CPU
工具箱
所有 Windows Form
通用控制項
指標
Button
CheckBox
CheckedListBox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
NotifyIcon
NumericUpDown
PictureBox
ProgressBar
RadioButton
RichTextBox
TextBox
起始頁
Form1.cs
Form1.cs [設計]
Form1
label1
timer1
方案總管 - 方案 'TimerDemo' (1 專案)
方案 'TimerDemo' (1 專案)
TimerDemo
Properties
參考
Form1.cs
Form1.Designer.cs
Form1.resx
Program.cs
屬性
label1 System.Windows.Forms.Label
MinimumSize 0, 0
Padding 0, 0, 0, 0
Size 93, 36
設計
(Name) label1
GenerateMember True
Locked False
Modifiers Private
焦點
CausesValidation True
(Name)
表示於程式碼中用於辨識物件的名稱。
錯誤清單
0 項錯誤
0 項警告
0 項訊息
描述
檔案
行
欄
專案
就緒

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace TimerDemo
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            timer1_Tick(this, null);
            timer1.Interval = 1000; // 設定每秒觸發一次
            timer1.Enabled = true; // 啟動 Timer
```

```
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            DateTime time = DateTime.Now;
//          label1.Text = time.TimeOfDay.ToString();
//          label1.Text = time.Hour + ":" + time.Minute + ":" + time.Second;
            label1.Text = String.Format("{0:00}:{1:00}:{2:00}",
                                        time.Hour, time.Minute, time.Second);
        }
    }
}
```

用 Timer 控制動畫

教學錄影：C# Timer 與移動球 -- http://youtu.be/6Gs4MPzt6q4

C# 視窗程式：設計文字編輯器

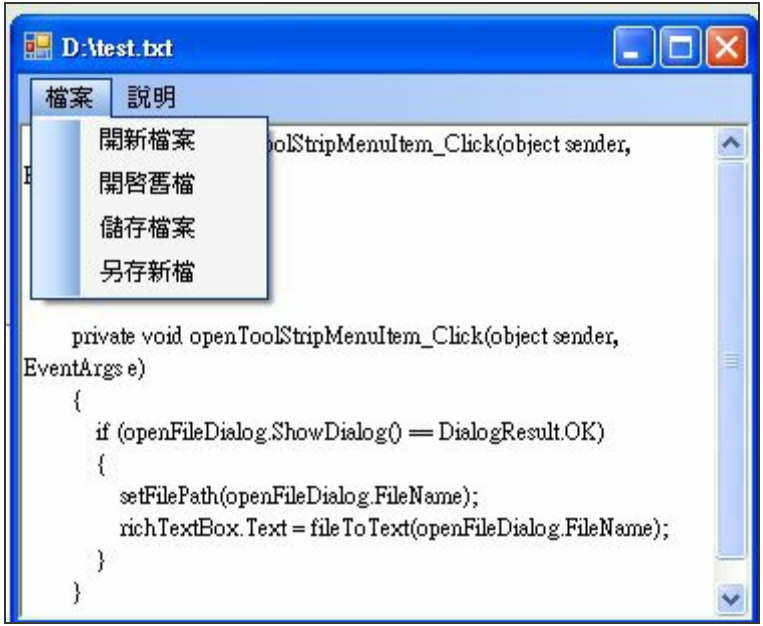
教學錄影：

- C# 檔案處理 -- http://youtu.be/3EyPcAddd70
- C# 文字編輯器 1 -- http://youtu.be/xymT54El53E
- C# 文字編輯器 2 -- http://youtu.be/xz5sKvZjLZI

專案下載：

- 只有介面的版本 (英文) -- https://dl.dropbox.com/u/101584453/cs/code/Editor1.zip
- 只有介面的版本 (中文) -- https://dl.dropbox.com/u/101584453/cs/code/Editor2.zip
- 完整版 -- https://dl.dropbox.com/u/101584453/cs/code/TextEditor2012.zip

執行結果



文字編輯器執行畫面

程式碼

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

```
using System.IO;
namespace WindowsFormsApplication1
{
    public partial class FormEditor : Form
    {
        String filePath = null;

        public FormEditor()
        {
            InitializeComponent();
        }

        public static String fileToText(String filePath)
        {
            StreamReader file = new StreamReader(filePath);
            String text = file.ReadToEnd();
            file.Close();
            return text;
        }

        public static void textToFile(String filePath, String text)
```

```
{
    StreamWriter file = new StreamWriter(filePath);
    file.Write(text);
    file.Close();
}

private void openFileToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (openFileDialog.ShowDialog() == DialogResult.OK)
    {
        String text = fileToText(openFileDialog.FileName);
        richTextBox.Text = text;
        filePath = openFileDialog.FileName;
    }
}

private void newFileToolStripMenuItem_Click(object sender, EventArgs e)
{
    richTextBox.Text = "";
    filePath = null;
}
```

```
private void saveFileToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (filePath == null)
    {
        dialogSaveFile();
    }
    else
    {
        textToFile(filePath, richTextBox.Text);
    }
}

private void saveAsToolStripMenuItem_Click(object sender, EventArgs e)
{
    dialogSaveFile();
}

public void dialogSaveFile()
{
    if (saveFileDialog.ShowDialog() == DialogResult.OK)
```

```
            {
                textToFile(saveFileDialog.FileName, richTextBox.Text);
                filePath = saveFileDialog.FileName;
            }
        }


    }
}
```

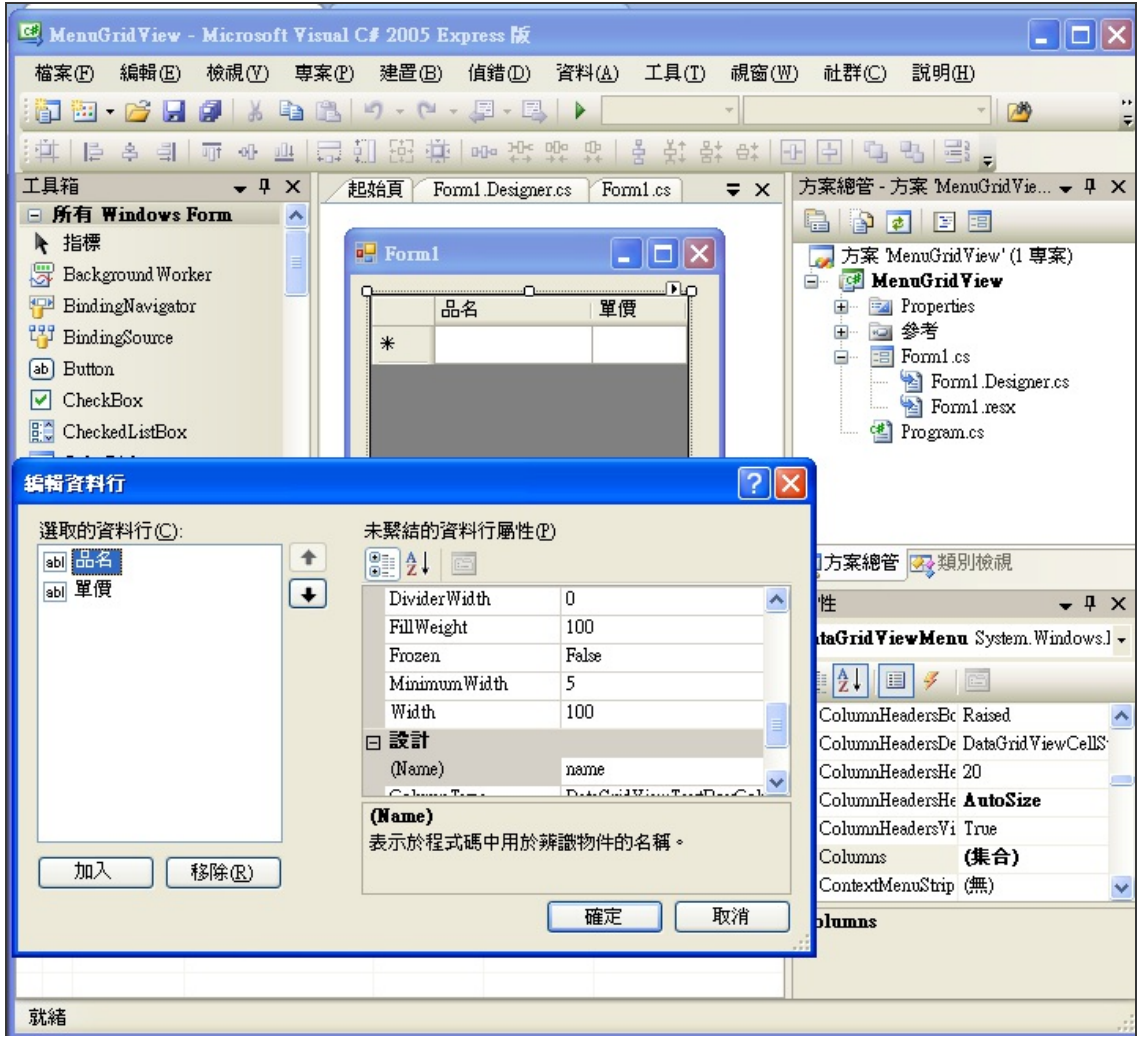
C# 視窗程式：DataGridView 與 ListView 元件

DataGridView 的使用

在微軟 .NET 的架構中，DataGridView 是一個強大而複雜的元件，我們欣賞其功能的強大，但也相當畏懼其複雜性，有時我們並不想將這個元件與資料庫綁在一起使用，但是卻不知道應該如何直接在元件中填入表格資料，以下是這個問題的一個簡單解法。

專案下載：https://dl.dropbox.com/u/101584453/cs/code/MenuGridView.zip

設計：當您用 Visual Studio 拉出一個 DataGridView 物件 (假設稱為 dataGridViewMenu) 後，可以點選其屬性中的 Columns 欄位，然後用視覺化的方式加入兩個 Column，並設定好名稱為 name (品名), price (單價)，以下是筆者設定時的畫面。


MenuGridView - Microsoft Visual C# 2005 Express 版
檔案(F) 編輯(E) 檢視(V) 專案(P) 建置(B) 偵錯(D) 資料(A) 工具(T) 視窗(W) 社群(C) 說明(H)
工具箱
所有 Windows Form
指標
BackgroundWorker
BindingNavigator
BindingSource
Button
CheckBox
CheckedListBox
起始頁
Form1.Designer.cs
Form1.cs
Form1
品名
單價
方案總管 - 方案 'MenuGridVie...
方案 'MenuGridView' (1 專案)
MenuGridView
Properties
參考
Form1.cs
Form1.Designer.cs
Form1.resx
Program.cs
編輯資料行
選取的資料行(C):
品名
單價
加入
移除(R)
未繫結的資料行屬性(P)
DividerWidth 0
FillWeight 100
Frozen False
MinimumWidth 5
Width 100
設計
(Name) name
(Name)
表示於程式碼中用於辨識物件的名稱。
確定
取消
方案總管
類別檢視
ColumnHeadersBc Raised
ColumnHeadersDe DataGridViewCellS
ColumnHeadersHe 20
ColumnHeadersHe AutoSize
ColumnHeadersVi True
Columns (集合)
ContextMenuStrip (無)
就緒

授課錄影：C# DataGridView 元件的使用 -- http://youtu.be/XzsZRqLAi30

賣泡沫紅茶的系統（Point of Sale, POS）

授課錄影：

- C# 製作賣紅茶的 POS 系統 -- http://youtu.be/XX0wHhvkkiE
- 設計飲料店點菜系統 (1/2: 介面設計) -- http://youtu.be/4NRzpSNLQ44
- 設計飲料店點菜系統 (2/2: 程式邏輯) -- http://youtu.be/KO6fENU5Yfo

為了展示如何用 C# 設計一個小型的銷售點系統，我們設計了一個販賣飲料的 POS 軟體，可以讓販售者於電腦上點選飲料與數量，然後系統會自動計算總金額，這個程式可以用 ListView 或 DataGridView 元件進行設計，在上述的專案下載連結中，我們分別設計了這些專案，以下說明是我們使用 DataGridView 的設計方式的結果。

專案下載：

- 簡易版 -- https://dl.dropbox.com/u/101584453/cs/code/Menu1.zip (只有一樣商品：紅茶)
- 完整版 -- https://dl.dropbox.com/u/101584453/cs/code/Menu2.zip (使用 ListView)
- 豪華版 -- https://dl.dropbox.com/u/101584453/cs/code/POS2012.zip (2012 年更新版，使用 DataGridView)

賣泡沫紅茶的系統的畫面設計

程式碼

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace POS
{
    public partial class FormMenu : Form
    {
        String name; // 品名
        double price; // 單價
        double number; // 數量
        double subTotal; // 小計

        public FormMenu()
        {
```

```
        InitializeComponent();

        // 填入銷售項目到菜單中
        DataGridViewRowCollection rows = dataGridViewMenu.Rows;
        rows.Add(new Object[] { "紅茶", 25 });
        rows.Add(new Object[] { "綠茶", 25 });
        rows.Add(new Object[] { "奶茶", 30 });
        rows.Add(new Object[] { "珍珠奶茶", 35 });
    }

    private void dataGridViewMenu_CellClick(object sender, DataGridViewCellEventArgs e)
    {
        // 選取某項目，將該項目的品名價格放入待購項目中。
        if (dataGridViewMenu.Rows[e.RowIndex].Cells[0].Value == null)
            return;
        buttonName.Text = dataGridViewMenu.Rows[e.RowIndex].Cells[0].Value.ToString();
        textBoxPrice.Text = dataGridViewMenu.Rows[e.RowIndex].Cells[1].Value.ToString();
    }

    private void buttonAdd_Click(object sender, EventArgs e)
    {
```

```
    //加入紐被按下
    calculateSubTotal(); //計算小計，並加入到訂單中
    dataGridViewOrder.Rows.Add(new Object[] { name, price, number, subTotal });
    calculateTotal(); // 重新計算總價
}

private void calculateSubTotal() //計算小計
{
    name = buttonName.Text;
    price = double.Parse(textBoxPrice.Text);
    number = (double)numericUpDownNumber.Value;
    subTotal = price * number; // 小計 = 價格 * 數量
    textBoxSubtotal.Text = subTotal.ToString();
}

private void calculateTotal() //計算總價
{
    double total = 0.0;
    for (int i = 0; i < dataGridViewOrder.Rows.Count; i++)
    {
        DataGridViewRow row = dataGridViewOrder.Rows[i];
```

```
            if (row.Cells[0].Value != null)
                total += (double)row.Cells[3].Value;
        }
        textBoxTotal.Text = total.ToString();
    }

    private void numericUpDownNumber_ValueChanged(object sender, EventArgs e)
    {
        calculateSubTotal(); //數量修改時，重新計算小計。
    }

    private void buttonTotal_Click(object sender, EventArgs e)
    {
        calculateTotal(); //按下總價鈕時，重新計算總價。
    }
  }
}
```

執行結果

賣泡沫紅茶的系統的執行結果

參考文獻

- Data Binding in .NET / C# Windows Forms -- http://www.akadia.com/services/dotnet_databinding.html
- C# DataRow Examples -- http://dotnetperls.com/datarow
- How to bind a DataGridView to a DataSet without a database? -- http://www.devnewsgroups.net/windowsforms/t57859-how-bind-datagridview-dataset-without-database.aspx
- Data Binding in DataGrid Control - Part 1 -- http://www.c-sharpcorner.com/UploadFile/mahesh/DataBindingInDataGridMCB112022005002207AM/DataBindingInDataGr

C# 視窗程式：繪圖功能

畫圖功能示範

教學錄影：C# 畫圖功能示範 -- http://youtu.be/8aAql8R4WSg

專案下載：https://dl.dropbox.com/u/101584453/cs/code/PaintDemo.zip

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
```

```
{
    Graphics g;
    Pen pen;
    Font font;
    Brush brush;

    public Form1()
    {
        InitializeComponent();

        g = this.CreateGraphics();
        pen = new Pen(Color.Black, 3);
        font = new Font("標楷體", 16);
        brush = new SolidBrush(Color.Black);
    }

    private void Form1_Paint(object sender, PaintEventArgs e)
    {
        g.DrawLine(pen, new Point(1, 1), new Point(300, 100));
        g.DrawLine(pen, new Point(100, 1), new Point(300, 100));
        g.DrawRectangle(pen, new Rectangle(50, 50, 100, 100));
```

```
            g.DrawString("Hello! 你好！", font, brush, new PointF(150.0F, 150.0F));

            Image image = Image.FromFile("../../ccc.jpg");
            g.DrawImage(image, new Point(200, 200));
        }
    }
}
```

小畫板

教學錄影： * C# 小畫板 (2) -- http://youtu.be/HkOkWRQ_Ad4 * C# 小畫板 (3) -- http://youtu.be/VXaQu_yYu08

專案下載：https://dl.dropbox.com/u/101584453/cs/code/Painter.zip

小畫板執行畫面

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
```

```
namespace Painter
{
    public partial class Form1 : Form
    {
        Graphics g;                 //繪圖區
        Pen pen;                    //畫筆
        bool isMouseDown = false;   //紀錄滑鼠是否被按下
        List<Point> points = new List<Point>(); //紀錄滑鼠軌跡的陣列。

        public Form1()
        {
            InitializeComponent();

            g = this.CreateGraphics(); //取得繪圖區物件
            pen = new Pen(Color.Black, 3); //設定畫筆為黑色、粗細為 3 點。
        }

        private void Form1_MouseDown(object sender, MouseEventArgs e)
        {
            isMouseDown = true; //滑鼠被按下後設定旗標值。
            points.Add(e.Location); //將點加入到 points 陣列當中。
```

```
        }

        private void Form1_MouseMove(object sender, MouseEventArgs e)
        {
            if (isMouseDown) // 如果滑鼠被按下
            {
                points.Add(e.Location); // 將點加入到 points 陣列當中。
                // 畫出上一點到此點的線段。
                g.DrawLine(pen, points[points.Count - 2], points[points.Count - 1]);
            }
        }

        private void Form1_MouseUp(object sender, MouseEventArgs e)
        {
            points.Add(new Point(-1, -1)); // 滑鼠放開時，插入一個斷點 (-1,-1)，以代表前後兩點之間有斷開。
            isMouseDown = false; // 滑鼠已經沒有被按下了。
        }
    }
}
```

C# 網路程式：簡介、IP 與 URL

簡介

最常被使用的網路函式庫稱為 Socket，這個名詞起源於柏克萊大學於 1983 年所釋放出來的 Berkeley Sockets 函式庫，該函式庫將網路視為串流。因而使存取網路的動作，與存取檔案一樣，都可以透過串流機制運行。

雖然 Socket 函式庫將網路抽像化為串流，但是理解網路的架構對程式的學習仍有很大的幫助，目前我們所使用的 Internet 網路是基於 TCP/IP 的網路架構，要能理解目前的網路程式架構，首先要從 TCP/IP 的架構下手。

TCP/IP 網路架構

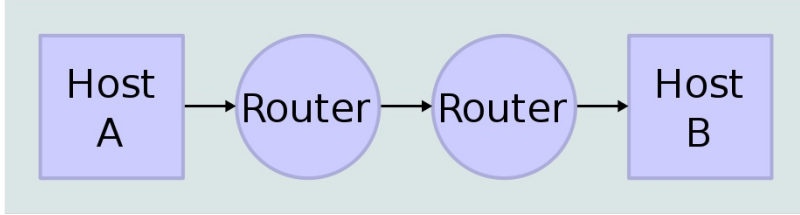
網路的 OSI 參考模型將網路的層次分為七層，但是 TCP/IP 架構所依據的 ARPANET DoD 模型則只有四層，兩者之間存在某種對應關係，這個對應關係顯示於下圖當中，能正確的理解該圖，將有助於程式設計師理解網路程式的原理。

層次	OSI	TCP/IP	協定範例
7	應用層	HTTP	HTTP、SMTP、SNMP、FTP、Telnet、

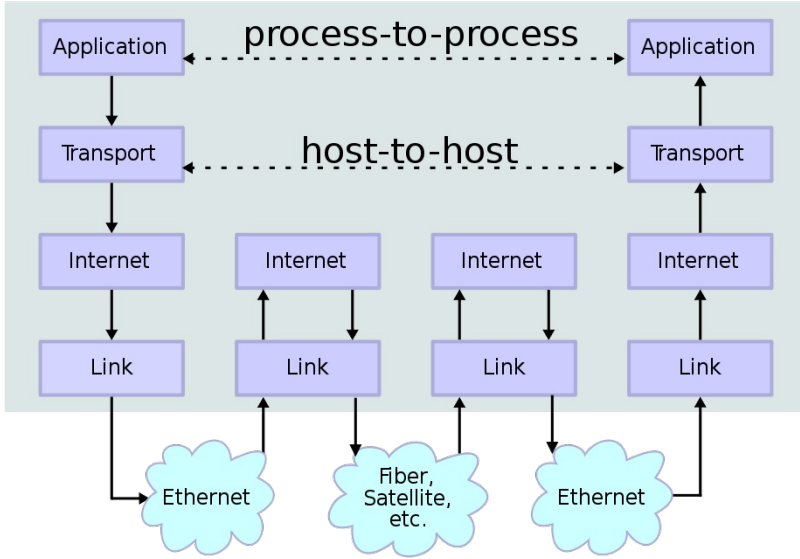
			SIP、SSH、NFS、RTSP、XMPP、Whois、ENRP
6	表示層		XDR、ASN.1、SMB、AFP、NCP
5	會話層		ASAP、SSH、ISO 8327 / CCITT X.225、 RPC、NetBIOS、ASP、Winsock、BSD sockets
4	傳輸層	TCP/UDP	TCP、UDP、TLS、RTP、SCTP、SPX、ATP、IL
3	網路層	IP	IP、ICMP、IGMP、IPX、BGP、OSPF、RIP、 IGRP、EIGRP、ARP、RARP、 X.25
2	鏈結層		乙太網、令牌環、HDLC、幀中繼、ISDN、 ATM、IEEE 802.11、FDDI、PPP
1	實體層		線路、無線電、光纖

舉例而言，當我們使用 TCP 的方式傳輸訊息時，由於 TCP 傳輸層會自動維持封包排列的順序，因此先傳送的封包一定會先到達，這讓程式設計師不需要擔心封包的先後順序問題。但在使用 UDP 傳輸的時候，先傳送的封包可能反而會後到達，因此訊息的順序將無法確保，這種 UDP 傳送方式雖然較快，但是卻較不方便，通常只被使用在強調速度的領域，像是立即影音傳輸的應用上。

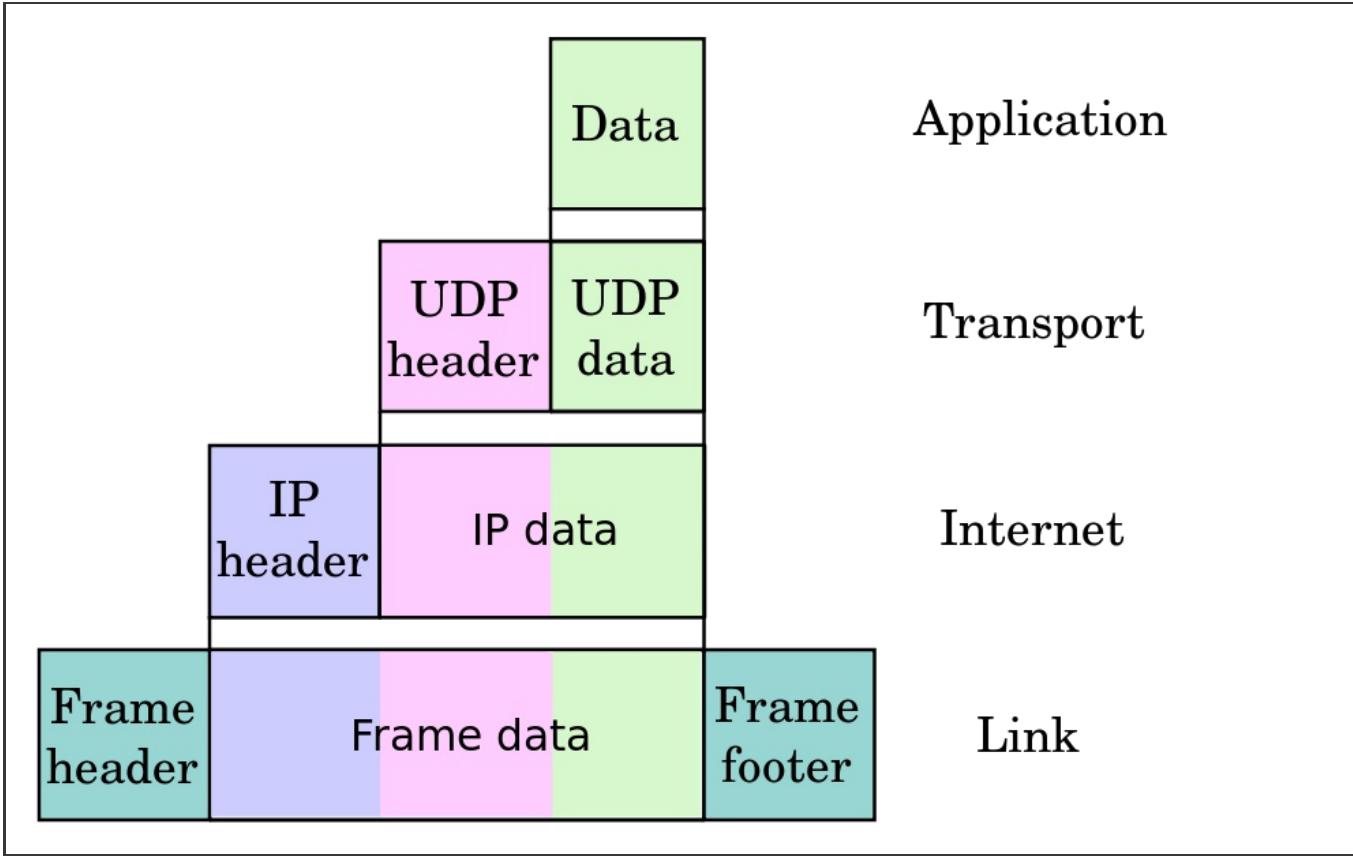
Network Topology



Data Flow



圖、OSI 與 TCP/IP 的層次對應圖


Data
Application
UDP header
UDP data
Transport
IP header
IP data
Internet
Frame header
Frame data
Frame footer
Link

圖、UDP 的封包

基於 HTTP 的 Web

HTTP 是 Web 全球資訊網 (萬維網) 的基礎協定，該協定架構在 TCP/IP 架構之上，屬於應用層的協定。構成 HTTP 的主要兩個應用程式是瀏覽器 (Browser) 與網站 (Web Server)。HTTP 是一個典型的 Client-Server (客戶端-伺服端) 架構的協定，使用者透過 Client 端的瀏覽器連結到 Server 的伺服器，然後由伺服端將結果以 HTML 的網頁格式傳回。 HTML 的網頁當中包含了許多超連結 (Hyperlink)，這些超連結連接到某些網址 (URL)，於是使用者可以透過瀏覽器中的超連結，進一步點選其他的網頁，進行網路瀏覽 (衝浪) 的行為。

Socket 函式庫

目前大部分的程式語言與平台 (像是 Java, C#, .NET, UNIX, Linux, ...) 都已經支援了 Socket 函式庫。但是由於語言與設計者的不同，這些 Socket 函式庫的使用方式都略有差異。在 C# 最常使用的是微軟 .NET 平台當中 的 Socket 函式庫，這個函式庫相當的成熟，除了將 TCP/IP 封裝成 Socket 函式庫之外，微軟甚至進一步將 HTTP 、加解密、甚至是高階的 Web 服務等機制，都包含在 .NET 平台當中，因此 C# 的程式設計師可以很輕易的寫出網路應用程式。

Socket 函式庫與 TCP/IP/HTTP 等層級的協定，是學習網路程式的基礎，接下來，我們將會進一步以範例說明網路程式的架構。

IP 層的程式設計

IP 是 TCP/IP 架構當中代表網址的層次，在撰寫 C# 網路程式時，幾乎每個程式都會用到 IP 層的物件，像

是 IPAddress，IPEndPoint 等。我們將在本文當中介紹這些物件的使用方式。

IPAddress 物件代表一個 IP 網址，像是 210.59.154.30 就是一個 IP。在一個大機構當中，由於有自身的內部網路，因此 IP 通常也分為對內與對外兩種。舉例而言，筆者在金門技術學院電腦的內部 IP 是 192.168.60.155，外部 IP 是 210.59.154.30。學校內部的電腦可以透過內部 IP 192.168.60.155 連接到該電腦，但是校外的電腦就只能透過外部 IP 210.59.154.30 連結到該電腦。

但是，IP 畢竟是不好記的數字，因此就發展出了 DNS (Domain Name Server, 網域名稱伺服器) 機制，用來將文字型的網址對應到數字型的 IP，這個文字型的網址稱為 URL (Universial Resource Locator)。

操作實驗

```
C:\Documents and Settings\ccc.CCC-KMIT2>ipconfig /all

Windows IP Configuration

        Host Name . . . . . . . . . . . . : ccc-kmit3
        Primary Dns Suffix  . . . . . . . :
        Node Type . . . . . . . . . . . . : Mixed
        IP Routing Enabled. . . . . . . . : No
        WINS Proxy Enabled. . . . . . . . : No
        DNS Suffix Search List. . . . . . : internal
```

Ethernet adapter 區域連線:

Connection-specific DNS Suffix . : internal
Description . . . . . : Broadcom NetLink (TM) Gigabit Ethernet
Physical Address. . . . . : 00-01-6C-95-20-52
Dhcp Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
IP Address. . . . . : 192.168.60.155
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.60.254
DHCP Server . . . . . : 192.168.1.252
DNS Servers . . . . . : 10.10.10.3
10.10.10.10
Primary WINS Server . . . . . : 10.10.10.20
Lease Obtained. . . . . : 2010年3月8日 上午 09:45:01
Lease Expires . . . . . : 2012年2月6日 上午 09:45:01

C:\Documents and Settings\ccc.CCC-KMIT2>nslookup ccc.kmit.edu.tw
Server: ns1.kmit.edu.tw
Address: 10.10.10.3

```
Name:    ccc.kmit.edu.tw
Address:  192.168.60.155

C:\Documents and Settings\ccc.CCC-KMIT2>nslookup tw.yahoo.com
Server:  ns1.kmit.edu.tw
Address:  10.10.10.3

Non-authoritative answer:
Name:    tw-tpe-fo.fyap.b.yahoo.com
Address:  119.160.246.241
Aliases:  tw.yahoo.com, tw-cidr.fyap.b.yahoo.com

C:\ccc>nslookup
Default Server:  ns1.kmit.edu.tw
Address:  10.10.10.3

> server dns.hinet.net
Default Server:  dns.hinet.net
Address:  168.95.1.1

> ccc.kmit.edu.tw
```

```
Server:  dns.hinet.net
Address:  168.95.1.1

Non-authoritative answer:
Name:    ccc.kmit.edu.tw
Address:  203.72.226.32
```

IP 層程式範例 1

範例：建立 IPAddress 與 IPEndPoint。

檔案： IPAddressTest.cs

```
using System;
using System.Net;

class IPAddressTest {
   static void Main() {
      // 建立一個 IP 位址 (IPAddress)。
      IPAddress ipAddr = IPAddress.Parse("210.59.154.30");
      Console.WriteLine("ipAddr="+ipAddr);
```

```
        // 建立一個 IP 終端 (IPEndPoint = ipAddress + port)。
        IPEndPoint ipEndPoint = new IPEndPoint(ipAddr, 80);
        Console.WriteLine("ipEndPoint=" + ipEndPoint);

        // 將IPEndPoint序列化為SocketAddress
        SocketAddress socketAddr = ipEndPoint.Serialize();
        Console.WriteLine("socketAddr=" + socketAddr);
    }
}
```

執行結果：

```
D:\CSharp>csc IPAddressTest.cs
Microsoft (R) Visual C# 2008 Compiler version 3.5.30729.1
for Microsoft (R) .NET Framework version 3.5
Copyright (C) Microsoft Corporation. All rights reserved.

D:\CSharp>IPAddressTest
ipAddr=210.59.154.30
ipEndPoint=210.59.154.30:80
socketAddr=InterNetwork:16:{0,80,210,59,154,30,0,0,0,0,0,0,0,0}
```

範例：取得主機名稱

檔案：IpToHost.cs

```
using System;
using System.Net;
using System.Net.Sockets;

class IpToHost
{
   static void Main(String[] args)
   {
      IPAddress ipAddr = IPAddress.Parse(args[0]);

      // 透過DNS找尋IP位址相對應之主機名稱
      IPHostEntry remoteHostEntry = Dns.GetHostEntry(ipAddr);

      Console.WriteLine("host of ip " + ipAddr + " is " + remoteHostEntry.HostName);
   }
}
```

執行結果：

```
D:\CSharp>csc IpToHost.cs
Microsoft (R) Visual C# 2008 Compiler version 3.5.30729.1
for Microsoft (R) .NET Framework version 3.5
Copyright (C) Microsoft Corporation. All rights reserved.

D:\CSharp>IpToHost 210.59.154.30
host of ip 210.59.154.30 is 210.59.154.30

D:\CSharp>IpToHost 119.160.246.241
host of ip 119.160.246.241 is w1.www.vip.tw1.yahoo.com
```

URL 、DNS 與網址剖析

範例：使用 DNS 查詢 IP

檔案：DnsTest.cs

```
using System;
using System.Net;
```

```
class DnsTest
{
   static void Main(String[] args)
   {
      IPHostEntry hostEntry = Dns.GetHostEntry(args[0]);

      // 由於主機有可能有一個以上的 Alias
      // 因此程式中以迴圈方式判斷 Aliases
      string[] aliasList = hostEntry.Aliases;

      for (int i = 0; i <= aliasList.Length - 1; i++)
      {
       Console.WriteLine("Alias "+i+" : "+aliasList[i]);
      }

      // 由於主機有可能有一個以上的 IP Address
      // 因此程式中以迴圈方式判斷 AddressList
      IPAddress[] addrList = hostEntry.AddressList;

      for (int i = 0; i <= addrList.Length - 1; i++)
```

```
        {
         Console.WriteLine("Address " + i + " : " + addrList[i]);
        }
    }
}
```

執行結果

```
D:\CSharp>DnsTest tw.yahoo.com
Address 0 : 119.160.246.241
```

範例：剖析網址 URL

檔案：UrlParseTest.cs

```
using System;
using System.Net;

class UrlParseTest
{
    static void Main(String[] args)
    {
```

```
        // 由於 DOS 的命令列會以 & 符號做命令分隔字元，因此、若以指令模式下，網址中的 & 之後會被
        System.Uri URL = new System.Uri("http://findbook.tw/search?keyword_type=keyword&t=xxx");
//      System.Uri URL = new System.Uri(args[0]);
        // System.Uri類別之屬性
        Console.WriteLine("AbsolutePath: " + URL.AbsolutePath);
        Console.WriteLine("AbsoluteUri: " + URL.AbsoluteUri);
        Console.WriteLine("Authority: " + URL.Authority);
        Console.WriteLine("Host: " + URL.Host);
        Console.WriteLine("Port: " + URL.Port);
        Console.WriteLine("LocalPath: " + URL.LocalPath);
        Console.WriteLine("IsDefaultPort: " + URL.IsDefaultPort);
        Console.WriteLine("IsFile: " + URL.IsFile);
        Console.WriteLine("PathAndQuery: " + URL.PathAndQuery);
        Console.WriteLine("Query: " + URL.Query);
        Console.WriteLine("Scheme: " + URL.Scheme);
        Console.WriteLine("UserEscaped: " + URL.UserEscaped);
//      Console.WriteLine("UserInfo: " + URL.UserInfo);
    }
}
```

執行結果

```
C:\ccc>csc UrlParseTest.cs
Microsoft (R) Visual C# 2008 Compiler version 3.5.30729.1
for Microsoft (R) .NET Framework version 3.5
Copyright (C) Microsoft Corporation. All rights reserved.

C:\ccc>UrlParseTest
AbsolutePath: /search
AbsoluteUri: http://findbook.tw/search?keyword_type=keyword&t=xxx
Authority: findbook.tw
Host: findbook.tw
Port: 80
LocalPath: /search
IsDefaultPort: True
IsFile: False
PathAndQuery: /search?keyword_type=keyword&t=xxx
Query: ?keyword_type=keyword&t=xxx
Scheme: http
UserEscaped: False
```

結語

微軟 C# 的 IP 層物件主要是 IPAddress 與 IPEndPoint，另外 IPHostEntry可以用來代表 URL，也可以用 Dns.GetHostEntry() 查詢主機名稱。這些是 C# 較常使用的 IP 層物件。

C# 網路程式：UDP 程式設計

UDP 簡介

UDP 是網路程式設計當中，最簡單的一種模式。本文將介紹如何使用 C# 撰寫 UDP 的『傳送-接收程式』。

UDP 程式的架構：使用 UDP 方式傳送訊息，由於封包是以一個一個的方式分別傳輸，先傳送者不一定會先到，甚至於沒有到達也不進行處理。由於這種方式不是連線導向的方式，因此不需要記住連線的 Socket，只要直接用 Socket 當中的 ReceiveFrom(data, ref Remote) 函數即可。

UDP 的程式必須有『傳送-接收』兩端，通常傳送端稱為 Client，接收端稱為 Server，以下是一個 UDP Client-Server 的 C# 程式架構。

客戶端：傳送訊息的 Client

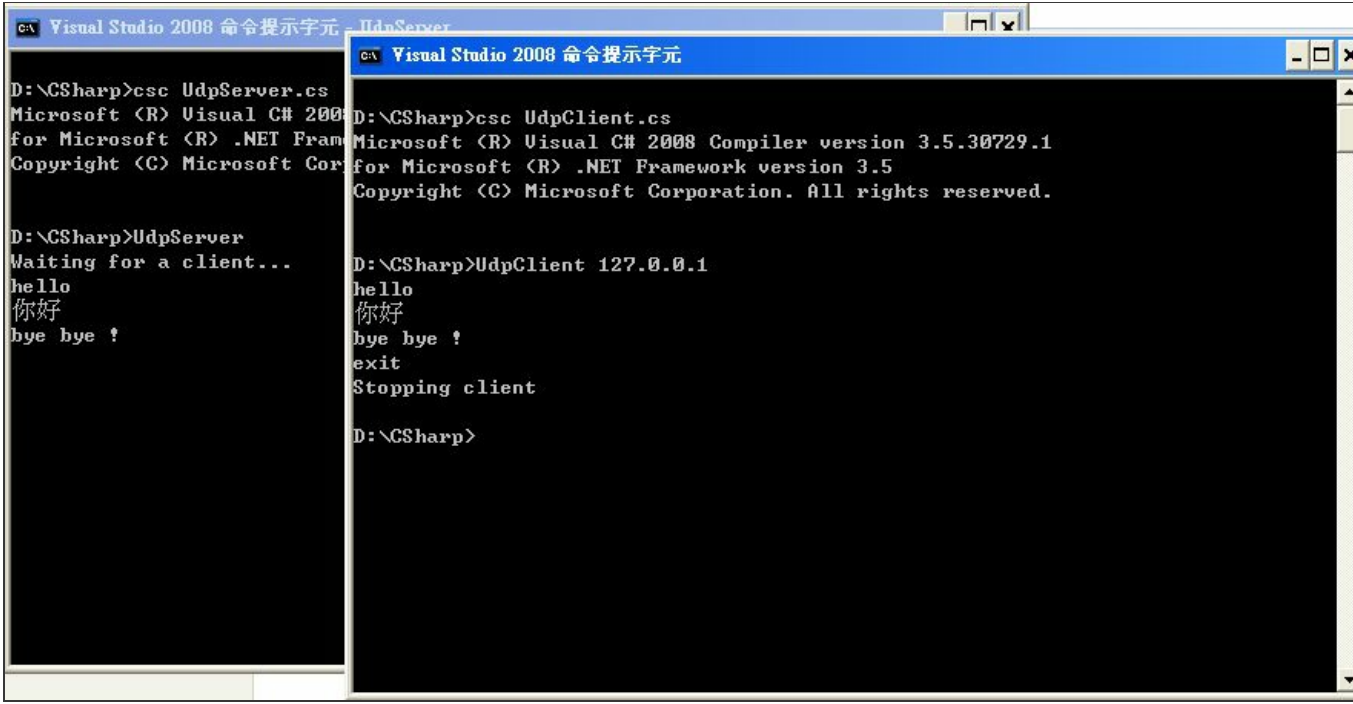
```
IPEndPoint ip = new IPEndPoint(address, port);
Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);
....
while(true) {
    ...
```

```
    server.SendTo(data, ip);
}
...
server.Close();
```

伺服端：接收訊息的 Server

```
ip = new IPEndPoint(IPAddress.Any, port);
socket = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);
socket.Bind(ipep);
...
while(true) {
  byte[] data = new byte[size];
  int recv = socket.ReceiveFrom(data, ref Remote);
  ...
}
```

為了說明 UDP 網路程式的設計方式，我們設計了一個單向的 UDP 聊天程式專案，這個專案包含 UdpClient.cs 與 UdpServer.cs 等兩個 C# 的程式。使用者可以在 UdpClient 當中輸入要傳送給 UdpServer 的訊息，而程式會忠實的將訊息從 Client 傳送到 Server 中。當使用者輸入 exit 的時候，Client 程式將會結束，以下是該程式執行時的畫面。



圖一、單向的 UDP 聊天專案的執行畫面

單向 UDP 訊息傳遞程式

以下是一個 UDP 客戶端 UdpClient，該客戶端會接受使用者的輸入，然後將訊息傳遞給伺服端的

UdpServer。

檔案：UdpClient.cs

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;

public class UdpClient
{
   public static void Main(string[] args)  // 主程式開始
   {
      // 連接到 args[0] 參數所指定的 Server，使用連接埠 5555
      IPEndPoint ipep = new IPEndPoint(IPAddress.Parse(args[0]), 5555);
      // 建立 Socket，連接到 Internet (InterNetwork)，使用 Udp 協定的 Datagram 方式 (Dgram)。
      Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);
      while(true)  // 不斷讀取鍵盤輸入，並傳送輸入訊息給伺服器。
      {
       string input = Console.ReadLine(); // 讀取鍵盤輸入
       if (input == "exit")   // 如果輸入為 exit，則強制離開程式。
```

```
            break;
            server.SendTo(Encoding.UTF8.GetBytes(input), ipep);    //將訊息以 UTF8 的方式編碼後傳出。
        }
        Console.WriteLine("Stopping client");   //印出 Stopping client 訊息。
        server.Close(); //關閉連線。
    }
}
```

以下是一個 Udp 的伺服端，利用無窮迴圈接收上述客戶端傳來的訊息，然後列印在螢幕上。

檔案：UdpServer.cs

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;

public class UdpServer
{
    public static void Main()
    {
```

```
        // 開啟伺服器的 5555 連接埠，用來接收任何傳送到本機的訊息。
        IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 5555);
        // 建立接收的 Socket，並使用 Udp 的 Datagram 方式接收。
        Socket newsock = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);
        // 綁定 Socket (newsock) 與 IPEndPoint (ipep)，讓 Socket 接收 5555 埠的訊息。
        newsock.Bind(ipep);
        Console.WriteLine("Waiting for a client...");// 顯示 Waiting for client ...。
        // 建立 Remote 物件以便取得封包的接收來源的 EndPoint 物件。
        IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);
        EndPoint Remote = (EndPoint)(sender);
        while(true) // 無窮迴圈，不斷接收訊息並顯示到螢幕上。
        {
         byte[] data = new byte[1024]; // 設定接收緩衝區的陣列變數。
         int recv = newsock.ReceiveFrom(data, ref Remote); // 接收對方傳來的封包。
         // 將該封包以 UTF8 的格式解碼為字串，並顯示到螢幕上。
         Console.WriteLine(Encoding.UTF8.GetString(data, 0, recv));
        }
    }
}
```

程式說明：

在上述程式當中，讀者會看到兩個無窮迴圈，這在網路程式設計領域是很常見的。Server 通常是一個由無窮迴圈所構成的程式，該程式不斷的接收由 Client 所傳來的訊息，然後進行處理與顯示。

程式中的 IPEndPoint 所代表的就是 TCP/IP 協定中的 IP 層，也就是網址。建構函數 new IPEndPoint(address, port) 有兩個參數，第一個是 IP 地址 (address)，第二個是連接埠 (port)。這個物件在撰寫 Client 與 Server 時都會用到。

在 UdpClient 的程式中，我們直接利用下列程式連接到 Server，您可以使用

```
IPEndPoint ipep = new IPEndPoint(IPAddress.Parse(args[0]), 5555);
```

UdpServer 程式中的 new IPEndPoint(IPAddress.Any, 5555) 會開啟該伺服器電腦的 5555 這個連接埠 (port)，讓其他程式可以透過網路傳送訊息給該程式。一但某程式開啟了特定連接埠，就會霸佔該連接埠。作業系統不會允許其他程式再度開啟這個連接埠，因為該連接埠已經被用掉了。

讀者可能會對其中的 IPAddress.Any 的用意感到納悶，為何不是指定本機的 IP，而是用 IPAddress.Any 呢？

參考文獻

- MSDN/.NET Framework 類別庫/Socket 類別 -- http://msdn.microsoft.com/zh-tw/library/system.net.sockets.socket(VS.80).aspx
- Socket.ReceiveFrom 方法 -- http://msdn.microsoft.com/zh-tw/library/wdfskwcy(VS.80).aspx

C# 網路程式：TCP 程式設計

TCP 簡介

TCP 是一個連線導向的網路傳輸協定，程式通常在斷線之前會一直記住這個連線。在我們使用 Socket 函式庫設計 TCP 網路程式時，通常會讓一個 Thread 負責處理一條連線，這樣可以讓問題變得較為單純，因為不需要用表格去記住連線。

所以，在學習 TCP 程式設計之前，我們有必要先複習一下 Thread (台灣稱為執行緒、中國大陸稱為線程) 的程式設計方式，若讀者對 Thread 尚不熟悉，請先閱讀本書的 Thread 章節。

TCP 的程式架構

TCP 客戶端的通常在連上伺服端後，就會用一個迴圈透過 Socket.send 函數傳送訊息，以下是一個典型的 TCP 客戶端程式。

客戶端：Client

```
IPEndPoint ip = new IPEndPoint(IPAddress.Parse(address), port);
Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
server.Connect(ip);
```

```
...
while(true) {
    ...
    server.Send(data);
    ...
}
...
server.Shutdown(SocketShutdown.Both);
server.Close();
```

TCP 伺服端通常較為複雜，首先在利用 Socket 中的 bind() 函數佔用特定的 port 之後，會再進入一個無窮迴圈，在迴圈中利用 socket.Accept() 接受客戶端的連線，此時會產生一個新的 Socket 物件。通常，我們會建立一個新的 Thread 去處理這個連線，因此在下列程式中，我們建立了 TcpListener 這個物件，以便讓一個 Listener 處理一個 Socket 連線。

伺服端：Server

```
public class TcpServer
{
   public static void Main()
   {
      IPEndPoint ip = new IPEndPoint(IPAddress.Any, port);
```

```
        Socket socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
        socket.Bind(ip);
        socket.Listen(10);
        while(true)
        {
            Socket client = socket.Accept();
            TcpListener listener = new TcpListener(client); // 建立 Listener 物件處理這個連線。
            Thread thread = new Thread(new ThreadStart(listener.run)); // 建立 listener.run 的 Thread。
            thread.Start(); // 啟動 Thread
            ...
        }
        ...
    }
}

public class TcpListener {
    Socket socket;

    public TcpListener(Socket s)  { socket = s; }

    public void run() { // Thread 的主要函數
```

```
        while (true) {
            ...
            byte[] data = new byte[1024];
            int recv = socket.Receive(data); //接收資料
            ...
        }
        socket.Close();
    }
}
```

以上就是一個典型的 TCP 程式架構，該架構包含了『客戶端-伺服端』兩個部分，客戶端透過 Socket.send 傳送訊息，而伺服端則會針對每一個連線，建立新的 Thread ，以便處理連線的互動關係。這種架構對 TCP 而言是較為簡單的，有幾個連線就建立幾個 Thread。由於這種架構不需要用表格記住連線，因此簡化了伺服端的管理，讓程式設計更為容易。

在理解了這個架構後，讓我們來看看幾個真實的程式範例：

TCP 單向的訊息傳遞程式

由於網路程式必須有連線，因此至少要有兩個程式才能運作，通常主動連線的一方稱為 Client，被動等待連線的一方稱為 Server，這就是所謂的 Client - Server 架構。

在本範例中，我們撰寫了一個 TcpClient 與一個 TcpServer 程式，TcpClient 連線到 TcpServer 之後，會將使

用者所輸入的文字傳司送給 TcpServer。當 TcpServer 收到這些文字之後會印出在螢幕上，這個範例示範了一個極為簡單的網路程式架構。

檔案：TcpClient1.cs

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;

public class TcpClient
{
   public static void Main(string[] args)
   {
      IPEndPoint ipep = new IPEndPoint(IPAddress.Parse(args[0]), 20);

      Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
      server.Connect(ipep);

      while(true)
      {
```

```
        string input = Console.ReadLine();
        if (input == "exit")
        break;
        byte[] data = Encoding.UTF8.GetBytes(input);
//   byte[] data = Encoding.ASCII.GetBytes(input);
        server.Send(data);
       }
       Console.WriteLine("Disconnecting from server...");
       server.Shutdown(SocketShutdown.Both);
       server.Close();
   }
}
```

檔案：TcpServer1.cs

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
```

```
public class TcpServer
{
   public static void Main()
   {
      IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 20);

      Socket newsock = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

      newsock.Bind(ipep);
      newsock.Listen(10);

      while(true)
      {
       Socket client = newsock.Accept();
       IPEndPoint clientep = (IPEndPoint) client.RemoteEndPoint;
       Console.WriteLine("Client End Point = " + clientep);
       // create a new thread and then receive message.
       TcpListener listener = new TcpListener(client);
       Thread thread = new Thread(new ThreadStart(listener.run));
       thread.Start();
      }
```

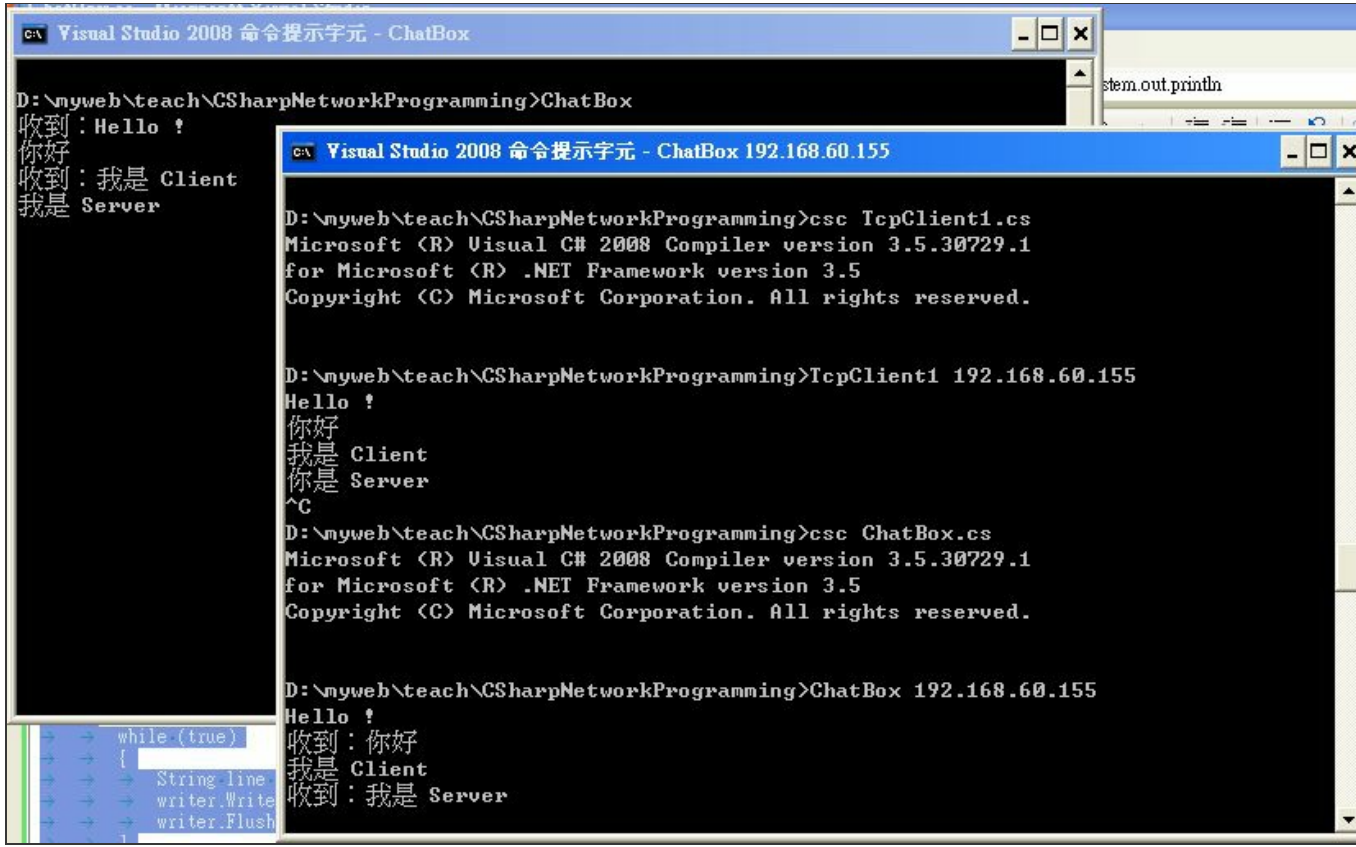
```
//    newsock.Close();
    }
}

public class TcpListener {
    Socket socket;

    public TcpListener(Socket s)
    {
        socket = s;
    }

    public void run()
    {
        try {
          while (true)
          {
            byte[] data = new byte[1024];
            int recv = socket.Receive(data);
            if (recv == 0) break;
            Console.WriteLine(Encoding.UTF8.GetString(data, 0, recv));
```

```
        }
        socket.Close();
      } catch (Exception e) {
        Console.WriteLine("Client "+socket.RemoteEndPoint+" Error : close");
        Console.WriteLine(e);
      }
    }
}
```

上述程式的執行結果如下所示：

TCP 雙向聊天程式

聊天程式是學習網路程式設計很好的入門題目之一，在本文中，我們將示範如何用 C# 的 Socket 函式庫設計一個網路聊天程式。以下是這個程式的執行時的一個畫面，讀者可以看到我們在兩個命令列視窗中執行同一個 ChatBox 程式，左上角是 Server 程式，右下角是 Client 程式。


Visual Studio 2008 命令提示字元 - ChatBox
D:\myweb\teach\CSharpNetworkProgramming>ChatBox
收到：Hello !
你好
收到：我是 Client
我是 Server
Visual Studio 2008 命令提示字元 - ChatBox 192.168.60.155
D:\myweb\teach\CSharpNetworkProgramming>csc TcpClient1.cs
Microsoft (R) Visual C# 2008 Compiler version 3.5.30729.1
for Microsoft (R) .NET Framework version 3.5
Copyright (C) Microsoft Corporation. All rights reserved.
D:\myweb\teach\CSharpNetworkProgramming>TcpClient1 192.168.60.155
Hello !
你好
我是 Client
你是 Server
^C
D:\myweb\teach\CSharpNetworkProgramming>csc ChatBox.cs
Microsoft (R) Visual C# 2008 Compiler version 3.5.30729.1
for Microsoft (R) .NET Framework version 3.5
Copyright (C) Microsoft Corporation. All rights reserved.
D:\myweb\teach\CSharpNetworkProgramming>ChatBox 192.168.60.155
Hello !
收到：你好
我是 Client
收到：我是 Server
while (true)
String line
writer.Write
writer.Flush

聊天程式的執行畫面

在上圖中，Client 可以傳送訊息給 Server ，而 Server 也可傳送訊息給 Client，由於雙方在這個傳送接收的動作上幾乎是相同的，因此在寫程式時就將這些動作封裝後共用，可以讓程式更為精簡，這就是 P2P 架構的程式。

當然，這個程式也可以放在兩台不同的電腦上執行，其結果將與上述結果類似，只是無法同時在一台電腦上看到輸入與輸出的結果而已。

以下是上述雙向聊天程式 ChatBox 的原始程式碼，其中的 TcpListener 程式是 Client-Server 雙方所共用的部分，因此獨立出來變成一個類別，並且分別在 Client 與 Server 主程式當中都用 new TcpListener() 的方式呼叫。此時雙方都會建立一個寫入迴圈 (outLoop) 與讀取回圈 (inLoop)，寫入迴圈不斷等待鍵盤的輸入，並在有輸入時將訊息傳遞給對方。而讀取回圈則是不斷接收對方所傳來的訊息並顯示在螢幕上。

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.IO;
using System.Threading;

class ChatBox
{
```

```
int port = 20;

public static void Main(String[] args)
{
    ChatBox chatBox = new ChatBox();
    if (args.Length == 0)
     chatBox.ServerMain();
    else
     chatBox.ClientMain(args[0]);
}

public void ServerMain()
{
    IPEndPoint ipep = new IPEndPoint(IPAddress.Any, port);
    Socket newsock = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
    newsock.Bind(ipep);
    newsock.Listen(10);
    Socket client = newsock.Accept();
    new TcpListener(client); // create a new thread and then receive message.
    newsock.Close();
}
```

```
    public void ClientMain(String ip)
    {
        IPEndPoint ipep = new IPEndPoint(IPAddress.Parse(ip), port);
        Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
        server.Connect(ipep);
        new TcpListener(server);
        server.Shutdown(SocketShutdown.Both);
        server.Close();
    }

}

public class TcpListener
{
    Socket socket;
    Thread inThread, outThread;
    NetworkStream stream;
    StreamReader reader;
    StreamWriter writer;
```

```
public TcpListener(Socket s)
{
    socket = s;
    stream = new NetworkStream(s);
    reader = new StreamReader(stream);
    writer = new StreamWriter(stream);
    inThread = new Thread(new ThreadStart(inLoop));
    inThread.Start();
    outThread = new Thread(new ThreadStart(outLoop));
    outThread.Start();
    inThread.Join(); // 等待 inThread 執行續完成，才離開此函數。
    // (注意、按照 inLoop 的邏輯，這個函數永遠不會跳出，因為 inLoop 是個無窮迴圈)。
}

public void inLoop()
{
    while (true)
    {
     String line = reader.ReadLine();
     Console.WriteLine("收到：" + line);
    }
```

```
    }

    public void outLoop()
    {
        while (true)
        {
         String line = Console.ReadLine();
         writer.WriteLine(line);
         writer.Flush();
        }
    }
}
```

TCP 多人聊天室（視窗版）

專案程式下載：https://dl.dropbox.com/u/101584453/cs/code/ChattingRoom.zip

使用方法

```
// ------------------------------------------------------------
// 共有四個程式檔案
//
```

```
// 程式檔 : ChatLib.cs
// 程式檔 : ChattingServer.cs
// 程式檔 : FormChatClient.cs
// 程式檔 : FormChatClient.Designer.cs
//
// 編譯方式 :
//   步驟 1 : csc ChatServer.cs ChatLib.cs
//          會產生 ChattingServer.exe 檔
//
//   步驟 2 : csc WinChatClient.cs FormChatClient.cs FormChatClilent.Designer.cs ChatLib.cs
//          會產生 WinChatClient.exe 檔
//
// 執行方式 :
//   步驟 1 : 執行 ChatServer.exe
//   步驟 2 : 執行 WinChatClient.exe (使用者 1)
//   步驟 3 : 執行 WinChatClient.exe (使用者 2)
//
// 如此，兩個 WinChatClient.exe 的視窗間即可透過本機聊天
// ------------------------------------------------------------
```

檔案：ChatLib.cs

```
// ------------------------ ChatLib.cs -------------------------
using System;
using System.Net;
using System.Net.Sockets;
using System.IO;
using System.Threading;

namespace ChattingRoom
{
    public class ChatSetting
    {
        public static String serverIp = "192.168.100.172";
        public static int port = 3766;
    }

    public delegate String StrHandler(String str);

    public class ChatSocket
    {
        public Socket socket;
```

```
public NetworkStream stream;
public StreamReader reader;
public StreamWriter writer;
public StrHandler inHandler;
public EndPoint remoteEndPoint;
public bool isDead = false;

public ChatSocket(Socket s)
{
    socket = s;
    stream = new NetworkStream(s);
    reader = new StreamReader(stream);
    writer = new StreamWriter(stream);
    remoteEndPoint = socket.RemoteEndPoint;
}

public String receive()
{
    return reader.ReadLine();
}
```

```
public ChatSocket send(String line)
{
    writer.WriteLine(line);
    writer.Flush();
    return this;
}

public static ChatSocket connect(String ip)
{
    IPEndPoint ipep = new IPEndPoint(IPAddress.Parse(ip), ChatSetting.port);

    Socket socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
    socket.Connect(ipep);
    return new ChatSocket(socket);
}

public Thread newListener(StrHandler pHandler)
{
    inHandler = pHandler;

    Thread listenThread = new Thread(new ThreadStart(listen));
```

```
        listenThread.Start();
        return listenThread;
    }

    public void listen()
    {
        try
        {
            while (true)
            {
                String line = receive();
                inHandler(line);
            }
        }
        catch (Exception ex)
        {
            isDead = true;
            Console.WriteLine(ex.Message);
        }
    }
}
```

```
}
```

檔案：ChattingServer.cs

```
// ------------------------ ChatServer.cs ------------------
using System;
using System.Net;
using System.Net.Sockets;
using System.IO;
using System.Threading;
using System.Collections.Generic;

namespace ChattingRoom
{
   public class ChatServer
   {
      List<ChatSocket> clientList = new List<ChatSocket>();

      public static void Main(String[] args)
      {
         ChatServer chatServer = new ChatServer();
```

```
        chatServer.run();
    }

    public void run()
    {
        IPEndPoint ipep = new IPEndPoint(IPAddress.Any, ChatSetting.port);

        Socket newsock = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

        newsock.Bind(ipep);
        newsock.Listen(10);

        while (true)
        {
            Socket socket = newsock.Accept();
            Console.WriteLine("接受一個新連線!");
            ChatSocket client = new ChatSocket(socket);
            try
            {
                clientList.Add(client);
                client.newListener(processMsgComeIn);
```

```
            }
            catch
            {
            }
//            clientList.Remove(client);
        }
        //    newsock.Close();
    }

    public String processMsgComeIn(String msg)
    {
        Console.WriteLine("收到訊息："+msg);
        broadCast(msg);
        return "OK";
    }

    public void broadCast(String msg)
    {
        Console.WriteLine("廣播訊息給 " + msg+" 線上使用者共"+clientList.Count+"個人!");
        foreach (ChatSocket client in clientList)
        {
```

```
        if (!client.isDead) {
        Console.WriteLine("Send to "+client.remoteEndPoint.ToString()+":"+msg);
        client.send(msg);
        }
          }
        }
    }
}
```

檔案：FormChatClient.cs

```
// ------------------------ FormChatClient.cs ------------------
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace ChattingRoom
```

```
{
    public partial class FormChatClient : Form
    {
        ChatSocket client;
        StrHandler msgHandler;

        public FormChatClient()
        {
            InitializeComponent();

            msgHandler = this.addMsg;
        }

        private void buttonSend_Click(object sender, EventArgs e)
        {
            sendMsg();
        }

        private void textBoxMsg_KeyPress(object sender, KeyPressEventArgs e)
        {
         if (e.KeyChar == '\r')
```

```
        sendMsg();
    }

    public String user() {
        return textBoxUser.Text.Trim();
    }

    public String msg() {
        return textBoxMsg.Text;
    }

    public void sendMsg()
    {
        if (user().Length == 0)
        {
            MessageBox.Show("請輸入使用者名稱!");
            return;
        }
        if (client == null) {
            client = ChatSocket.connect(ChatSetting.serverIp);
            client.newListener(processMsgComeIn);
```

```
            client.send(user() + " : 新使用者進入!");
            textBoxUser.Enabled = false;
        }
        if (msg().Length > 0) {
            client.send(user()+" : "+msg());
     textBoxMsg.Text = "";
     }
    }

    public String processMsgComeIn(String msg)
    {
        this.Invoke(msgHandler, new Object[] { msg });
        return "OK";
    }

    public String addMsg(String msg)
    {
        richTextBoxBoard.AppendText(msg + "\n");
        return "OK";
    }
}
```

```
}
```

檔案：FormChatClient.Designer.cs

```
// ------------------------ FormChatClient.Designer.cs ----------
namespace ChattingRoom
{
    partial class FormChatClient
    {
        /// <summary>
        /// 設計工具所需的變數。
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// 清除任何使用中的資源。
        /// </summary>
        /// <param name="disposing">如果應該公開 Managed 資源則為 true，否則為 false。</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
```

```
        {
            components.Dispose();
        }
        base.Dispose(disposing);
    }

    #region Windows Form 設計工具產生的程式碼

    /// <summary>
    /// 此為設計工具支援所需的方法 - 請勿使用程式碼編輯器修改這個方法的內容。
    /// 
    /// </summary>
    private void InitializeComponent()
    {
        this.panelInput = new System.Windows.Forms.Panel();
        this.labelSaid = new System.Windows.Forms.Label();
        this.textBoxUser = new System.Windows.Forms.TextBox();
        this.textBoxMsg = new System.Windows.Forms.TextBox();
        this.buttonSend = new System.Windows.Forms.Button();
        this.panelPadding1 = new System.Windows.Forms.Panel();
        this.panelMsg = new System.Windows.Forms.Panel();
```

```
        this.richTextBoxBoard = new System.Windows.Forms.RichTextBox();
        this.panelInput.SuspendLayout();
        this.panelMsg.SuspendLayout();
        this.SuspendLayout();
        //
        // panelInput
        //
        this.panelInput.Controls.Add(this.labelSaid);
        this.panelInput.Controls.Add(this.textBoxUser);
        this.panelInput.Controls.Add(this.textBoxMsg);
        this.panelInput.Controls.Add(this.buttonSend);
        this.panelInput.Controls.Add(this.panelPadding1);
        this.panelInput.Dock = System.Windows.Forms.DockStyle.Bottom;
        this.panelInput.Location = new System.Drawing.Point(0, 283);
        this.panelInput.Name = "panelInput";
        this.panelInput.Size = new System.Drawing.Size(494, 64);
        this.panelInput.TabIndex = 0;
        //
        // labelSaid
        //
        this.labelSaid.AutoSize = true;
```

```
this.labelSaid.Font = new System.Drawing.Font("DFKai-SB", 15.75F, System.Drawing.FontStyle.Regular, S
this.labelSaid.Location = new System.Drawing.Point(94, 17);
this.labelSaid.Name = "labelSaid";
this.labelSaid.Size = new System.Drawing.Size(54, 21);
this.labelSaid.TabIndex = 4;
this.labelSaid.Text = "說：";
//
// textBoxUser
//
this.textBoxUser.Location = new System.Drawing.Point(3, 17);
this.textBoxUser.Name = "textBoxUser";
this.textBoxUser.Size = new System.Drawing.Size(88, 22);
this.textBoxUser.TabIndex = 3;
//
// textBoxMsg
//
this.textBoxMsg.Location = new System.Drawing.Point(154, 17);
this.textBoxMsg.Name = "textBoxMsg";
this.textBoxMsg.Size = new System.Drawing.Size(250, 22);
this.textBoxMsg.TabIndex = 2;
this.textBoxMsg.KeyPress += new System.Windows.Forms.KeyPressEventHandler(this.textBoxMsg_Ke
```

```
            //
            // buttonSend
            //
            this.buttonSend.Location = new System.Drawing.Point(410, 14);
            this.buttonSend.Name = "buttonSend";
            this.buttonSend.Size = new System.Drawing.Size(81, 28);
            this.buttonSend.TabIndex = 1;
            this.buttonSend.Text = "送出";
            this.buttonSend.UseVisualStyleBackColor = true;
            this.buttonSend.Click += new System.EventHandler(this.buttonSend_Click);
            //
            // panelPadding1
            //
            this.panelPadding1.Dock = System.Windows.Forms.DockStyle.Bottom;
            this.panelPadding1.Location = new System.Drawing.Point(0, 48);
            this.panelPadding1.Name = "panelPadding1";
            this.panelPadding1.Size = new System.Drawing.Size(494, 16);
            this.panelPadding1.TabIndex = 0;
            //
            // panelMsg
            //
```

```
this.panelMsg.Controls.Add(this.richTextBoxBoard);
this.panelMsg.Dock = System.Windows.Forms.DockStyle.Fill;
this.panelMsg.Location = new System.Drawing.Point(0, 0);
this.panelMsg.Name = "panelMsg";
this.panelMsg.Size = new System.Drawing.Size(494, 283);
this.panelMsg.TabIndex = 1;
//
// richTextBoxBoard
//
this.richTextBoxBoard.Dock = System.Windows.Forms.DockStyle.Fill;
this.richTextBoxBoard.Location = new System.Drawing.Point(0, 0);
this.richTextBoxBoard.Name = "richTextBoxBoard";
this.richTextBoxBoard.Size = new System.Drawing.Size(494, 283);
this.richTextBoxBoard.TabIndex = 0;
this.richTextBoxBoard.Text = "";
//
// FormChatClient
//
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 12F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(494, 347);
```

```
        this.Controls.Add(this.panelMsg);
        this.Controls.Add(this.panelInput);
        this.Name = "FormChatClient";
        this.Text = "C# 聊天室";
        this.panelInput.ResumeLayout(false);
        this.panelInput.PerformLayout();
        this.panelMsg.ResumeLayout(false);
        this.ResumeLayout(false);

    }

    #endregion

    private System.Windows.Forms.Panel panelInput;
    private System.Windows.Forms.Panel panelMsg;
    private System.Windows.Forms.TextBox textBoxMsg;
    private System.Windows.Forms.Button buttonSend;
    private System.Windows.Forms.Panel panelPadding1;
    private System.Windows.Forms.RichTextBox richTextBoxBoard;
    private System.Windows.Forms.Label labelSaid;
```

```
        private System.Windows.Forms.TextBox textBoxUser;
    }
}
```

檔案：Program.cs (主程式)

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace ChattingRoom
{
    static class Program
    {
        /// <summary>
        /// 應用程式的主要進入點。
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
```

```
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new FormChatClient());
            Application.Exit();
        }
    }
}
```

C# 網路程式：HTTP 程式設計（1）-- Server

HTTP 協定簡介

HTTP 協定是 Web 的基礎，Web 是一個典型的 Client-Server 架構，主要由 HTTP+URL+HTML 所組成。

Web Server (網站伺服器) 是 WWW 網路的基礎， 1991 年 Tim Burner Lee 發明 HTML 與 URL 後，就自己寫了第一個 Web Server，導至後來 Web 網路的興起，因此、Tim Burner Lee被尊稱為 WWW 之父，然而、對於一般程式設計人員而言， Web Server 的設計方法卻是個難解的謎，本篇文章將以一個簡單而完整的 Java 程式，讓大家了解 Web Server 的設計原理。

一個最簡單的 Web Server 之功能包含下列三個步驟:

- 步驟一 : 接收瀏覽器所傳來的網址。
- 步驟二 : 取出相對應的檔案。
- 步驟三 : 將檔案內容傳回給瀏覽器。

然而、在這個接收與傳回的過程中，所有的資訊都必須遵照固定的格式，規範這個接收/傳送格式的協定，稱為超文字傳送協定 (Hyper Text Transfer Protocol)，簡稱為 HTTP 協定。

HTTP 協定格式的基礎，乃是建構在網址 URL 上的傳輸方式，早期只能用來傳送簡單的 HTML 檔案，後來經擴充後也可以傳送 其他類型的檔案，包含 影像、動畫、簡報、Word 文件等。

在本文中，我們將先簡介 HTTP 協定的訊息內容，然後在介紹如何以 Java 語言實作 HTTP 協定，以建立一個簡單的 Web Server。

HTTP 表頭：

當你在瀏覽器上打上網址(URL)後，瀏覽器會傳出一個HTTP訊息給對應的 Web Server，Web Server 再接收到這個訊息後， 根據網址取出對應的檔案，並將該檔案以 HTTP 格式的訊息傳回給瀏覽器，以下是這個過程的一個範例。

豬小弟上網，在瀏覽器中打上 http://ccc.kmit.edu.tw/index.htm，於是、瀏覽器傳送下列訊息給 ccc.kmit.edu.tw 這台電腦。

```
GET /index.htm HTTP/1.0
Accept: image/gif, image/jpeg, application/msword, */*
Accept-Language: zh-tw
User-Agent: Mozilla/4.0
Content-Length:
Host: ccc.kmit.edu.tw
Cache-Control: max-age=259200
Connection: keep-alive
```

當 ccc.kmit.edu.tw 電腦上的 Web Server 程式收到上述訊息後，會取出指定的路徑 /index.htm ，然後根據預設的網頁根目錄 (假設為 c:)，合成一個 c:.htm 的絕對路徑，接著從磁碟機中取出該檔案，並傳回下列訊息給豬小弟的瀏覽器。

```
HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 438

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

其中第一行 HTTP/1.0 200 OK 代表該網頁被成功傳回，第二行 Content-Type: text/html 代表傳回文件為 HTML 檔案， Content-Length: 438 代表該 HTML 檔案的大小為 438 位元組。

HelloServer: 永遠傳回 Hello 的 WebServer

檔案 HelloServer.cs

```
using System;
using System.Net;
```

```
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.IO;

public class HttpServer
{
   public static void Main()
   {
      IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 80);

      Socket newsock = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

      newsock.Bind(ipep);
      newsock.Listen(10);

      while(true)
      {
       Socket client = newsock.Accept();
       IPEndPoint clientep = (IPEndPoint) client.RemoteEndPoint;
```

```
        // create a new thread and then receive message.
        HttpListener listener = new HttpListener(client);
        Thread thread = new Thread(new ThreadStart(listener.run));
        thread.Start();
       }
//    newsock.Close();
   }
}

public class HttpListener {
   Socket socket;

   public HttpListener(Socket s)
   {
      socket = s;
   }

   public void run()
   {
      String msg = "Hello!";
      String helloMsg = @"HTTP/1.0 200 OK\nContent-Type: text/plain\nContgent-Length: "+msg.Length+"\n\n"+ms
```

```
        NetworkStream stream = new NetworkStream(socket);
        StreamReader reader = new StreamReader(stream);
        String header = "";
        while (true)
        {
         String line = reader.ReadLine();
         Console.WriteLine(line);
         if (line.Trim().Length==0)
         break;
         header += line+"\n";
        }
        socket.Send(Encoding.UTF8.GetBytes(helloMsg));
        socket.Close();
    }
}
```

完整的 Web Server

專案程式下載：https://dl.dropbox.com/u/101584453/cs/code/WebServer.zip

請注意：此範例必須在專案中加入 System.Web 套件引用。

根據 HTTP 協定，我們以 C# 實作了一個 Web Server 程式 (WebServer.cs)，該程式是利用一個稱為 Socket 的物件來實作的，這個物件位於 C# 的網路函式庫 System.Net 中。

Socket 是根據博克萊 (U.C.Berkley) 大學早期發展的 Socket 概念寫成的，其設計理念是是將網路傳輸類比成檔案讀取與寫入 (傳送的動作被視為是寫入/接收的動作被視為是讀取)，如此、傳送與接收就簡化為程式人員比較容易懂的 讀取與寫入，降低了網路程式的學習困難度。

要使用 Socket 的方式寫網路程式，首先要登記網路的埠號 (port)，將該 port 占領下來，以防止其他程式使用該 port 而互相干擾，HTTP 協定所預設使用的是 port 80。

一但完成登記，就可以開始接受瀏覽器的請求，並根據請求回傳檔案訊息，以下程式為其 (接收/傳送) 程序的核心程式。

這個最簡單版以 Socket 的方式，不斷讀取資料直到發現有一空白行即結束，然而、這樣的程式是過度簡化的結果，無法處理有 POST 訊息的狀況，然而、何謂 POST 訊息呢？

所謂 POST 訊息、乃是 HTML 為了傳遞較大量的填表資料，所發展出來的一種訊息格式，以下是POST訊息的一個範例：

```
POST /getMsg.asp HTTP/1.0
Accept: image/gif, image/jpeg, application/msword, */*
Accept-Language: zh-tw
```

```
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0
Content-Length: 66
Host: ccc.kmit.edu.tw
Cache-Control: max-age=259200
Connection: keep-alive

user=ccc&password=1234&msg=Hello+%21+%0D%0AHow+are+you+%21%0D%0A++
```

其中的HTTP的訊息開頭以 POST 取代原來的 GET ，並且多了一個 Content-Length:66 的欄位，該欄位指示了訊息結尾會有 66 個位元組的填表資料，這些資料會被編碼成特輸的格式以利在網路上傳遞。

一但取得了瀏覽器傳來的 GET 或 POST 訊息後，我們就可以根據其訊息，決定回應的方式，在 WebServer.java 中，我們只是單純的將對應的檔案取出，並附在回應的訊息表頭後傳回，其程式碼如下。

以上就是 Web Server 的簡單設計方式，若想了解更多細節，請直接閱讀 WebServer.java 檔並執行之，執行時請安裝好 Visual Studio 後，並於 WebServer.cs 的存檔路徑上打 csc WebServer.cs, 之後再打 WebServer 即可啟動，其執行畫面如下：

檔案： WebServer.cs

```
using System;
```

```
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.IO;
using System.Web;

public class WebServer
{
  public static void Main()
  {
    IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 8085);

    Socket newsock = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

    newsock.Bind(ipep);
    newsock.Listen(10);

    while(true)
    {
      Socket client = newsock.Accept();
```

```
        IPEndPoint clientep = (IPEndPoint) client.RemoteEndPoint;

        // create a new thread and then receive message.
        HttpListener listener = new HttpListener(client);
        Thread thread = new Thread(new ThreadStart(listener.run));
        thread.Start();
      }
    }
}

class HttpListener
{
    String[] map ={"mpeg=video/mpeg", "mpg=video/mpeg", "wav=audio/x-wav", "jpg=image/jpeg",
"gif=image/gif", "zip=application/zip", "pdf=application/pdf", "xls=application/vnd.ms-excel",
"ppt=application/vnd.ms-powerpoint", "doc=application/msword", "htm=text/html",
"html=text/html", "css=text/plain", "vbs=text/plain", "js=text/plain", "txt=text/plain",
"java=text/plain"};
    Socket socket;
    NetworkStream stream;
    String header;
    String root = ".";
```

```
public HttpListener(Socket s)
{
    socket = s;
}

public void run()
{
    stream = new NetworkStream(socket);
    request();
    response();
    stream.Close();
}

public void send(String str) {
    socket.Send(Encoding.UTF8.GetBytes(str));
}

public static String innerText(String pText, String beginMark, String endMark)
{
    int beginStart = pText.IndexOf(beginMark);
```

```
    if (beginStart < 0) return null;
    int beginEnd = beginStart + beginMark.Length;
    int endStart = pText.IndexOf(endMark, beginEnd);
    if (endStart < 0) return null;
    return pText.Substring(beginEnd, endStart - beginEnd);
}

public void request()
{
    try {
     StreamReader reader = new StreamReader(stream);
     header = "";
     while (true)
     {
     String line = reader.ReadLine();
     Console.WriteLine(line);
     if (line.Trim().Length == 0)
     break;
     header += line + "\n";
     }
    } catch {
```

```
      Console.WriteLine("request error!");
    }
}

void response()
{
  try
  {
    Console.WriteLine("==========response()==========");
    String path = innerText(header, "GET ", "HTTP/").Trim(); // 取得檔案路徑：GET 版。
    HttpUtility.UrlDecode(path);
    String fullPath = root+path;
    FileInfo info = new FileInfo(fullPath);
    if (!info.Exists)
        throw new Exception("File not found !");
    send("HTTP/1.0 200 OK\n");
    send("Content-Type: "+type(fullPath)+"\n");
    send("Content-Length: "+info.Length+"\n");
    send("\n");
    byte[] buffer = new byte[4096];
    FileStream fileStream = File.OpenRead(fullPath);
```

```
    while (true)
    {
        int len = fileStream.Read(buffer, 0, buffer.Length);
        socket.Send(buffer, 0, len, SocketFlags.None);
        if (len < buffer.Length) break;
    }
    fileStream.Close();
  } catch {
    try {
     send("HTTP/1.0 404 Error\n");
     send("\n");
    } catch {
     Console.WriteLine("Send Error Msg fail!");
    }
  }
}

String type(String path)
{
    String type = "*/*";
    path = path.ToLower();
```

```
        for (int i = 0; i < map.Length; i++)
        {
            String[] tokens = map[i].Split('=');
            String ext = tokens[0], mime = tokens[1];
            if (path.EndsWith("." + ext)) type = mime;
        }
        return type;
    }
}
```

參考文獻

- How Java Web Servers Work - http://www.onjava.com/pub/a/onjava/2003/04/23/java_webserver.html
- JDK API : java.net.ServerSocket - http://java.sun.com/j2se/1.4.2/docs/api/java/net/ServerSocket.html
- Hypertext Transfer Protocol:HTTP/1.0 - http://www.w3.org/Protocols/HTTP/1.0/draft-ietf-http-spec.html

C# 網路程式：HTTP 程式設計（2）-- Client

HTTP Client 端程式簡介

在 C# 當中，HTTP Client 端元件的主要物件有 WebClient 與 Browser 元件，其中的 WebClient 物件是一個單純用來下載網頁的程式，這樣的物件可以用來撰寫像 WebCrawler 這樣的搜尋引擎關鍵程式，用來抓取全世界的網頁。

有時我們需要將整個瀏覽器嵌入到視窗程式中，此時就可以使用 Browser 元件。事實上、Visual C# 當中的 Browser 元件就是 Internet Explorer (IE)，因此您可以很容易的用 Visual C# 做出類似 IE 的功能，並且與其他視窗程式進行互動。

單一網頁下載

```
using System;
using System.IO;
using System.Net;
```

```
class PageDwnloader
{
    public static void Main(String[] args)
    {
        String text = fileToText(args[0]);
        String[] urls = text.Split('\n');
        for (int i = 0; i < urls.Length; i++)
        {
         Console.WriteLine(i + ":" + urls[i]);
         UrlToFile(urls[i], i+".htm");
        }
//   Console.WriteLine(text);
    }

    public static String fileToText(String filePath)
    {
        StreamReader file = new StreamReader(filePath);
        String text = file.ReadToEnd();
        file.Close();
        return text;
    }
```

```
    public static void UrlToFile(String url, String file) {
        WebClient webclient = new WebClient();
        webclient.DownloadFile("http://"+url, file);
    }
}
```

網路爬蟲 WebCrawler 的設計

搜尋引擎是網際網路興起後最常被使用的工具之一，其主要技術包含前端的全文檢索與後端的網頁蒐集兩類，本文將介紹搜尋引擎後端的網頁蒐集技術 - 并且以 C# 語言實作一個100行左右的網頁蒐集程式 "網路爬蟲 (Crawler)", 然後說明其製作方法與原理。

Google 等搜尋網站的背後，都有一個強大的網頁蒐集程式，可以將全世界的網頁通通抓回去儲存以便提供搜尋之用，這個程式就稱為 "爬行者 (Crawler)"，也有人索性稱為蜘蛛 (Spider)，因為這個就好像在網路上爬來爬去的蜘蛛一樣，到處抓網頁回家放。

Crawler 的設計原理，簡要來說是透過程式去追蹤網頁上的超連結，然後不斷往外擴張，以便將全世界中曾經被連結到的網頁全部都抓回到來，這也是 Google, Baidu, Bing, Yahoo 等網站背後最重要的程式之一。

要啟動 Crawler，首先要給一個起始點，以下的範例是利用台灣的 tw.msn.com 網站作為起始點，然後不斷透過「透過連結抓取網頁、取出連結再抓網頁」的方式擴展出去，以下是 Crawler 的核心程式。

每當抓到新的網頁時，我們繪將其存檔，然後抽取出其中所有的超連結，並將這些超連結放入等待抓取的網址庫中，以便下次可以抓取該網頁。

檔案：WebCrawler.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Text;
using System.IO;
using System.Net;
using System.Text.RegularExpressions;

class WebCrawler
{
//    WebProxy proxy = new WebProxy("http://proxy.internal:3128/", true);
    List<String> urlList = new List<String>();
    // Dictionary<String, String>

    public static void Main(String[] args)
    {
```

```
    WebCrawler crawler = new WebCrawler();
    crawler.urlList.Add("http://tw.msn.com/");
    crawler.craw();
}

public void craw()
{
    int urlIdx = 0;
    while (urlIdx < urlList.Count)
    {
        try
        {
            String url = urlList[urlIdx];
            String fileName = "data/" + toFileName(url);
            Console.WriteLine(urlIdx + ":url=" + url + " file=" + fileName);
            urlToFile(url, fileName);
            String html = fileToText(fileName);
            foreach (String childUrl in matches("\\shref\\s*=\\s*\"(.*?)\"", html, 1))
            {
                Console.WriteLine(childUrl);
                urlList.Add(childUrl);
```

```
            }
        }
        catch
        {
            Console.WriteLine("Error:" + urlList[urlIdx] + " fail!");
        }
        urlIdx++;
    }
}

public static IEnumerable matches(String pPattern, String pText, int pGroupId)
{
    Regex r = new Regex(pPattern, RegexOptions.IgnoreCase | RegexOptions.Compiled);
    for (Match m = r.Match(pText); m.Success; m = m.NextMatch())
     yield return m.Groups[pGroupId].Value;
}

public static String fileToText(String filePath)
{
    StreamReader file = new StreamReader(filePath);
    String text = file.ReadToEnd();
```

```
        file.Close();
        return text;
    }

    public void urlToFile(String url, String file)
    {
        WebClient webclient = new WebClient();
//        webclient.Proxy = proxy;
        webclient.DownloadFile(url, file);
    }

    public static String toFileName(String url)
    {
        String fileName = url.Replace('?', '_');
        fileName = fileName.Replace('/', '_');
        fileName = fileName.Replace('&', '_');
        fileName = fileName.Replace(':', '_');
        fileName = fileName.ToLower();
        if (!fileName.EndsWith(".htm") && !fileName.EndsWith(".html"))
         fileName = fileName + ".htm";
        return fileName;
```

```
    }
}
```

以上的 Crawler 是搜尋引擎中的關鍵技術，在本文中我們實作了一個簡單的 Crawler ，這個程式可以用來作為個人抓取網頁個工具程式，作為建立搜尋引擎的基礎。

Browser 的控制

教學錄影： * C# 瀏覽器控制 1/3 -- http://youtu.be/CIwYabPN7qA * C# 瀏覽器控制 2/3 -- http://youtu.be/sJ6cfuL3-ZA * C# 瀏覽器控制 3/3 -- http://youtu.be/YThlDxk-E7U

專案程式下載：https://dl.dropbox.com/u/101584453/cs/code/WebBrowser.zip

在 C# 當中控制 Internet Explorer (IE) 瀏覽器是一件很簡單的事情，因為 .NET framework 當中已經將 IE 的 WebBrowser 內建成一個控制元件，只要利用這個控制元件中的網址 (Url) 欄位，以及瀏覽 Navigate(url)、向前 GoForward()、向後 GoBack() 等函數，就可以輕鬆的控制瀏覽器元件的行為了。

檔案：WebBrowser.cs

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

```
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void buttonPrev_Click(object sender, EventArgs e)
        {
            webBrowser.GoBack();
        }

        private void buttonNext_Click(object sender, EventArgs e)
        {
            webBrowser.GoForward();
        }

        private void updateState()
        {
```

```
    buttonPrev.Enabled = webBrowser.CanGoBack;
    buttonNext.Enabled = webBrowser.CanGoForward;
    if (webBrowser.Url != null)
        comboBoxUrl.Text = webBrowser.Url.ToString();
}

private void buttonGo_Click(object sender, EventArgs e)
{
    webBrowser.Navigate(comboBoxUrl.Text);
}

private void webBrowser_Navigated(object sender, WebBrowserNavigatedEventArgs e)
{
    updateState();
}

private void comboBoxUrl_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
        webBrowser.Navigate(comboBoxUrl.Text);
}
```

```
    }
}
```

您可以看到在 Visual C# 當中使用瀏覽器元件事非常容易的事情，這樣的功能可以讓您在應用程裏輕易的嵌入瀏覽器以便讓使用者看到某些網頁，讓 Web 程式的開發變得輕鬆且愉快。

C# 遊戲程式：XNA 遊戲引擎

XNA 遊戲程式架構

XNA 是微軟於 2007 年推出的遊戲設計引擎，以 C# 為主要語言。因此，您可以輕易的用 C# 在 Visual Studio 的 XNA 開發工具中撰寫 2D、3D 電腦遊戲。

XNA 所撰寫的遊戲可以在 MS. Windows 的電腦中，或者是 XBOX 遊戲機中使用，最近微軟更將 XNA 加入到 Window Phone 7 版中，讓手機也納入到 XNA 所支援的平台中。

大部分對 XNA 有興趣的人都會將焦點放在 3D 遊戲上，而這也正是 XNA 最強最好用的地方，用 XNA 寫 3D 遊戲其實相當簡單，其難度與 2D 遊戲差不多。

一個 XNA 的遊戲，是一個繼承 Microsoft.Xna.Framework.Game 這個類別的程式，其架構大致如下。

```
public class MyGame : Microsoft.Xna.Framework.Game
{
    GraphicsDeviceManager graphics; // 繪圖裝置管理器
    GamePadState previousState; // XBOX 的按鈕狀態
#if !XBOX
    KeyboardState previousKeyboardState;    // 個人電腦的鍵盤狀態。
```

```
#endif
        GameObject gameObj = null;  // 模型物件
        SoundEffect sound = null; // 音效物件
        ...

        public GameMain()
        {
            graphics = new GraphicsDeviceManager(this);
            Content.RootDirectory = "Content";
        }

        protected override void Initialize() // 初始化
        {
            camera = new Camera(this);
            this.Components.Add(camera);
            base.Initialize();
        }

        protected override void LoadContent()
        {
            gameObj.model = Content.Load<Model>("MyModel"); // 載入模型
```

```
    gameObj.scale = 0.1f; // 設定放大倍率
    gameObj.position = new Vector3(0.0f, 0.0f, -10.0f); // 設定起始位置
    ...
}

protected override void Update(GameTime gameTime)
{
  // 1. 偵測鍵盤或滑鼠，根據輸入決定主角如何移動。
    if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
        this.Exit();

    KeyboardState keyboardState = Keyboard.GetState();  //得到目前鍵盤每一個按鍵的狀況
    if (keyboardState.IsKeyDown(Keys.Space) && previousKeyboardState.IsKeyUp(Keys.Space))
    {
      // 當空白鍵被按下時，...
    }

  // 2. 移動物件
  gameObj.position += gameObj.velocity;

  // 3. 播放聲音
```

```
        if (...)  explodeSound.Play();

        // 4. 呼叫基礎類別的 Update() 函數。
        base.Update(gameTime);
    }

    protected override void Draw(GameTime gameTime)
    {
        GraphicsDevice.Clear(Color.CornflowerBlue);

        gameObj.Draw(camera);

        base.Draw(gameTime);
    }

    public void Draw(Model model)
    {
        foreach (ModelMesh mesh in model.Meshes)
        {
            foreach (BasicEffect effect in mesh.Effects)
            {
```

```
                effect.EnableDefaultLighting();
                effect.PreferPerPixelLighting = true;

                effect.World =
                    Matrix.CreateFromYawPitchRoll(
                    rotation.Y,
                    rotation.X,
                    rotation.Z) *

                    Matrix.CreateScale(scale) *

                    Matrix.CreateTranslation(position);

                effect.Projection = camera.projection;
                effect.View = camera.view;
            }
            mesh.Draw();
        }
    }
}
```

在以上的程式架構中，MyGame : Microsoft.Xna.Framework.Game 這行程式宣告 MyGame 繼承 Xna.Framework.Game 這個類別，這是所有 XNA 程式共同的起點。透過這樣的繼承關係，MyGame 就只要實作 XNA 遊戲需要實作的部份，其餘的邏輯就由 Xna.Framework 幫我們處理掉了，於是、我們就只要實作以下的幾個函數，就能讓 XNA 遊戲運作起來了。

```
protected override void Initialize() // 初始化
...

protected override void LoadContent() // 載入資源
...

protected override void Update(GameTime gameTime) // 更新畫面
...

protected override void Draw(GameTime gameTime)   // 繪製畫面
```

通常，在 XNA 遊戲程式中，我們會在 Initialize()　進行初始化動作 (物件建立、參數設定、....)，然後在 LoadContent() 函數當中載入資源 (像是 2D 的圖檔或 3D 的模型檔等)。然後、整個 XNA 程式就開始運作了，XNA 引擎會不斷的呼叫 Update 動作 (通常每秒六十次)，以變更新遊戲畫面，而我們所要作的工作，就只剩下決定要如何畫圖，以及鍵盤滑鼠被按下時應該作甚麼事就行了。

如果讀者想進一步理解 XNA 的 2D 或 3D 遊戲，以下是筆者用 C# 所撰寫的幾個 XNA 遊戲程式，請讀者直接參考這些內容學習 XNA 遊戲程式的寫法。

XNA 的 2D 打飛碟遊戲

專案下載： * 最簡單版本 -- https://dl.dropbox.com/u/101584453/cs/code/2D_FighterGame.zip * 有爆炸效果 -- https://dl.dropbox.com/u/101584453/cs/code/2D_FighterGameWithExplode.zip * 有爆炸聲音 -- https://dl.dropbox.com/u/101584453/cs/code/2D_FighterGameWithSound.zip

本節將介紹如何用微軟的 XNA 遊戲引擎設計一個 2D 的打飛碟遊戲。在該遊戲中，共有 3 隻飛碟會亂飛，玩家的符號是一個砲台，您必須發射砲彈打飛碟，並且一邊閃避飛碟，每打死一台飛碟就會加一分，一但被飛碟撞到後就會死亡，導致遊戲結束，最後看看你能得到多少分，下圖顯示了遊戲的一個擷取畫面。


BallGame
Score:7
Game Over!

設計方法：XNA 遊戲的主要架構包含載入 (LoadContent)、更新 (Update) 、繪出 (Draw) 等三個動作，以下是本遊戲中的主要程式片斷。

資料結構

```
public class Game1 : Microsoft.Xna.Framework.Game
{
    GraphicsDeviceManager graphics;
    SpriteBatch spriteBatch;
    Sprite playerSprite;
    const int MAX_MISSILES = 5, MAX_ENEMIES = 3;
    Sprite[] missileSprites = new Sprite[MAX_MISSILES];
    Sprite[] enemySprites = new Sprite[MAX_ENEMIES];
    Vector2 screenSize;
    Texture2D playerTexture, enemyTexture, missileTexture;
    KeyboardState previousKeyboardState = Keyboard.GetState();
    bool isGameOver = false;
    Random random = new Random();
    SpriteFont font;
    int score = 0;
```

載入資源（圖檔+字型）

```
protected override void LoadContent()
{
    // Create a new SpriteBatch, which can be used to draw textures.
    spriteBatch = new SpriteBatch(GraphicsDevice);

    // TODO: use this.Content to load your game content here
    screenSize = new Vector2(graphics.GraphicsDevice.Viewport.Width, graphics.GraphicsDevice.Viewport.H
    playerTexture = Content.Load<Texture2D>("player");
    enemyTexture = Content.Load<Texture2D>("enemy");
    missileTexture = Content.Load<Texture2D>("missile");

    playerSprite = new Sprite(playerTexture, new Vector2(300, 400));
    playerSprite.size.Y = 140;
    for (int i = 0; i < MAX_MISSILES; i++)
    {
        missileSprites[i] = new Sprite(missileTexture, new Vector2(0, 0));
        missileSprites[i].isAlive = false;
    }
    for (int i = 0; i < MAX_ENEMIES; i++)
```

```
        {
            enemySprites[i] = new Sprite(enemyTexture, randomEnemyXY());
            enemySprites[i].velocity = new Vector2(random.Next(-3, 3), random.Next(-3, 3));
            enemySprites[i].isAlive = false;
        }

        font = Content.Load<SpriteFont>("SpriteFont");
        base.LoadContent();
    }

    public Vector2 randomEnemyXY()
    {
      int enemyX = random.Next(0, (int) (screenSize.X-100) );
      int enemyY = random.Next(0);
      return new Vector2(enemyX, enemyY);
    }
```

更新邏輯模型

```
    protected override void Update(GameTime gameTime)
    {
        // Allows the game to exit
```

```
        if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
            this.Exit();

        // TODO: Add your update logic here
        if (isGameOver) return;
#if !XBOX
        KeyboardState keyboardState = Keyboard.GetState();
        if (keyboardState.IsKeyDown(Keys.Left))  playerSprite.velocity = new Vector2(-5, 0);
        if (keyboardState.IsKeyDown(Keys.Right)) playerSprite.velocity = new Vector2(5, 0);
        if (keyboardState.IsKeyDown(Keys.Up)) playerSprite.velocity = new Vector2(0, -5);
        if (keyboardState.IsKeyDown(Keys.Down)) playerSprite.velocity = new Vector2(0, 5);
        if (keyboardState.IsKeyDown(Keys.Space) && previousKeyboardState.IsKeyUp(Keys.Space))
        {
            float x = playerSprite.position.X + playerTexture.Width/2 - missileTexture.Width/2;
            float y = playerSprite.position.Y;
            fireMissile(x, y);
        }
        previousKeyboardState = keyboardState;
#endif
        Rectangle playerRect = new Rectangle((int)playerSprite.position.X + 20, (int)playerSprite.position.Y + 20,
                                             playerTexture.Width-40, playerTexture.Height/2);
```

```
        foreach (Sprite enemy in enemySprites)
        {
            if (!enemy.isAlive)
            {
                enemy.isAlive = true;
                enemy.position = randomEnemyXY();
            }

            enemy.Move(screenSize);
            Rectangle enemyRect = new Rectangle((int) enemy.position.X+20, (int) enemy.position.Y+20,
                                        enemyTexture.Width-40, enemyTexture.Height-40);
            if (enemyRect.Intersects(playerRect))
                isGameOver = true;

            foreach (Sprite missile in missileSprites)
            {
                if (missile.isInside(screenSize))
                    missile.Move(screenSize);
                else
                    missile.isAlive = false;
```

```
            if (missile.isAlive)
            {
                Rectangle missileRect = new Rectangle((int) missile.position.X, (int) missile.position.Y,
                                                      missileTexture.Width, missileTexture.Height);
                if (missileRect.Intersects(enemyRect))
                {
                    enemy.isAlive = false;
                    score ++;
                }
            }
        }
    }

    playerSprite.Move(screenSize);

    base.Update(gameTime);
}

public void fireMissile(float x, float y)
{
    foreach (Sprite missile in missileSprites)
```

```
        {
            if (!missile.isAlive)
            {
                missile.isAlive = true;
                missile.position = new Vector2(x, y);
                missile.velocity = new Vector2(0, -5);
                return;
            }
        }
    }
```

繪出 – 更新畫面

```
    protected override void Draw(GameTime gameTime)
    {
        GraphicsDevice.Clear(Color.CornflowerBlue);

        // TODO: Add your drawing code here
        spriteBatch.Begin(SpriteBlendMode.AlphaBlend);
        playerSprite.draw(spriteBatch);
        foreach (Sprite enemy in enemySprites)
            enemy.draw(spriteBatch);
```

```
        foreach (Sprite missile in missileSprites)
        {
            if (missile.isAlive)
                missile.draw(spriteBatch);
        }
        spriteBatch.DrawString(font, String.Format("Score:{0}", score), new Vector2(10, 10), Color.Black);
        if (isGameOver)
            spriteBatch.DrawString(font, "Game Over!", new Vector2(screenSize.X / 2, screenSize.Y / 2), Color.Bla
        spriteBatch.End();

        base.Draw(gameTime);
    }
}
```

透過以上程式，我們示範了如何使用 XNA 設計一個簡單的小遊戲『打飛碟』。透過這樣的範例，應該可以呈現出 XNA 的 2D 遊戲設計主要架構。

XNA 的 3D 遊戲架構

在 3D 的世界裏，每個物件都有其 3D 座標，而且最後還是必須要透過某個相機將整個視角畫面呈現在 2D 的螢幕上，因此 XNA 的 3D 遊戲設計牽涉到遊戲物件 (Game Object) 與照相機 (Camera) 等物體。

通常 XNA 程式裏的每個模型都會有對應的物件，如果把這些物件抽象化，那就會抽象出下列 GameObject 結構，於是我們就只要繼承這個物件，然後為每個物件的 model 填入適當的模型就行了。

```
// 遊戲模型物件
class GameObject
{
    public Model model = null;
    public Vector3 position = Vector3.Zero;
    public Vector3 rotation = Vector3.Zero;
    public float scale = 1.0f;
    public Vector3 velocity = Vector3.Zero;
    public bool alive = false;

    public BoundingSphere boundingSphere()
    {
        BoundingSphere sphere = model.Meshes[0].BoundingSphere;
        sphere.Center = position;
        sphere.Radius *= scale;
        return sphere;
    }
```

```
public void Draw(Camera camera)
{
    foreach (ModelMesh mesh in model.Meshes)
    {
        foreach (BasicEffect effect in mesh.Effects)
        {
            effect.EnableDefaultLighting();
            effect.PreferPerPixelLighting = true;

            effect.World =
                Matrix.CreateFromYawPitchRoll(
                rotation.Y,
                rotation.X,
                rotation.Z) *

                Matrix.CreateScale(scale) *

                Matrix.CreateTranslation(position);

            effect.Projection = camera.projection;
            effect.View = camera.view;
```

```
            }
            mesh.Draw();
        }
    }
}
```

在遊戲當中，除了模型物件之外，還有一個很特殊的物件，那就是像機，這個物件的邏輯較為獨特，幾乎有固定的寫法，以下是一個基本的像機物件之寫法。

```
// 相機物件
class Camera : Microsoft.Xna.Framework.GameComponent
{
    public Matrix view;        // 視覺矩陣
    public Matrix projection =        // 投影矩陣
            Matrix.CreatePerspectiveFieldOfView(
                    MathHelper.PiOver4, // 視角 45度
                    1.333f, // 螢幕 寬高比
                    1,     // 最近的Z軸截點
                    10000); // 最遠的Z軸截點

    public Vector3 FrontVector = new Vector3(0, 0, -1); //相機的 往前向量
```

```
public Vector3 Position = new Vector3(0.0f, 2.0f, 20.0f); // 相機的位置
public float Yaw = 0.0f, Pitch = 0.0f, Roll = 0.0f;
public float YawDelta = 0.2f, PitchDelta = 0.2f, RollDelta = 0.2f;
public float MoveDelta = 200;   // 前後走 的 增減量

public Camera(Game game)
    : base(game)
{
    // TODO: Construct any child components here

}

//更新 相機 的位置
public void UpdateCamera(Vector3 Position, float Yaw, float Pitch, float Roll)
{
    //攝影機一開始是 朝向 負 Z 軸 看著(0, 0, -1) 的
    Vector3 LookAt = Position + Vector3.TransformNormal(FrontVector,
                                Matrix.CreateFromYawPitchRoll(Yaw, Pitch, Roll));

    // 根據 相機的位置 相機要看到的點 相機上方的向量 得到 視覺矩陣
    view = Matrix.CreateLookAt(Position,  // 相機的位置
```

```
                        LookAt,    // 相機要看到的點
                        Vector3.Up); // 相機上方的向量
    }

    public override void Initialize()
    {
        base.Initialize();
    }

    public override void Update(GameTime gameTime)
    {
        // TODO: Add your update code here
        float elapsedTime = (float) gameTime.ElapsedGameTime.TotalSeconds;

        KeyboardState newState;  // 宣告一個KeyboardState 結構的變數
        newState = Keyboard.GetState();  //得到目前鍵盤每一個按鍵的狀況
        if (newState.IsKeyDown(Keys.Right)) Yaw -= YawDelta * elapsedTime; //右鍵按下
        if (newState.IsKeyDown(Keys.Left)) Yaw += YawDelta * elapsedTime;  //左鍵按下
        if (newState.IsKeyDown(Keys.Up)) Pitch += PitchDelta * elapsedTime;
        if (newState.IsKeyDown(Keys.Down)) Pitch -= PitchDelta * elapsedTime;
        UpdateCamera(Position, Yaw, Pitch, Roll);
```

```
            base.Update(gameTime);
        }
    }
```

XNA 的 3D 打飛碟遊戲

- 最簡單版本 -- https://dl.dropbox.com/u/101584453/cs/code/3D_FighterGameMissileSound.zip
- 有碰撞偵測 -- https://dl.dropbox.com/u/101584453/cs/code/3D_FighterGameWithCollision.zip
- 有爆炸效果 -- https://dl.dropbox.com/u/101584453/cs/code/3D_FighterGameParticle.zip

程式碼：

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
```

```
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Net;
using Microsoft.Xna.Framework.Storage;
using System.Diagnostics;


namespace _3D_FighterGame
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        static void Main(string[] args)
        {
            using (Game1 game = new Game1())
            {
                game.Run();
            }
        }
```

```
    }

    /// <summary>
    /// This is the main type for your game
    /// </summary>
    public class Game1 : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;
        GameObject launcherBase = new GameObject();
        GameObject launcherHead = new GameObject();
        Model enemyModel = null, missileModel = null;
        List<GameObject> enemies = new List<GameObject>();
        List<GameObject> missiles = new List<GameObject>();
        Camera camera = null;
        Random random = new Random(7);
        SoundEffect explodeSound = null, fireMissileSound = null;

        GamePadState previousState;
#if !XBOX
        KeyboardState previousKeyboardState;
```

```
#endif

        public Game1()
        {
            graphics = new GraphicsDeviceManager(this);
            Content.RootDirectory = "Content";
        }

        /// <summary>
        /// Allows the game to perform any initialization it needs to before starting to run.
        /// This is where it can query for any required services and load any non-graphic
        /// related content.  Calling base.Initialize will enumerate through any components
        /// and initialize them as well.
        /// </summary>
        protected override void Initialize()
        {
            // TODO: Add your initialization logic here
            camera = new Camera(this);
            this.Components.Add(camera);

            base.Initialize();
```

```
        }

        /// <summary>
        /// LoadContent will be called once per game and is the place to load
        /// all of your content.
        /// </summary>
        protected override void LoadContent()
        {
            // Create a new SpriteBatch, which can be used to draw textures.
            spriteBatch = new SpriteBatch(GraphicsDevice);

            // TODO: use this.Content to load your game content here
            camera = new Camera(this);

            launcherBase.model = Content.Load<Model>("Models\\launcher_base");
            launcherBase.scale = 0.1f;
            launcherBase.position = new Vector3(0.0f, 0.0f, -10.0f);

//            launcherHead.model = Content.Load<Model>("Models\\launcher_head");
//            launcherHead.scale = 0.2f;
//            launcherHead.position = launcherBase.position + new Vector3(0.0f, 20.0f, -120.0f);
```

```
    enemyModel = Content.Load<Model>("Models\\enemy");
    missileModel = Content.Load<Model>("Models\\missile");
    explodeSound = Content.Load<SoundEffect>("Sound\\explosion");
    fireMissileSound = Content.Load<SoundEffect>("Sound\\missilelaunch");
}

/// <summary>
/// UnloadContent will be called once per game and is the place to unload
/// all content.
/// </summary>
protected override void UnloadContent()
{
    // TODO: Unload any non ContentManager content here
}

protected void newEnemy()
{
    GameObject e = new GameObject();
    e.model = enemyModel;
    e.position = new Vector3(random.Next(-5, 5) * 50.0f, random.Next(-5, 5) * 50.0f, -2000.0f);
```

```
        e.velocity = new Vector3(0, 0, 2.0f);
        e.scale = 0.03f;
        enemies.Add(e);
    }

    protected void newMissile()
    {
        Matrix rotationMatrix = Matrix.CreateFromYawPitchRoll(camera.Yaw, camera.Pitch, camera.Roll);

        GameObject m = new GameObject();
        m.model = missileModel;
        m.position = launcherBase.position + Vector3.Transform(Vector3.Forward * 10, rotationMatrix);
        m.scale = 10.0f;
        // m.velocity = new Vector3(0, 0, -10.0f);
        m.velocity = Vector3.Transform(Vector3.Forward * 10, rotationMatrix);
        missiles.Add(m);
        fireMissileSound.Play();
    }

    protected void getLuncherBasePosition()
    {
```

```
            Matrix rotationMatrix = Matrix.CreateFromYawPitchRoll(camera.Yaw, camera.Pitch, camera.Roll);
            launcherBase.position = camera.Position + Vector3.Transform(Vector3.Forward * 50, rotationMatrix); //  +
            launcherBase.rotation.Y = camera.Yaw;
            launcherBase.rotation.X = camera.Pitch;
            launcherBase.rotation.Z = camera.Roll;
//            Trace.WriteLine(launcherBase.position);
        }

        double lastEnemyDelay = 0.0f;
        /// <summary>
        /// Allows the game to run logic such as updating the world,
        /// checking for collisions, gathering input, and playing audio.
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing values.</param>
        protected override void Update(GameTime gameTime)
        {
            // Allows the game to exit
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
                this.Exit();

            getLuncherBasePosition();
```

```
KeyboardState keyboardState = Keyboard.GetState();  //得到目前鍵盤每一個按鍵的狀況
if (keyboardState.IsKeyDown(Keys.Space) && previousKeyboardState.IsKeyUp(Keys.Space))
{
    newMissile();
}

double elapsedTime = gameTime.ElapsedGameTime.TotalSeconds;
lastEnemyDelay += elapsedTime;
if (lastEnemyDelay >= 1)
{
    if (enemies.Count < 10)
    {
        newEnemy();
        lastEnemyDelay = 0.0f;
    }
}

GameObject[] eList = enemies.ToArray();
foreach (GameObject e in eList)
{
```

```
        if (e.position.Z >= 0)
            enemies.Remove(e);
        else
            e.position += e.velocity;
    }

    GameObject[] mList = missiles.ToArray();
    foreach (GameObject m in mList)
    {
        if (m.position.Z <= -3000)
            missiles.Remove(m);
        else
            m.position += m.velocity;
    }

    // TODO: Add your update logic here
    camera.Update(gameTime);

    previousKeyboardState = keyboardState;

    base.Update(gameTime);
```

```
        }

        /// <summary>
        /// This is called when the game should draw itself.
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing values.</param>
        protected override void Draw(GameTime gameTime)
        {
            GraphicsDevice.Clear(Color.CornflowerBlue);
            // TODO: Add your drawing code here
            DrawGameObject(launcherBase);
//          DrawGameObject(launcherHead);
            foreach (GameObject e in enemies)
                DrawGameObject(e);
            foreach (GameObject m in missiles)
                DrawGameObject(m);

            base.Draw(gameTime);
        }

        void DrawGameObject(GameObject gameobject)
```

```
{
    foreach (ModelMesh mesh in gameobject.model.Meshes)
    {
        foreach (BasicEffect effect in mesh.Effects)
        {
            effect.EnableDefaultLighting();
            effect.PreferPerPixelLighting = true;

            effect.World =
                Matrix.CreateFromYawPitchRoll(
                gameobject.rotation.Y,
                gameobject.rotation.X,
                gameobject.rotation.Z) *

                Matrix.CreateScale(gameobject.scale) *

                Matrix.CreateTranslation(gameobject.position);

            effect.Projection = camera.projection;
            effect.View = camera.view;
        }
```

```
            mesh.Draw();
        }
    }
}

class GameObject
{
    public Model model = null;
    public Vector3 position = Vector3.Zero;
    public Vector3 rotation = Vector3.Zero;
    public float scale = 1.0f;
    public Vector3 velocity = Vector3.Zero;
    public bool alive = false;
}

/// <summary>
/// This is a game component that implements IUpdateable.
/// </summary>
public class GameComponent_Grid : Microsoft.Xna.Framework.DrawableGameComponent
{
    public int gridSize = 12; // 每邊有 幾格
```

```
public float gridScale = 1.0f; // 每格 的 寬
public Color gridColor = new Color(0xFF, 0xFF, 0xFF, 0xFF);   // 格線 的顏色 內定是黑色

VertexBuffer vertexBuffer; // 頂點緩衝區
VertexDeclaration vertexDeclaration; // 頂點格式 (每個頂點 包含什麼內容)
BasicEffect effect; // 產出時會用到的 效果
int vertexCount;  // 頂點 數目
GraphicsDevice device; //繪圖設備

public Matrix world = Matrix.Identity; // 世界 觀測 投影 矩陣

public Matrix view = Matrix.CreateLookAt(new Vector3(0.0f, 20.0f, 20.0f),
                                         Vector3.Zero,
                                         Vector3.Up);

public Matrix projection = Matrix.CreatePerspectiveFieldOfView(
                                        MathHelper.ToRadians(45.0f),
                                        1.333f, 1.0f, 10000.0f);

public GameComponent_Grid(Game game)
    : base(game)
```

```
{
    // TODO: Construct any child components here
    this.device = game.GraphicsDevice;
}

/// <summary>
/// Allows the game component to perform any initialization it needs to before starting
/// to run.  This is where it can query for any required services and load content.
/// </summary>
public override void Initialize()
{
    // TODO: Add your initialization code here
    effect = new BasicEffect(device, null); // 效果

    vertexCount = (gridSize + 1) * 4; // 每邊的頂點數 比 每邊的格數 多一，共有四個邊

    // 每個頂點 包含 位置 和 顏色 ，先空出 vertexCount 個 頂點
    VertexPositionColor[] vertices = new VertexPositionColor[vertexCount];

    float length = (float)gridSize * gridScale; // 邊長 等於 格數 乘以 格寬
    float halfLength = length * 0.5f;  // 半邊長 因為是要 以原點為中心 左右 各半
```

```
int index = 0; // 頂點 索引

// 定義頂點位置  頂都是 躺在 XZ 平面上
for (int i = 0; i <= gridSize; ++i)
{
    vertices[index++] = new VertexPositionColor(
        new Vector3(-halfLength, 0.0f, i * gridScale - halfLength),
        gridColor); // 左邊的頂點

    vertices[index++] = new VertexPositionColor(
        new Vector3(halfLength, 0.0f, i * gridScale - halfLength),
        gridColor); // 右邊的頂點

    vertices[index++] = new VertexPositionColor(
        new Vector3(i * gridScale - halfLength, 0.0f, -halfLength),
        gridColor); // 上緣的頂點

    vertices[index++] = new VertexPositionColor(
        new Vector3(i * gridScale - halfLength, 0.0f, halfLength),
        gridColor); // 下緣的頂點
```

```
        }

        // 建立 頂點緩衝區
        vertexBuffer = new VertexBuffer(device, vertexCount *  VertexPositionColor.SizeInBytes,
                                        BufferUsage.WriteOnly);
        // 將 頂點資料 複製入 頂點緩衝區 內
        vertexBuffer.SetData<VertexPositionColor>(vertices);

        // 頂點格式
        vertexDeclaration = new VertexDeclaration(device, VertexPositionColor.VertexElements);

        base.Initialize();
    }

    /// <summary>
    /// Allows the game component to update itself.
    /// </summary>
    /// <param name="gameTime">Provides a snapshot of timing values.</param>
    public override void Update(GameTime gameTime)
    {
        // TODO: Add your update code here
```

```
            base.Update(gameTime);
        }

        public override void Draw(GameTime gameTime)
        {

            // 效果 三大矩陣 設定
//              effect.World = world;
//              effect.View = view;
//              effect.Projection = projection;

            effect.VertexColorEnabled = true;  // 使用 頂點顏色 效果

            device.VertexDeclaration = vertexDeclaration; // 頂點格式
            device.Vertices[0].SetSource(vertexBuffer, 0, VertexPositionColor.SizeInBytes); // 頂點來源

            effect.Begin(); // 效果 開始

            foreach (EffectPass CurrentPass in effect.CurrentTechnique.Passes)
            {
```

```
            CurrentPass.Begin();
            device.DrawPrimitives(PrimitiveType.LineList, 0, vertexCount / 2); // 兩兩畫線 所以只有 vertexCount/
            CurrentPass.End();
        }
        effect.End();
    }
  }
}
```

從 XNA 到 Unity3D

雖然 XNA 已經是成熟的技術，但是卻不支援像 Window Form 或 WPF 這樣所示即所得的開發方式，還好現在已經有一款稱為 Unity 遊戲開發軟體，可以讓您用所視即所得的方式開發 3D 的遊戲程式。Unity 是一款強大的工業級跨平台遊戲引擎 （Win PC, XBox, PS, Wii，Android.....)，具有很棒的視覺化開發環境，並且支援 C# 與 JavaScript 等開發語言，並且有免費版可以下載。

- Unity 官網 -- http://unity3d.com/

目前也已經有一些中文書介紹 Unity 的程式設計，以下是這些書的網址：

- 全民做遊戲-Unity跨平台遊戲開發寶典 -- http://www.books.com.tw/exep/prod/booksfile.php?item=0010537427
- Unity 3D遊戲開發設計實務 -- http://www.books.com.tw/exep/prod/booksfile.php?item=0010551893

- Unity 3D遊戲開發設計學院 -- http://www.books.com.tw/exep/prod/booksfile.php?item=0010464868
- 遊戲工作室指導手冊：快速學會四種Unity遊戲設計(附光碟) -- http://www.books.com.tw/exep/prod/booksfile.php?item=0010570704

參考文獻

- 二維動畫圖 (Sprites) -- http://www.sprites-inc.co.uk/files/X/
- 聲音效果 -- http://www.ilovewavs.com/Effects/Music/Music.htm
- 聲音效果 -- http://www.ilovewaves.com/
- 網路資源
- XNA 官方網站 - http://creators.xna.com/
    - 包含 Getting Start, Tutorial, Samples, Starter Kits, Articles.
- Stormcode 的 XNA BLOG -- http://www.stromcode.com/category/xna/
- XNA+OGRE 的 BLOG -- http://blog.yam.com/xnaOgre
- Reimer's XNA Tutorial -- http://www.riemers.net/eng/Tutorials/XNA/Csharp/series1.php
- XNA 的粒子效果範例 -- http://creators.xna.com/en-US/sample/particle3d
- 如何將 Blender 模型匯出給 XNA 使用 -- http://forums.xna.com/forums/t/4614.aspx
- 如何將 Blender 的動作匯出給 XNA 使用 -- http://www.stromcode.com/2008/03/16/modeling-for-xna-with-blender-part-iv/
- Papervision3D - 五分鐘上手 (Flash 的 3D 遊戲引擎） -- http://blog.ring.idv.tw/comment.ser?i=194
- 2D/3D 遊戲程式設計入門ㄧ使用XNA3.0與C#(附光碟), 作者：鄞永傳、何振揚, 出版社：文魁, 出版

日期：2009年02月06日, 語言：繁體中文 ISBN：9789866482120
    - http://www.books.com.tw/exep/prod/booksfile.php?item=0010426322
- XNA Creator Club Online, Getting Start -- http://creators.xna.com/en-US/education/gettingstarted

C# 程式設計：結語

雖然我們已經介紹了有關 C# 的基本語法、命令列程式、視窗程式、Thread、網路程式、甚至是 XNA 遊戲程式等等，但仍然無法完整的涵蓋 C# 的所有面向，像是 ASP.NET 的網頁程式，Silverlight (類似 Flash) 的網頁應用、以及新一代的視窗 WPF 等等，都是我們沒有介紹到的主題，對於這些領域，只好請讀者自行查看相關的書籍。

事實上、C# 可以說是微軟的主力語言，因此所有的微軟技術幾乎都可以用 C# 進行存取與使用，這讓 C# 在微軟平台上幾乎可以做所有的事情，特別是在 .NET 平台上。而且 C# 也可以進行微軟系統呼叫，因此從 C# 出發幾乎可以深入整個微軟的 Windows 作業系統，這是本書無法更深入探討的主題。

但本書已經盡可能的涵蓋了筆者認為重要的部份，未來、如果有任何新的主題，筆者隨時會在補上，希望這樣的一本書能對讀者有所幫助。

附錄：教學錄影

主題	教學影片
第 1 章. 簡介與開發環境	
─ 安裝與課程規定	http://youtu.be/MtCx-T71_bg
─ 開發環境使用	http://youtu.be/_9rEEWE3lXQ
─ 控制項巡禮	http://youtu.be/e5366D_ngAA
─ 命令列 Hello 程式	http://youtu.be/WcN_Rzw1eac
─ 命令列編譯器 csc	http://youtu.be/biz51oMWRKk
第 2 章. 變數與運算式	
─ 基本型態宣告與觀察	http://youtu.be/T81pMd32tMY
─ 基本運算操作	http://youtu.be/yMOJjz7VmaE

─ 運算式家庭作業	http://youtu.be/JTUTdZjLBxI
第 3 章. 流程控制	
─ if 語法	http://youtu.be/qM1DpkVqEkQ
─ for, while 迴圈	http://youtu.be/Wk-RZvjgG2o
第 4 章. 陣列	
─ 陣列與迴圈	http://youtu.be/CoduSbHPKME
第 5 章. 函數	
- 函數的呼叫	http://youtu.be/dnkKId0ZO1c
第 6 章. 物件導向	
─ 封裝	http://youtu.be/HLNAbXHSQHs
─ 建構函數	http://youtu.be/HLNAbXHSQHs

─ 繼承 1	http://youtu.be/pNkvMXp28AE
─ 繼承 2	http://youtu.be/GKxxE3UdGms
─ 多型	http://youtu.be/G1QQyXEpSCk
─ 件導向練習題講解	http://youtu.be/Om50P0T_aVU
第 7 章. 例外處理	
─ 錯誤處理	http://youtu.be/7bCefu8BB00
第 8 章. 檔案處理	
─ 檔案處理	http://youtu.be/3EyPcAddd70
第 9 章. 資料結構	
─ Array 與 List 物件的使用	http://youtu.be/EHSGtKpRprI
─ HashTable 與 Dictionary 物件的使用	http://youtu.be/hYH-npRmmKM
第 10 章. 正規表達式	

第 11 章. 作業系統與 Thread	
─ 線程與死結	http://youtu.be/O1b31Cc9bfw
第 12 章. 視窗程式簡介	
─ 按鈕範例與視窗專案的結構	http://youtu.be/Vhk1SsZTTVk
─ 視窗程式設計簡介1	http://youtu.be/MtCx-T71_bg
─ 視窗程式設計簡介2	http://youtu.be/_9rEEWE3lXQ
─ 視窗控制項巡禮	http://youtu.be/e5366D_ngAA
─ 文字型計算機	http://youtu.be/bJ-fANUuRcU
─ 字典查詢程式	http://youtu.be/yS3G-H_hrFU
─ 將小字典擴充為小翻譯系統	http://youtu.be/_sFsTo41PXs
第 13 章. 時間驅動	

─ 時間驅動的邏輯	http://youtu.be/X0m8heWQrvM
─ 設計短跑用碼表	http://youtu.be/BAT8UsY__mU
─ Timer 與碼錶	http://youtu.be/UA0rizekLow
─ Timer 與小時鐘	http://youtu.be/NJ6B5-vO_88
─ Timer 與移動球	http://youtu.be/6Gs4MPzt6q4
第 14 章. 設計文字編輯器	
─ 文字編輯器	http://youtu.be/xz5sKvZjLZI
第 15 章. 畫圖功能	
─ 畫圖功能示範	http://youtu.be/8aAql8R4WSg
─ 小畫板	http://youtu.be/iD044JD3BI4
─ 繪圖並存檔	http://youtu.be/wjpHUdPqdwc
─ 抓取螢幕畫面	http://youtu.be/7qxv64yZCyE

─ 電子白板的設計	http://youtu.be/yqPR9IYD_6s
第 16 章. DataGridView 與 ListView 元件	
─ DataGridView 元件的使用	http://youtu.be/XzsZRqLAi30
─ 製作賣紅茶的 POS 系統	http://youtu.be/XX0wHhvkkiE
第 17 章. 網路程式設計	
─ 網路 UDP 程式設計	http://youtu.be/DxdfeURkNzU
第 18 章. UDP 程式設計	
─ 網路 UDP 程式設計	http://youtu.be/DxdfeURkNzU
第 19 章. TCP 程式設計	
─ 單向 TCP 訊息傳遞程式	http://youtu.be/Zc1NjR2v-5k
─ 雙向 P2P 命令列聊天室	http://youtu.be/W5HG9DtC6EI

─ 多人視窗版聊天室	http://youtu.be/lgojk1BOMsk
─ 多人視窗版聊天室(程式碼解析)	http://youtu.be/xRzh0yZ3rqw
第 20 章. HTTP 程式設計 (1) – Server	
─ HTTP 協定與 HelloServer 的實作	http://youtu.be/0pniFEvdjcc
─ 傳回超連結的 HelloServer	http://youtu.be/KGbZ6nQrATY
─ 設計簡易的 WebServer	http://youtu.be/QoqpaBAxPwQ
第 21 章. HTTP 程式設計 (2) – Client	
─ 單一網頁下載 (UrlDownloader)	http://youtu.be/eICPT158cxc
─ Crawler 網路爬蟲	http://youtu.be/Ytl19GjnhMo
─ 瀏覽器元件的控制	http://youtu.be/3mRW1_Usv6k
第 22 章. XNA 遊戲引擎	
─ 打飛碟遊戲	http://youtu.be/DMPl0x__dMg

第 23 章. 結語	