Chapter 1  
歡迎來到C#的世界

C#最吸引人的一點，莫過於親切直覺的開發環境了，用Visual Studio來開發程式時，大多數時候我們都身處在整合編輯器中，在這個編輯器裡，我們有各種方便的控制面板和強大的功能，省去了背各種指令的麻煩。

所以學習C#的第一步，就是要先熟悉Visual Studio的各項操作。

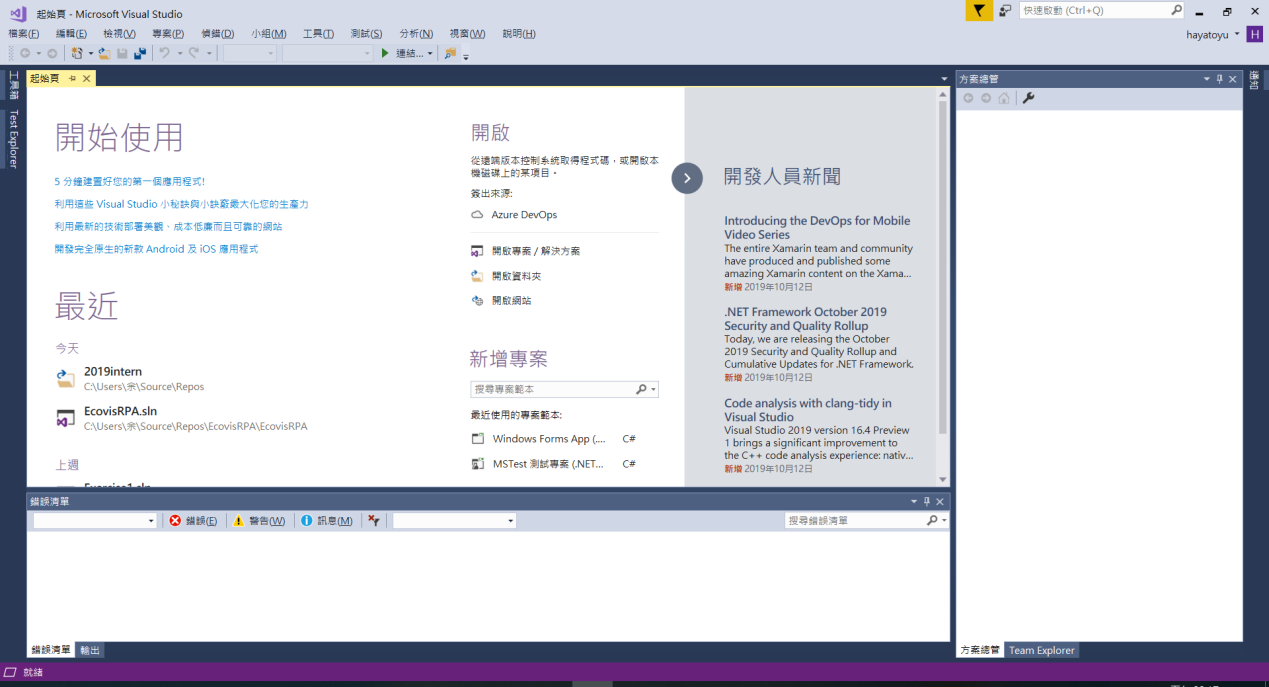
##### 完成本章後，您可以學會:

* 從零開始建置並了解一個主控台應用程式專案。
* 熟悉Visual Studio的操作介面及功能。
* 做出第一個Hello World程式。

## 1.1 新增專案

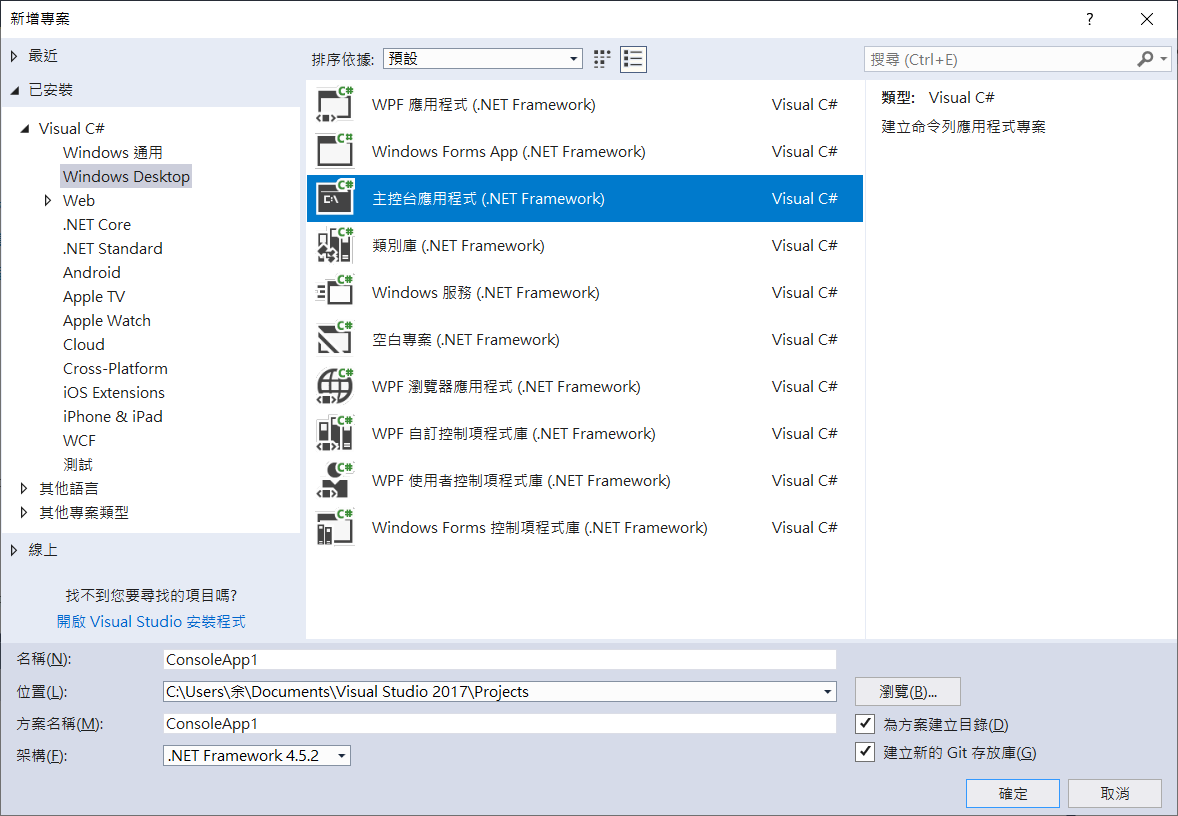
當您第一次啟動Visual Studio時，會來到起始頁。在此會列出您最近使用過的專案，以及新增、開啟等等功能。

在左上角的檔案選單中，能找到更多的功能，讓我們從這裡開始。請先點選左上角的檔案->新增->專案



|  |  |
| --- | --- |
| paper&pencil_48 | 由於不同版本的Visual Studio版面不盡相同，光看圖可能會對不上，記得一定要看文字敘述。基本上即使圖形不同，文字敘述也會是相同的。筆者所使用的為Visual Studio 2017。 |

點選完後會看到Visual Studio跳出選單，讓您選擇要新增哪種類型的專案。在此我們先選擇Visual C# -> Windows Desktop -> 主控台應用程式。這是最基本的專案類型，沒有任何圖形化介面，依靠命令提示字元作為人機互動的輸入與輸出。下方的名稱可自行修改。



|  |  |
| --- | --- |
| paper&pencil_48 | 如果您點選新增專案後，卻發現左邊已安裝中的專案類型及個數與筆者不相同，請回想在Chapter 0的安裝過程中，我們提過可以自由地選擇安裝的套件。如果您僅安裝.Net桌面開發，自然是不會看到其他的套件的。 |

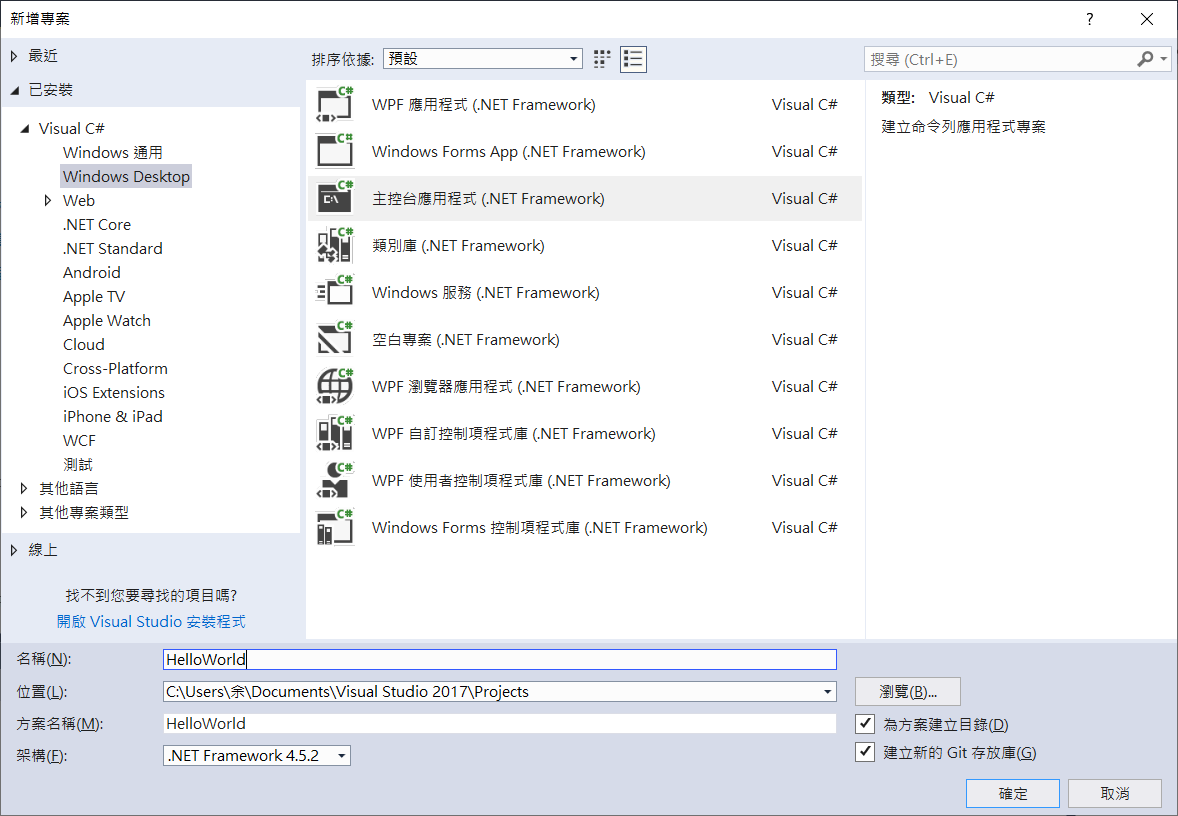
|  |  |
| --- | --- |
| paper&pencil_48 | 很多人不明白何謂主控台應用程式，事實上它的介面就是一個命令提示字元，在Windows 95時期有人會稱之為MS-DOS，您可以點您電腦上的開始 -> 輸入cmd，就可以叫出一個命令提示字元視窗。在Unix、Linux及OSX中，也有人稱之為終端機(terminal)或Shell。建立一個主控台應用程式，表示我們的程式執行起來，就會像下圖一樣，使用者可透過命令提示字元輸入資料，而程式執行的結果也會輸出在該命令提示字元中。 |

|  |  |
| --- | --- |
| warning_48 | 這裡提醒各位初學者們，雖然.Net本身對中文的支援非常好，但一般在命名專案的時候不會使用中文，還是請各位盡量以英文命名專案，以省去不必要的麻煩。 |

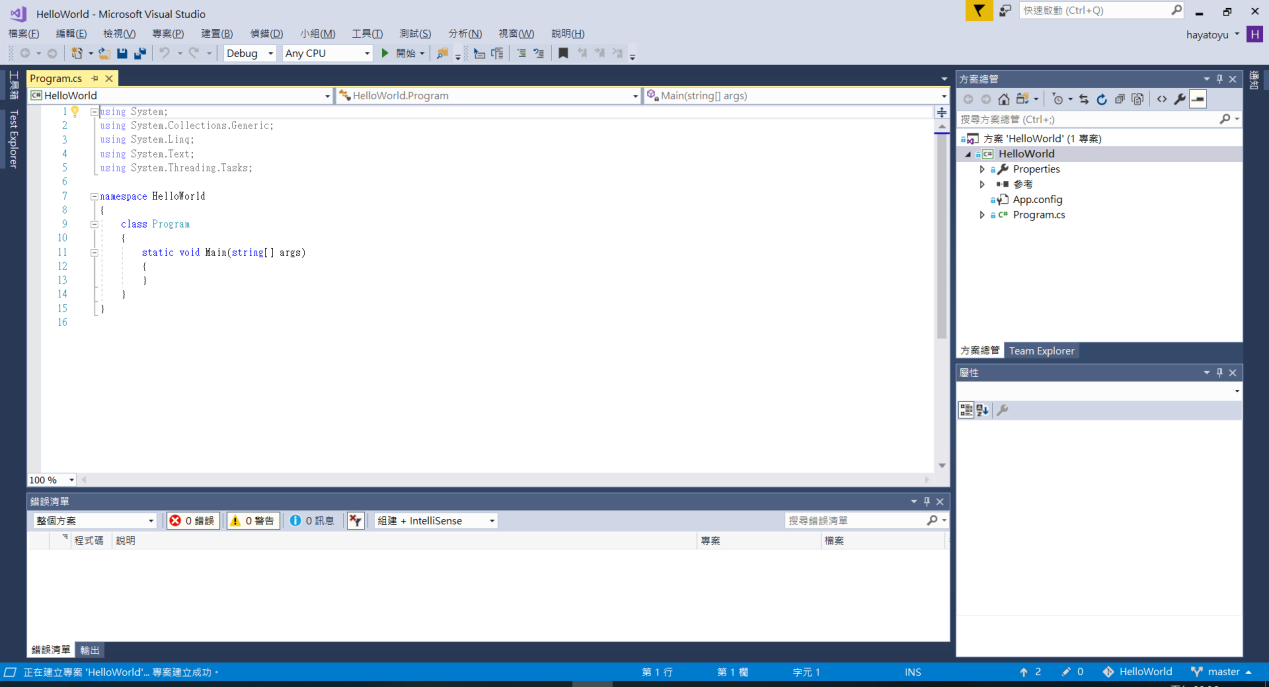
## 1.2 大名鼎鼎的Hello World – 第一個主控台應用程式

在程式設計師的行話間，Hello World可謂無人不曉，就讓我們從這裡來開始我們的旅程。

我們可將剛剛新增專案視窗下方的名稱改為Hello World，並點選確定。



點選確定之後，Visual Studio會自動建立專案，並將之載入。現在，我們可以看到如下的畫面：

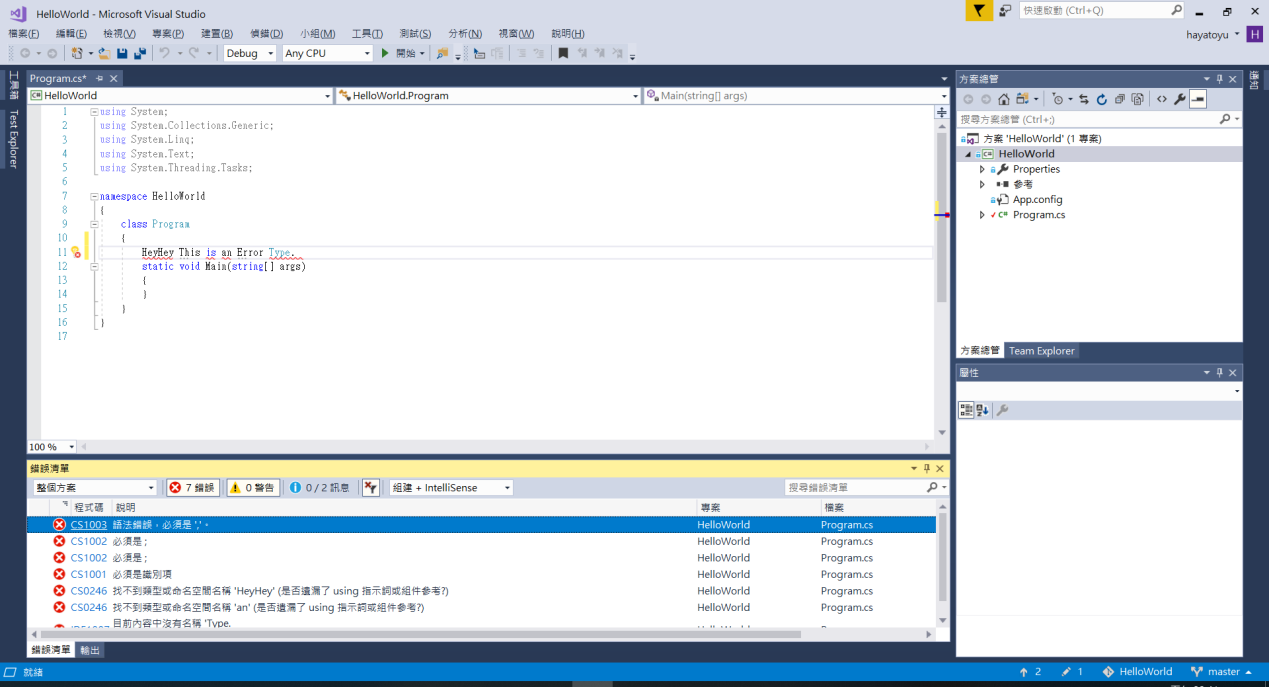


### 1.2.1 Visual Studio編輯器介面

這邊先簡單的介紹一下編輯器的介面。

左上角的一大塊區域，是程式碼的編輯區域，並且會依照不同的cs檔案做頁籤。例如我們現在看到的就是Program.cs檔案中自動產生的程式碼。

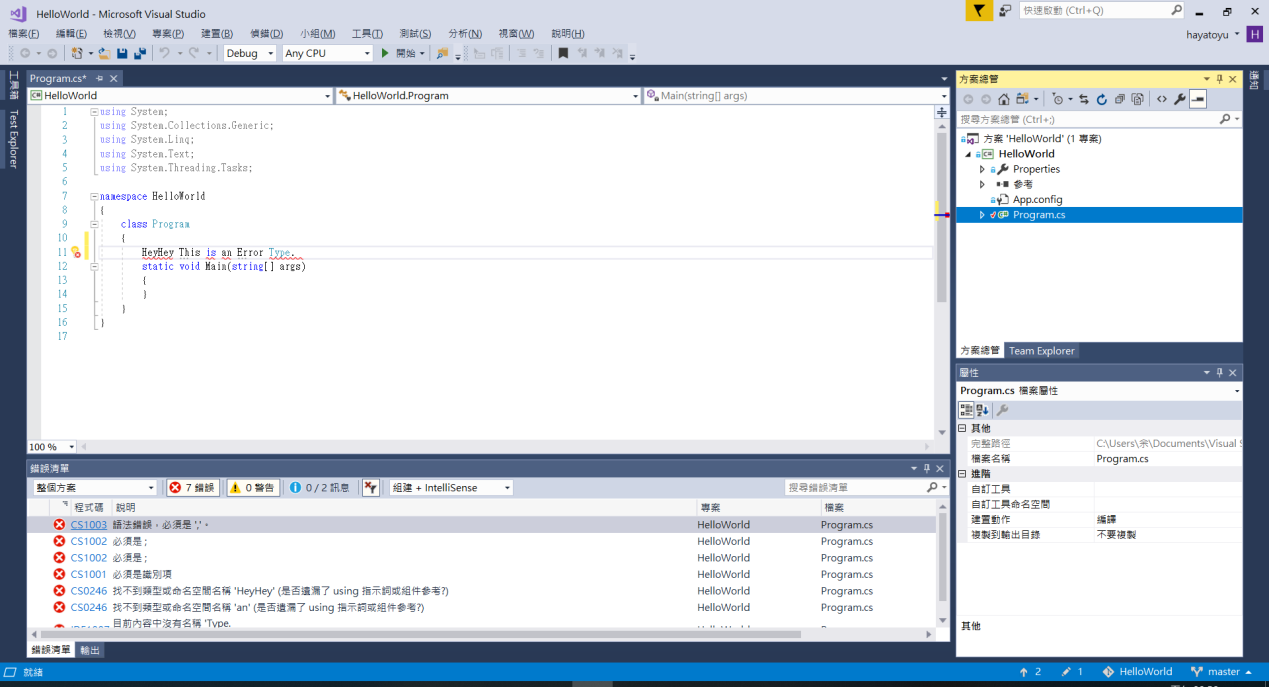
左下角長型區域，是錯誤清單。當程式碼編寫有誤時，便會在此輸出訊息。使用者可以藉著雙擊此處的錯誤訊息，快速移至出錯的程式碼以進行修正。我們不妨嘗試一下：



當我們在編輯器中編輯錯誤的程式碼時，在左上角的編輯器區塊，會以紅色底線描繪出錯誤的程式碼。而下方的錯誤清單，也會列出錯誤並提供可能的修正建議。

右上方的方案總管，列出本專案中的所有檔案，包含程式碼檔、設定檔及參考。如果您希望在這個專案中新增類別或參考，都是在此新增。本書包含自訂類別的章節，在該章節中會提到如何新增自訂類別。

而右下角的屬性視窗，則可進行對專案中各檔案的設定變更。我們可以試著點選方案總管中的Program.cs檔案，便可以看到下方的屬性視窗列出Program.cs檔案的各項屬性。但此處請不要去變更它的屬性。



|  |  |
| --- | --- |
| warning_48 | C#是編譯式語言，在執行前會先將程式碼透過編譯器編譯成可執行檔案(.exe)，如果程式碼在編寫時有誤，編譯器便可以在執行前將之揪出。然而，也有的情況是只有在執行期才會發生的錯誤(Runtime Error)，即使通過了編譯，也不代表程式必定可以順利執行。 |

|  |  |
| --- | --- |
| paper&pencil_48 | 錯誤清單區塊在編譯期僅列出錯誤訊息，在執行期還可多出動態監看式、呼叫堆疊、即時運算視窗等功能，是偵錯的好幫手。 |

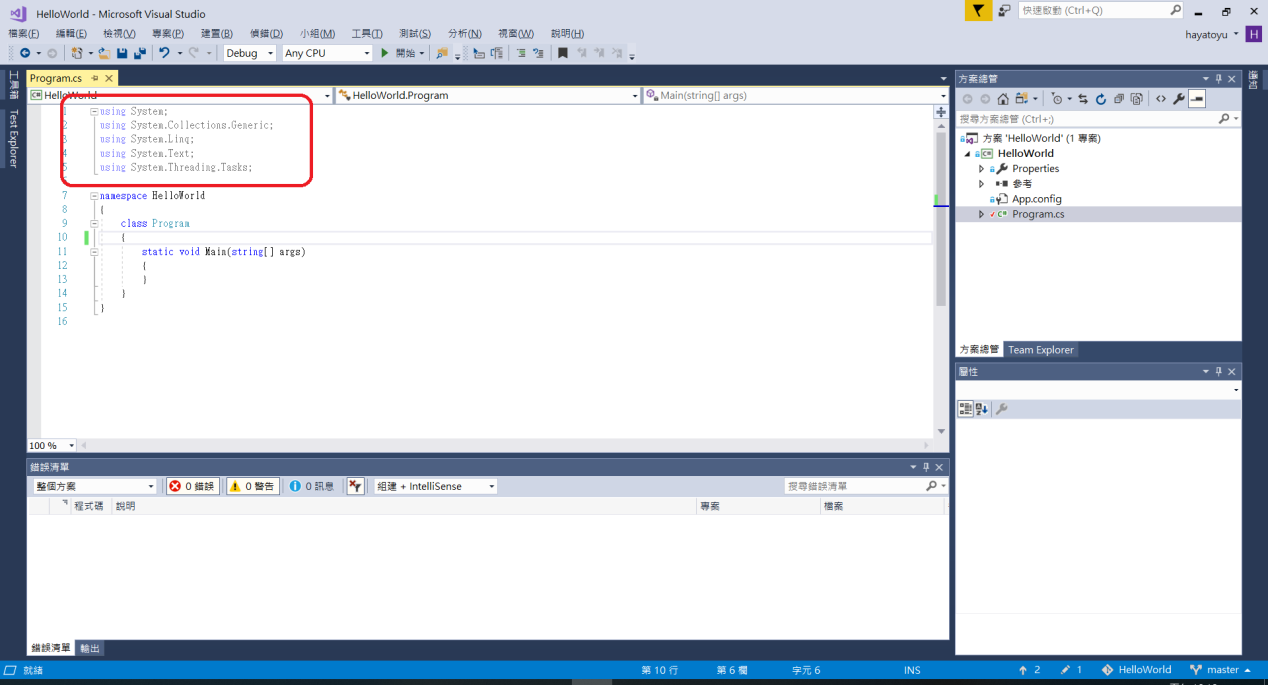
## 1.3 程式設計入門

稍微了解Visual Studio的功能後，我們還是要回到主題，此書旨在教導基礎程式設計，不能光介紹工具而已。現在讓我們來看看Program.cs吧。

### 1.3.1 引用函式庫與基本語法

.Net Framework內建相當多的函式庫，可供使用者在開發上直接引用。在程式碼編輯器的上方，可以看到有一堆using…的敘述句。這表示在此程式碼檔案中，我們將會引用這些函式庫協助我們進行開發，我們可以直接呼叫函式庫中所定義的類別及方法而不需要重新撰寫。

與許多著名的語言相同，在每一句敘述的結尾都會打上分號(;)，以表示該敘述句結束。

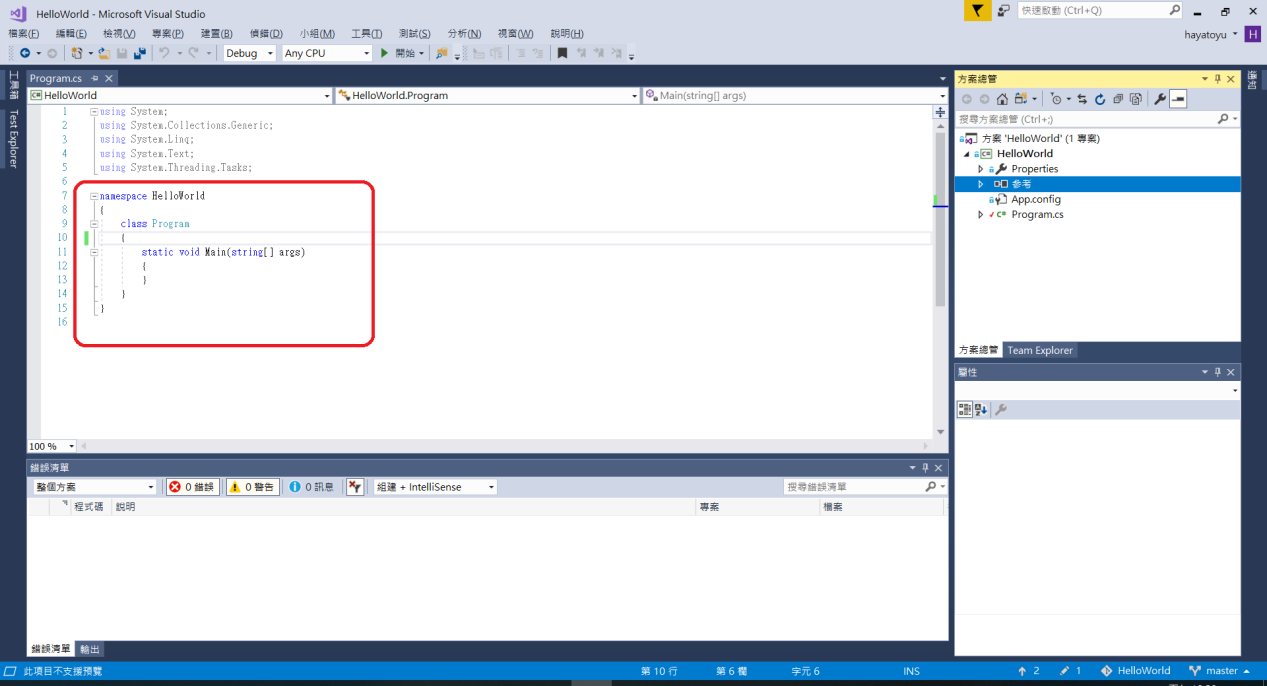


|  |  |
| --- | --- |
| warning_48 | 引用函式庫有可能造成函式庫名稱重複，此時編譯器便會提示。解決方法是可以另外命名函式庫的別名，或者辛苦一點直接以全名呼叫。 |

|  |  |
| --- | --- |
| paper&pencil_48 | 除了.Net Framework內建的函式庫外，也可以引用第三方函式庫或自己撰寫的函式庫(如Dll檔等)，方法是在方案總管中的參考上點滑鼠右鍵，選擇加入參考。 |

### 1.3.2 程式碼的撰寫

在引用的下方，從namespace開始，就是我們程式碼邏輯的區域。



每一個區域範圍，會使用大括號({})包覆。在此我們可以看到分為數個層級：命名空間(namespace)、類別(class)以及類別內的Main函式。

在此筆者先簡略的介紹一下，類別是物件導向語言的靈魂，也是最基本的操作單元。類別中可能有變數、屬性、建/解構子、函式等等，而類別需要實例化(Initialize)，才可在記憶體中配置空間並生成物件(object)。類別與物件的關係，就像你拿到了一台汽車的設計圖，它告訴你汽車該怎麼製造，有哪些零件及功能，但你光有汽車設計圖依然是當不了老司機的，你必須依照設計圖把這台車造出來，才有車可以開。設計圖就像是程式語言中的類別(class)，而你依照設計圖造出來的車，則是物件(object)。透過實例化，我們可以在記憶體中配置物件。想當然，你可以依照一張設計圖造出好多台車，所以一個類別也可以實例化出許多個物件，只是這些物件都必須遵守類別定義的規範而已。

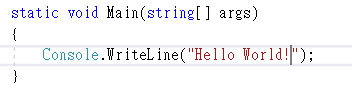
命名空間(namespace)中，可能有很多個類別，在同一個命名空間下的類別，可以互相參考。如果想參考不同命名空間下的類別，就必須先引用。

最後，這個專案的主角，Main函式。Main函式是程式的進入點，也就是程式在啟動時，會從Main函式開始執行。所以現在我們想完成這個專案，就必須把程式碼加入到Main函式中。

|  |  |
| --- | --- |
| paper&pencil_48 | 事實上類別、函式等在宣告時是可以定義存取範圍的，不一定都會被外部存取。筆者會在之後的章節說明。 |

### 1.3.3 Hello World！

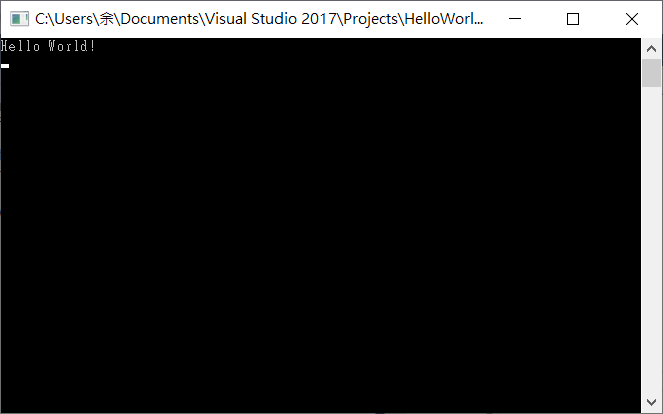
在Main函式中加入以下程式碼：



這行程式碼，呼叫了Console類別中的WriteLine()函式。Console類別是針對主控台應用程式的操作類，而其WriteLine()函式的功能，為將參數輸出至主控台，並在結束時換行。參數即為括號中以雙引號包覆的字串，也就是說這行程式碼的功能是在主控台上輸出Hello World字樣。

在編寫完成後，我們可以按下Visual Studio畫面上方的開始來執行程式。也可以直接點鍵盤上的F5來執行。

執行後的畫面應該如下：



|  |  |
| --- | --- |
| lightbulb_48 | 如果你發現主控台熟悉的黑色畫面一閃即逝，根本來不及看執行結果，那是因為程式一執行完就會自動關閉。您可以使用Ctrl + F5，讓程式執行完後點擊任意鍵再結束，不過這方法就不適用偵錯模式了。 |

|  |  |
| --- | --- |
| lightbulb_48 | 請注意，放在Console.WriteLine()中的字串一定要使用雙引號包覆。這樣編譯器才會判定它是字串資料。 |

至此我們的第一個 Hello World 專案便大功告成了，距離您學會程式設計，也就邁進了一大步了。在下一個章節，我們會介紹資料型態與變數的宣告，這會是極其無聊的一個章節，但我們還是希望您能耐著性子看完，我們相信這對您是會有幫助的。