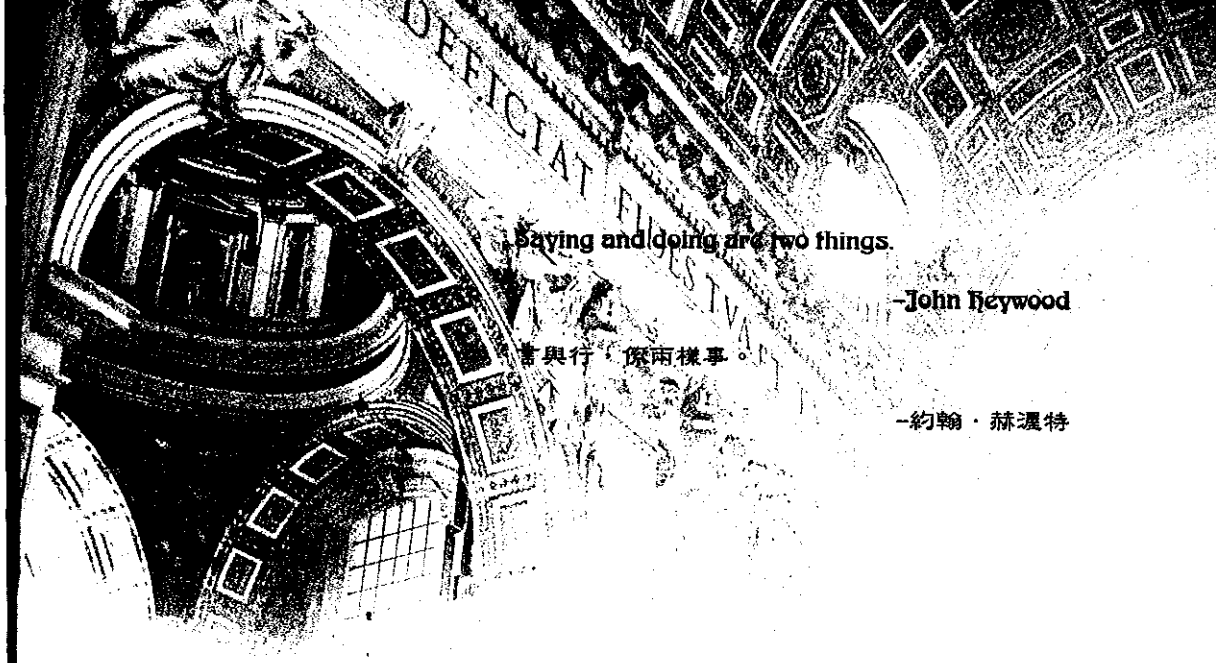


Saying and doing are two things.

–John Heywood

言與行，係兩樣事。

–約翰・赫濃特

1

簡介 MFC 視窗程式設計

本章導讀

學習視窗程式設計，往往讓人有一種難度頗高的感覺，這的確也是事實，相較於執行於**命令提示字元**模式之程式的開發，撰寫視窗程式的確技術高深、原理艱難了些。但這也是因為視窗程式擁有更人性化、更直覺的操作介面。這一章將簡單介紹撰寫視窗程式的一些基本觀念，並介紹 Visual Studio 這個好用的視窗程式整合開發環境。

各節標題

1-1 本書的主軸

目前語言的種類非常多，從 C/C++、Pascal、Visual Basic 到 Java，每種語言都可以開發視窗程式，由 Microsoft 所提出的 C#，更不在話下。本書的內容主要在講解如何運用 C++語言，並以 Microsoft 發行的 Visual Studio 為開發工具運用 MFC 程式庫，撰寫執行於 Windows 作業環境的視窗程式。

這本書相信能夠在帶領您學習 MFC 視窗程式設計的過程中，提供很大的幫助。

1-2 視窗程式設計的基本觀念

1-2-1 事件、訊息與視窗運作

當撰寫視窗程式時，在您的腦海裡，將不斷地詢問自己這樣的問題。

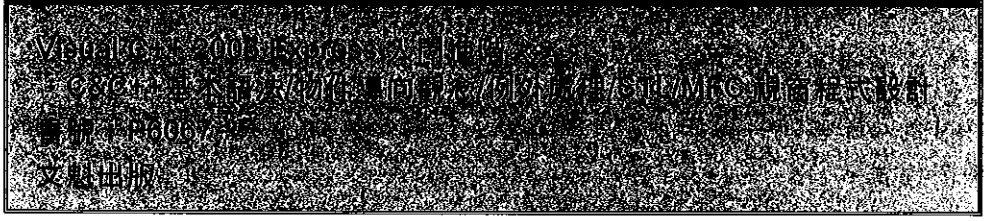
使用者會執行什麼動作？程式如何回應這些動作？

為什麼在設計視窗程式時，會思考這些問題呢？那是因為視窗的運作基礎就是事件回應。相較於**命令提示字元**模式下大部份以流程觀念為主的程式撰寫方式，視窗程式以**事件回應**為思考方向的設計觀念，對於初學者是比較難適應的。

補給小站 所謂的**命令提示字元模式**，是指**命令提示字元**視窗的文字畫面。欲開啟**命令提示字元**視窗，請執行 [開始 /所有程式/附屬應用程式/命令提示字元] 指令。

不過，若您已具備物件導向觀念，適應上並不會太難。若不懂什麼是物件導向？建議最好先看看以下這本書，學習一下物件導向觀念。

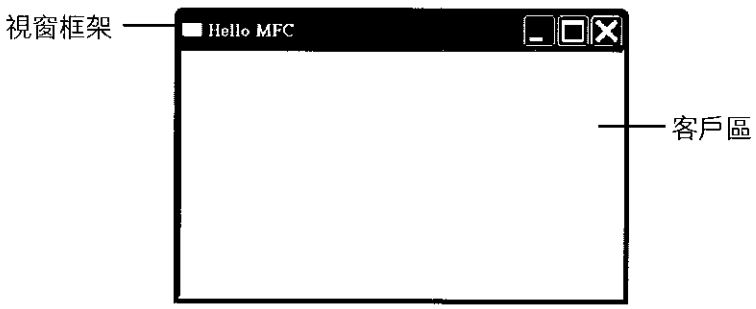
好書推薦



以**視窗**為介面的軟體操作環境下，使用者會透過滑鼠、鍵盤...等周邊設備，提出許多不同的**事件要求**，如：按下按鈕、點選功能表...等。這些由使用者觸發的事件（不論是按下按鈕，或點選功能表），都將產生訊息傳達給視窗。因此，在視窗執行的過程裡，使用者產生的**事件**，在程式裡都將化身為**訊息**傳遞給視窗。而不同事件將產生不同訊息，因此，在程式裡，將依照訊息種類建立回應機制。執行視窗程式的過程裡，使用者與程式的執行是一個交互影響的過程。程式的執行過程，將視使用者所觸發事件產生的訊息，而執行不同程式敘述，並做出相對的回應。

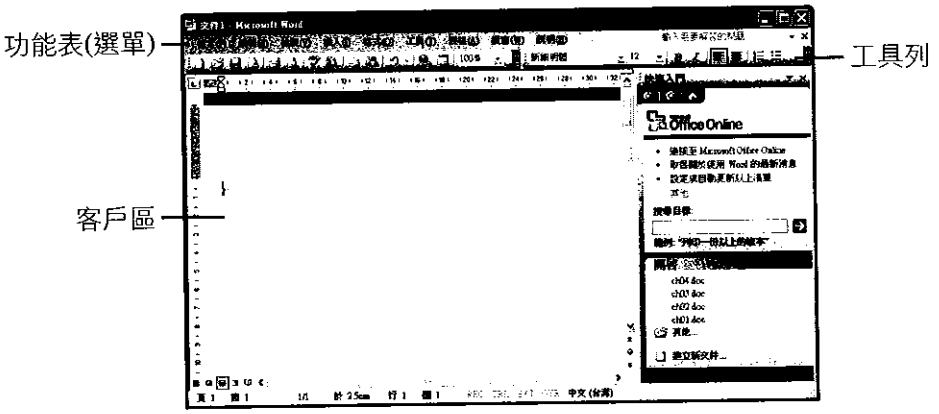
1-2-2 視窗的基本構造

下圖是最基本的視窗形式，整個視窗大致可分為兩個部份，一是**視窗框架**，二是**客戶區**（Client Area）。



視窗框架用於**容納**視窗介面的操作元件，一般視窗程式常見的視窗元件有功能表、工具列、狀態列...等。而**客戶區**（Client Area）則是使用者**工作**的區域，使用者可以在工作區輸入文字、繪圖...等，因此，您亦可以將此區域稱為**工作區**。

下圖是文書編輯軟體 – Word 的視窗介面，在該視窗介面裡，擁有常見的視窗元件與編輯文件用的客戶區。



1-2-3 資源的觀念

什麼是資源

前面介紹了訊息在視窗程式運作過程所扮演的角色，有了這樣的基本認知，接著，再來談談視窗程式的資源觀念。

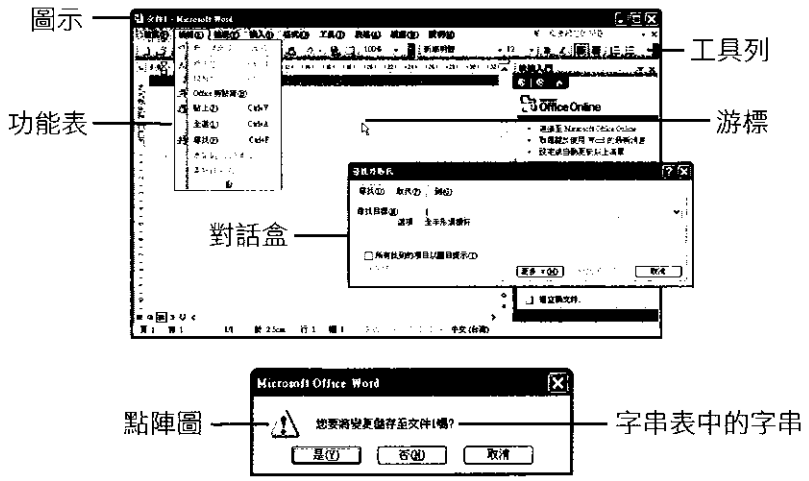
相對於**命令提示字元**模式下執行的程式，視窗程式的圖形化介面是相當複雜且多變化的。圖形化介面可以擁有功能表、工具列、狀態列...等視窗元件。這些協助使用者操作程式視覺化物件，在程式撰寫時，將被視為程式建立視窗介面所使用的**資源**。

資源檔的使用

前述視窗程式所使用的資源物件，開發視窗程式時，將定義於資源檔裡。下表中列出在 Visual Studio 以 MFC 開發視窗程式時，資源檔使用的各種視窗資源：

中文名稱	英文名稱
加速鍵表	Accelerator
點陣圖	Bitmap
游標	Cursor
對話盒	Dialog
圖示	Icon
功能表（選單）	Menu
字串表	String Table
工具列	Toolbar
版本資訊	Version

以下兩圖將把上表所提及的資源，利用 Word 的視窗介面加以說明，相信更能讓您瞭解資源在視窗應用程式中扮演的角色。



資源檔的建立

對於資源檔的建立，可運用 Visual Studio 所提供的資源編輯器建立/編輯您的資源，並自動產生資源檔，資源編輯器的使用請參考第 7 章。

1-3 如何撰寫視窗程式

1-3-1 什麼是 Application Frameworks

想要撰寫視窗程式，有許多方法，其中一個方法，是利用由軟體廠商所提供的類別庫，這種類別庫稱之為 Application Frameworks（以下簡稱為 AF）。什麼是 AF 呢？照字面的含意，可以翻譯成『**應用軟體架構**』。從字面含意可以清楚地瞭解，AF 的用途在提供建立視窗的軟體架構。

當利用 AF 建立視窗程式時，AF 將提供所有視窗程式共有的部份，具象的有視窗、功能表、工具列...等，不具象的有訊息的攔截機制。也就是說，AF 已經完成視窗程式的基本架構。程式設計師只要利用它，就可以擁有視窗的基本功能，其餘的心思只要放在增加或修改程式功能，符合自己的需求即可。

其實利用 AF 開發視窗程式與使用程式語言撰寫程式是一樣的，都是利用別人的成果，發展符合自己需求的程式。只是使用程式語言撰寫程式時，利用的是前人定義出來的特定語法，這些特定語法將操作電腦的細節包裝起來。就像在 C++中，將資料輸出到螢幕時，只需要使用 cout 物件，但不必去管究竟電腦如何從程式得到資料，又如何將這些資料輸出到螢幕上。同樣的道理，使用 AF 建立視窗程式時，亦不必理會電腦究竟如何在螢幕上顯示一個視窗，反正別人寫好的拿來用就是了。

1-3-2 AF 與物件導向

從以上說明可以看出 AF 實現了軟體再用的理想，這跟物件導向技術有密不可分的關係，因為 AF 本身就是一個很複雜的類別階層，可運用類別的**繼承**與**聚合**觀念。前者繼承 AF 的類別，以修改方式產生視窗，如：繼承文字視窗類別。後者則是直接使用 AF 的類別建立視窗元件，如：使用按鈕類別。所以，物件導向觀念是學習視窗程式設計的基本能力之一，有了物件導向觀念，才能充分利用 AF，設計出好的視窗程式。

1-4 強大的整合程式開發環境 - Microsoft Visual Studio 2005

1-4-1 什麼是整合程式開發環境

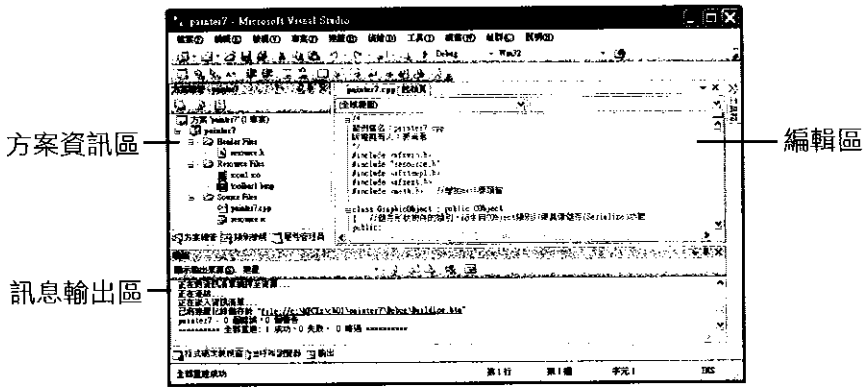
當進行程式開發時，一個提供程式編輯與除錯的良好程式開發環境，將可以提高程式設計師的工作效率。對於開發視窗程式而言，更是這樣，因為視窗程式除了程式碼以外，還有需要使用其他資源，如：圖示、功能表、對話盒...等。

若沒有一個能夠提供建立、編輯這些程式資源的工具，那對於程式設計師而言，開發視窗程式將會是一場夢魘。Visual Studio 就是一個整合各種視窗程式開發所需工具的工作環境，它除了提供上述程式資源的建立工具，便利視窗程式的開發外，更提供方便的除錯工具，可有效節省程式設計師花費在為程式除錯上的時間與精力。

除此之外，對於視窗程式設計方面，Visual Studio 更提供了精靈，協助使用者快速建立視窗程式的大致架構，讓程式設計師只需要撰寫部分程式碼，即可完成一個視窗應用程式。但是學習撰寫視窗程式的初期，並不建議您直接學習精靈的應用，因為在不瞭解整個視窗程式建立的過程時，就學習應用精靈，可能學了半天，還是不知道視窗程式的架構與原理。所以，在本書前半部裡，將以赤手空拳的方式建立視窗程式，到了後半部，才告訴您如何應用精靈的協助建立視窗程式，相信這樣的學習過程是比較適合初學者的。

1-4-2 Visual Studio 的畫面介紹

乍見 Visual Studio 的視窗畫面，相信大部分的人，都會被複雜的視窗工作畫面嚇住。不過，為了提供程式設計師具備完整訊息的工作環境，複雜的工作畫面是可理解的。因此，Visual Studio 建立的每個應用程式，除了有原始碼檔，以及機器碼檔外，還有儲存其他相關資訊的檔案，而這些檔案將會被組織成一個方/專案，方便我們利用 Visual Studio 管理它們。下圖是 Visual Studio 的工作畫面：



整個工作畫面主要分為三個區域。

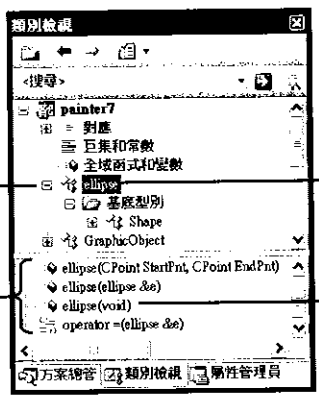
一、編輯區

顧名思義**編輯區**就是撰寫程式或編輯資源的區域。您可在這個區域撰寫程式，或者運用編輯器建立視窗程式所使用的相關資源。

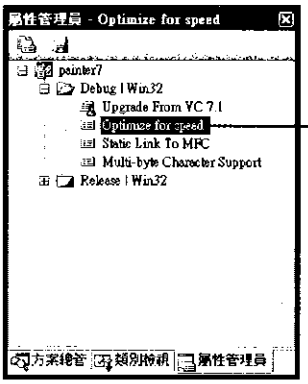
二、方案資訊區

該區有三個標籤。

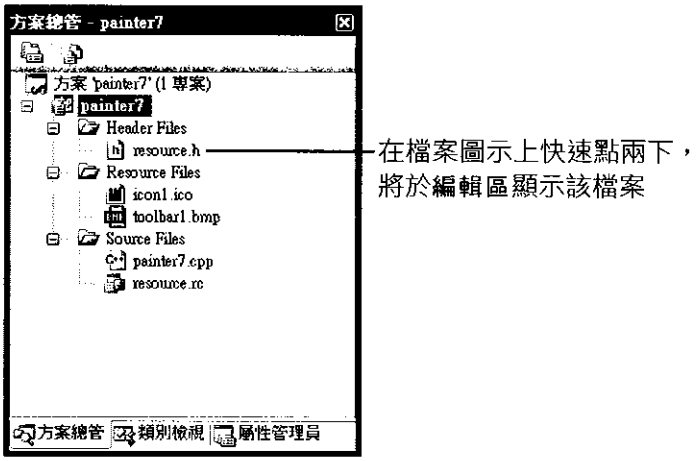
1. **類別檢視**：切換至該標籤後，您可以看到該專案內的所有類別，按下某類別前的 ⊞ 號，即可展開該類別，將可瞭解該類別的基底類別（基礎類別）。選取該類別項目，下方的視窗將顯示該類別的成員。於類別項目上連按兩下，將開啟定義該類別的程式碼，一般來說為**標頭檔**（.h 檔）。若欲更改該類別某成員的程式碼，更可直接在該成員上用滑鼠點兩下，右方的編輯區將開啟該類別成員的程式碼。



2. **屬性管理員**：按下標籤將可切換至檢視專案的屬性設定。

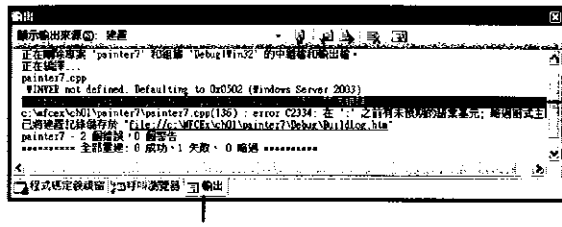


3. **方案總管**：切換至該標籤時，可知道該專案有哪些檔案。在欲檢視的檔案圖示上點兩下，右邊的**編輯區**將開啟該檔案。



三、訊息輸出區

訊息輸出區主要將顯示專案的編譯與錯誤訊息，與類別程式碼的定義內容，您可運用索引標籤切換畫面。當顯示錯誤訊息時，可在欲檢視的錯誤訊息上，快速點兩下，**編輯區**畫面將切換至發生錯誤的程式片段。

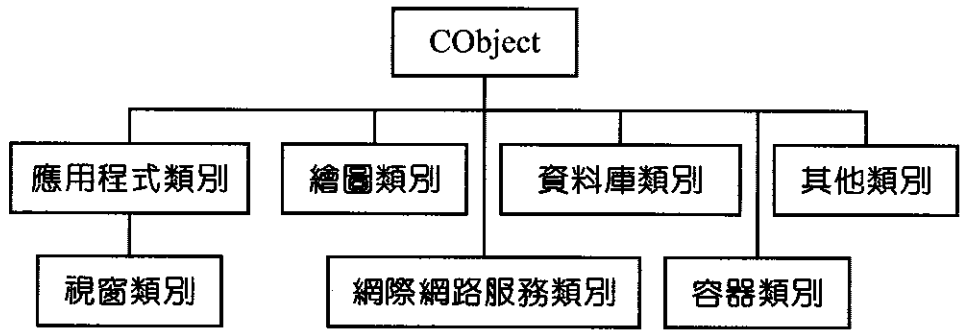


在錯誤訊息上快速點兩下，可在**編輯區**檢視發生錯誤的程式碼

按下，可切換檢視畫面

1-4-3 MFC 的類別繼承階層

什麼是 MFC 呢？MFC 的英文全文為 Microsoft Foundation Class，是 Microsoft 公司搭配在 Visual Studio 中，用於開發視窗程式的 AF。MFC 的類別繼承階層相當複雜，下圖就本書接下來將會介紹的相關類別，做一簡要的整理與介紹。



- CObject - CObject 類別是 MFC 類別庫最基礎的類別。
- **應用程式類別** - 應用程式類別定義了建立應用程式可能需要利用的類別，如：CWinApp（視窗應用程式類別）、CDocTemplate（文件範本類別）、CDocument（文件類別）...等類別。

- **視窗程式類別** - 視窗程式類別衍生於應用程式類別，視窗程式類別將定義建構視窗所需的類別，如：CFrameWnd（視窗框架類別）、CDialog（對話盒類別）、CView（瀏覽類別）...等類別。
- **繪圖類別** - 用於繪圖的類別，如：CDC（裝置內文類別）、CPen（畫筆類別）、CBrush（畫刷類別）...等。
- **網際網路服務類別** - 用於建立網路連結的類別，可建立的連結種類有 Ftp、Gopher、Http 三種。
- **資料庫類別** - 用於建立資料庫連結，取得資料庫資料的類別，主要有 ODBC 資料庫類別、DAO 資料庫類別，以及 CRecordset 類別（記錄集類別）。
- **容器類別** - CList、CArray、CMap...等容器類別。
- **其他類別** - 除了上述幾類別類別外，MFC 中還有例外處理類別、檔案操作類別...等幾類類別。

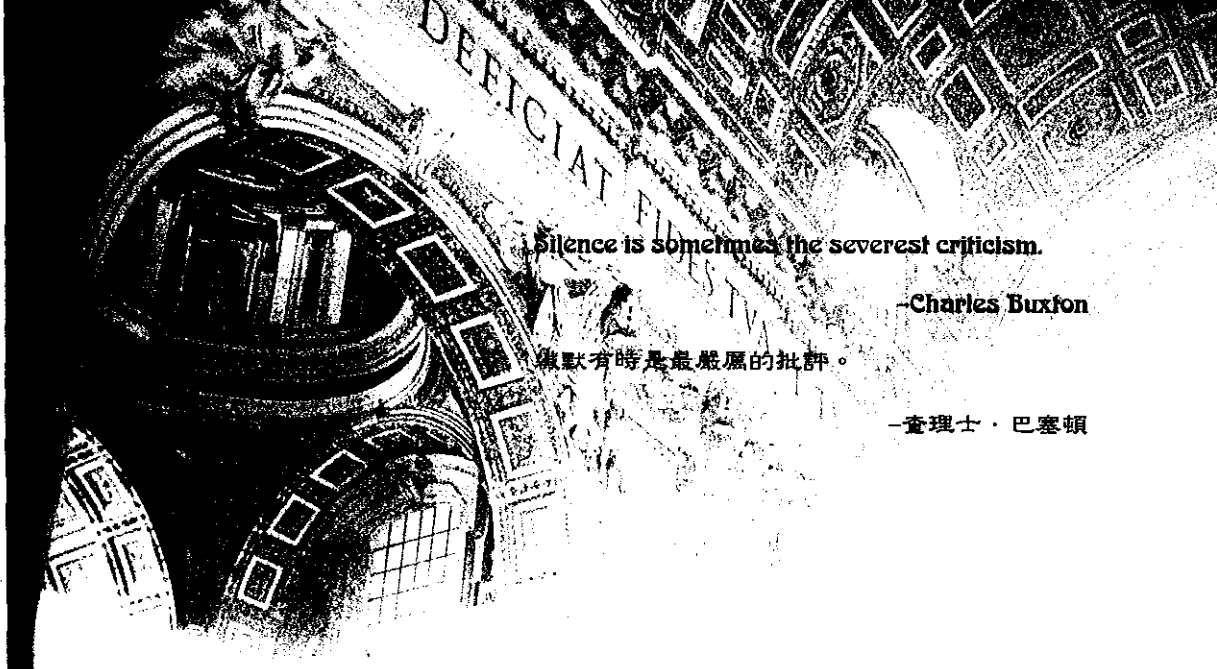
除了繼承 CObject 類別外，MFC 還有幾個本書將使用到的獨立類別。如用於資料儲存於檔案的 CArchive 類別，用於儲存簡單資料的 CPoint、CRect、CString...等。詳細的類別繼承階層圖，請參考 Visual Studio 內有關 MFC 類別庫的說明。

1-4-4 .Net 平台

從 2002 年開始.Net 平台就被 Microsoft 炒得火熱，讓人覺得似乎又是一場程式設計的大革命，尤其對於以往使用 Visual Basic 的程式設計師而言，更是一場挑戰。由於 Microsoft 一再強調 Visual Basic .Net 是全新的物件導向語言，且與以往版本的 Visual Basic 6.0 是多麼地不同之時，心裡忐忑著將面臨重新學習的驚恐心情。再加上又出現了什麼.Net Framework，還有什麼 Common Language Runtime（CLR）。天吶！程式設計師的命真是苦呀！技術三年一小修，五年一大改，真令人應接不暇。

但對以往使用 Visual Studio 開發 Windows 視窗程式的程式設計師來說，問題就小了點，甚至根本就感覺不到問題的存在。雖然 Visual Studio 多了 CLR 與.Net Framework，但它們可以說是另一個世界，至於在 Win32 平台下，利用 C++設計視窗程式，MFC 還是最重要的主角，且在使用上亦沒有很大的變革。

這對以往辛苦學習 Visual Studio 的程式設計師來說，衝擊的確比較小了些，需要重新學習的，大概就是比較不同的開發工具操作介面。雖然沒人能夠確定 MFC 的未來究竟如何，但在未來的 4 到 5 年，肯定它還是會存在。畢竟這麼多年來，許多企業已經投注相當大的心血在這個技術上，不可能短期內做很大的變革。關於.Net 平台概念、.NET Framework、CLR，以及 Visual Studio 關於.Net Framework 的應用，您可以參考 Visual Studio 所提供的說明文件。



Silence is sometimes the severest criticism.

–Charles Buxton

沉默有時是最嚴厲的批評。

–查理士・巴塞頓

2

視窗程式設計的初體驗 - Hello MFC!

本章導讀

這一章將告訴您如何借用 MFC 所提供的類別，建立最簡單的視窗程式，並以該視窗介紹視窗程式的基本架構。整個建立視窗程式的過程，將劃分為 7 個步驟，並以 Step by Step 的方式引導您操作 Visual Studio，建立第一支視窗程式 – Hello MFC。

各節標題

2-1 建立視窗程式的基本觀念

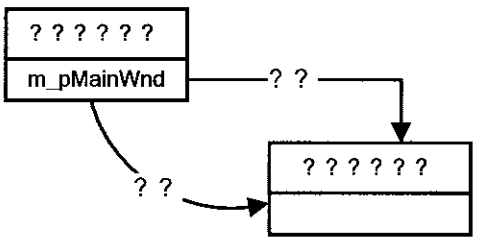
需載入的標頭檔 - afxwin.h

撰寫視窗程式時，必須載入 afxwin.h 標頭檔。該標頭檔內定義了所有 MFC 類別。

應用程式物件與視窗框架物件

即使我們建立的程式，僅是顯示在螢幕上的視窗框架，並不具任何功能的視窗程式。在此程式裡，還是必須建立兩種最基本的物件，一個是**應用程式物件**，另一個是**視窗框架物件**。

前者代表整個應用程式，後者則代表應用程式的介面。因此，每一個程式裡，均將建立一個**應用程式類別**，用於建立**應用程式物件**。在建立應用程式物件過程中，該物件將負責建立**視窗框架物件**，並將繼承的 m_pMainWnd 屬性（型態為 CWnd *）指向所使用的視窗框架物件，最後顯示視窗，兩者的關係如下圖：



借用 CWinApp 與 CFrameWnd 類別

雖然我們都不是孔明，不懂得如何利用草船借箭。但在學習利用 MFC 撰寫視窗程式時，必須懂得如何借用 MFC 類別建構視窗程式。

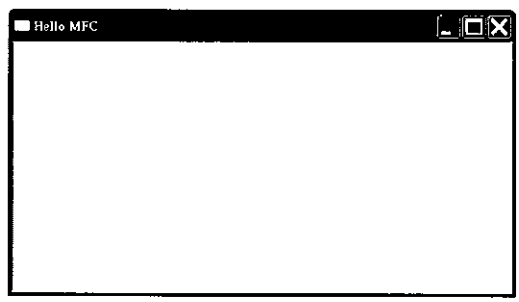
撰寫應用程式時，借用 MFC 類別的方式，大致來說有兩種，一是直接借用 MFC 提供的類別建立物件，另一種則是以繼承方式借用 MFC 類別，再根據自己的需求修改類別。以前面說到的應用程式物件與視窗框架物件為例，它們都是借用 MFC 類別而建立的。

本節 Hello MFC 程式範例借用 MFC 類別的方法主要有兩種，一是以**繼承** CWinApp 類別，建立應用程式類別，二是**直接使用** CFrameWnd 類別產生視窗框架物件。

2-2 您的第一個視窗程式 - Hello MFC

2-2-1 建立您的 Hello MFC

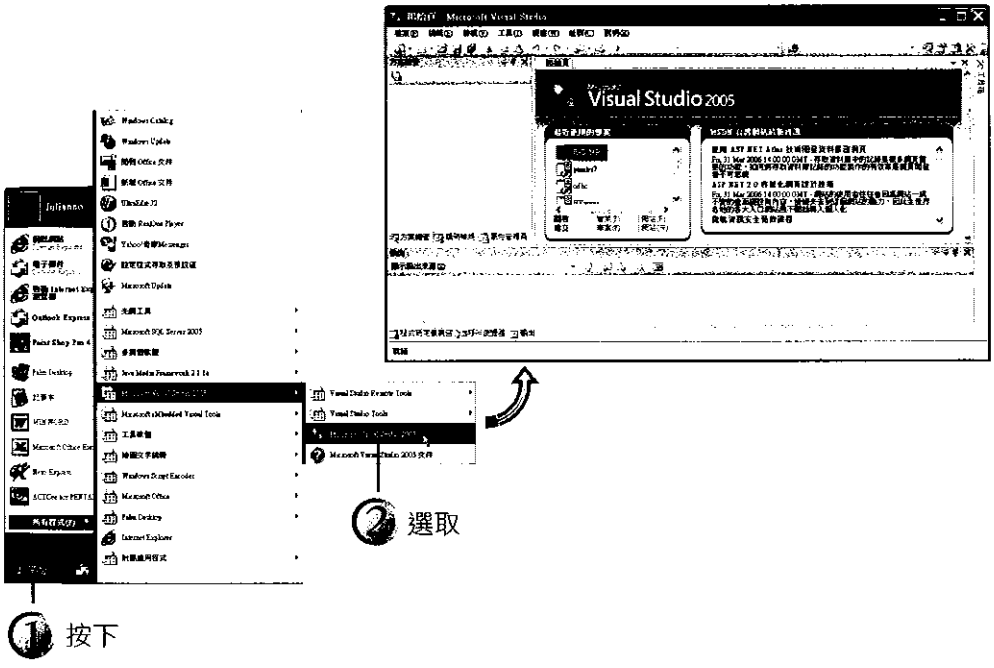
學習 MFC 建立視窗程式的第一步，將以一個簡單的 Hello MFC 程式範例做為起點。下圖為 Hello MFC 程式範例的執行結果。



前面說到撰寫視窗程式需要運用許多資源，當利用 Visual Studio 建立視窗程式時，Visual Studio 將為視窗程式建立專案資料夾，該資料夾下將建立許多檔案。這些檔案除了視窗程式的原始碼、資源檔外，還包括一些其他檔案，目前暫且不介紹這些檔案，將留待 3-1-1 節再做說明。本節先以 6 個步驟介紹如何利用 Visual Studio 建立第一個**視窗程式專案**。

Step 1 執行 Microsoft Visual Studio 2005

執行 開始 功能表的 [所有程式/Microsoft Visual Studio 2005/Microsoft Visual Studio 2005] 。



Step 2 進入新增專案對話盒

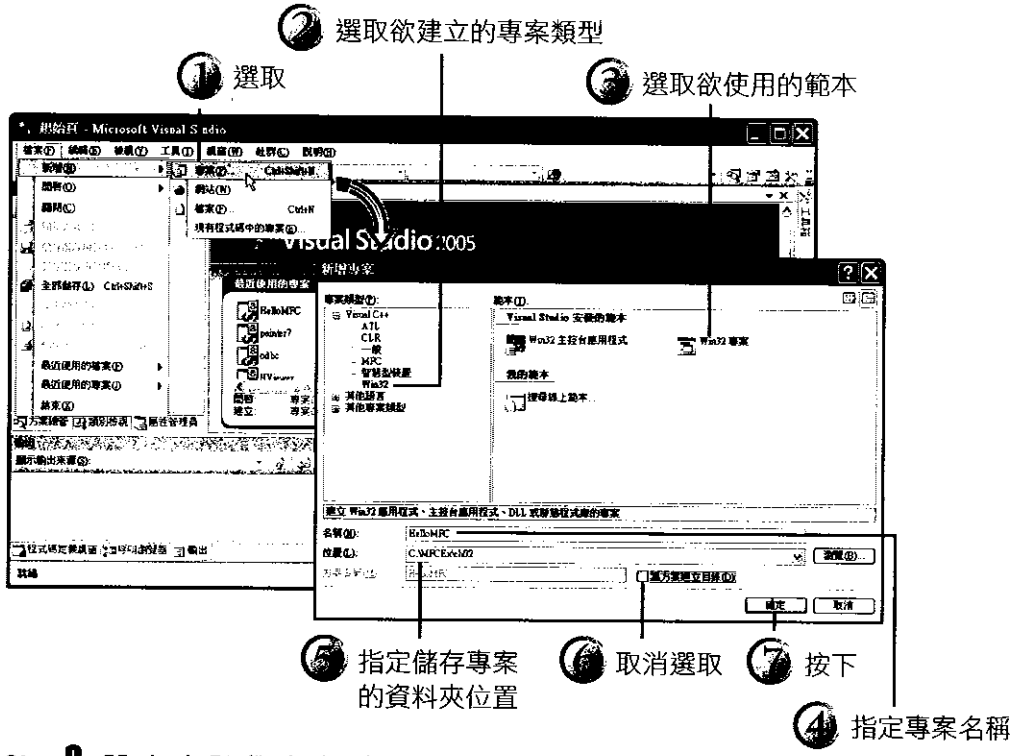
欲建立新視窗程式專案時，請選取 [**新增/專案**] 選項，進入**新增專案**對話盒，並於該對話盒內完成以下與建立專案有關的工作：

一、選取欲建立的專案類型，在這裡請選擇 Win32 專案。

二、輸入專案名稱

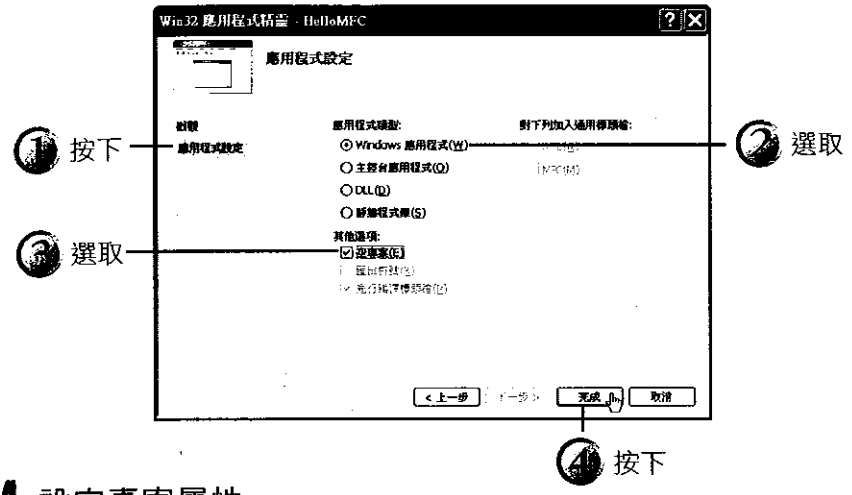
三、選擇欲儲存該專案的資料夾

完成這些工作後，按下 確定 按鈕將進入下個步驟，此處建立的專案名稱為 HelloMFC，而 Visual Studio 將自行於選定的資料夾路徑中，增加一個與鍵入專案名稱相同的子資料夾名稱。



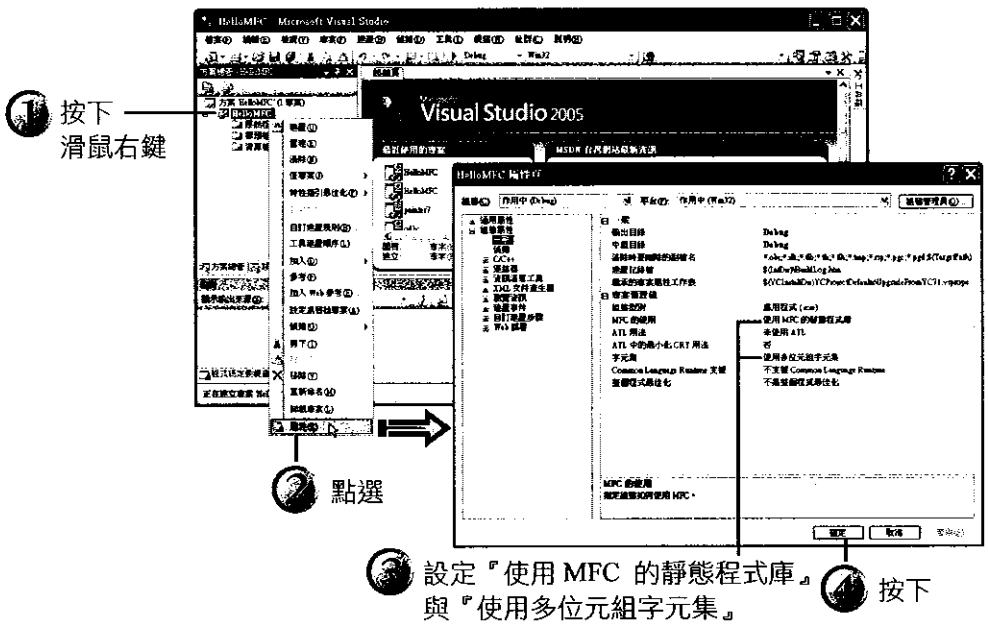
Step 3 設定應用程式資料

進入 Win 32 應用程式精靈對話盒，切換至應用程式設定畫面，執行應用程式資料的設定，請點選 Windows 應用程式選項，並設定建立空專案，再按下 完成 按鈕。



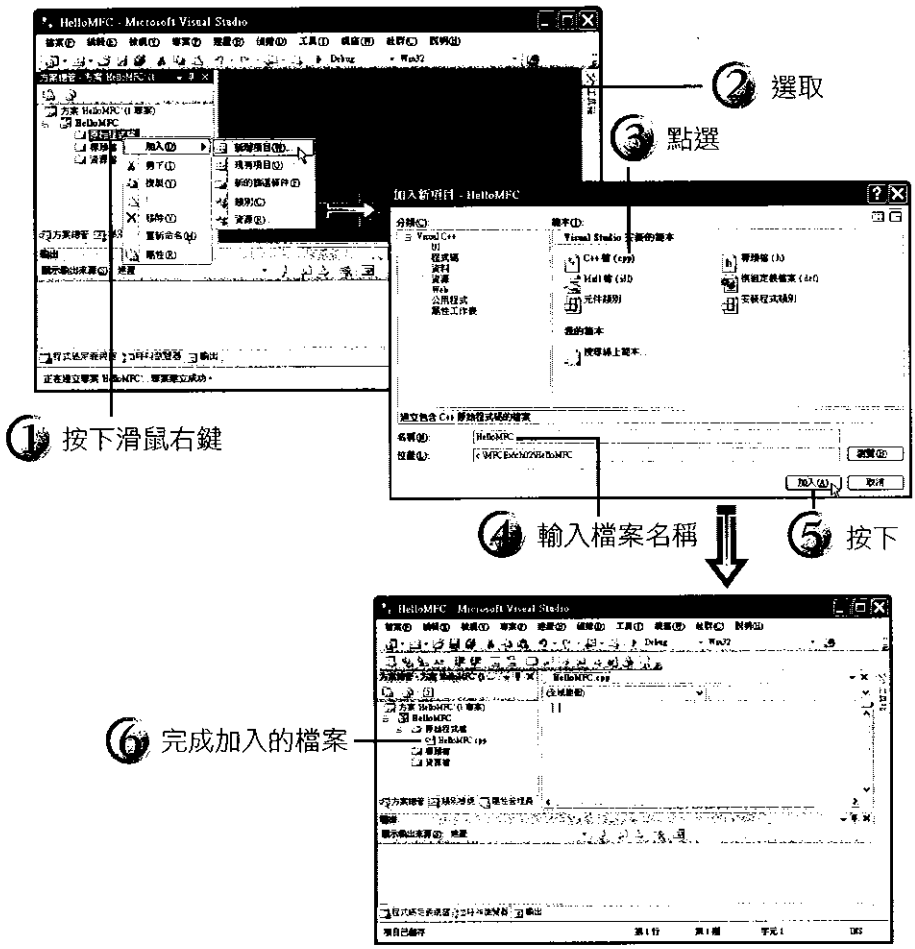
Step 4 設定專案屬性

欲建立視窗應用程式的專案，**千萬別忘了**設定專案使用 MFC 程式庫與多位元組字元集。不然，編譯程式時，將產生錯誤訊息。設定方法是在欲設定的專案上，按下滑鼠右鍵點選**屬性**選項，進入 HelloMFC 屬性頁。



Step 5 建立空白檔案

完成上述動作的操作後，於方案的檔案分類資料夾項目上按下滑鼠右鍵，點選 [加入/新增項目] ，將於編輯區開啟一個空白檔案。



然後，鍵入以下程式碼。

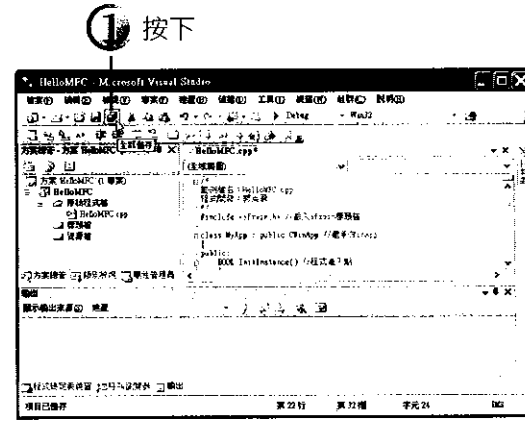
HelloMFC 程式範例 - Hello MFC！程式內容

檔案位置：HelloMFC\HelloMFC.cpp

```
1   /*
2   範例檔名：HelloMFC.cpp
3   程式開發：郭尙君
4   */
5   #include <afxwin.h> //載入 afxwin 標頭檔
6
7   class MyApp : public CWinApp //繼承 CWinApp
8   {
9   public:
10     BOOL InitInstance() //程式進入點 <-- 程式進入點
11     {
12        CFrameWnd *Frame = new CFrameWnd(); //建立 CFrameWnd 物件
13        m_pMainWnd = Frame; //將 m_pMainWnd 設定爲 Frame
14
15        Frame->Create(NULL,"Hello MFC"); //建立視窗
16        Frame->ShowWindow(SW_SHOW); //顯示視窗
17
18        return true;
19     }
20  };
21
22  MyApp a_app; //建立一個應用程式物件，MyApp 爲自訂的應用程式類別
```

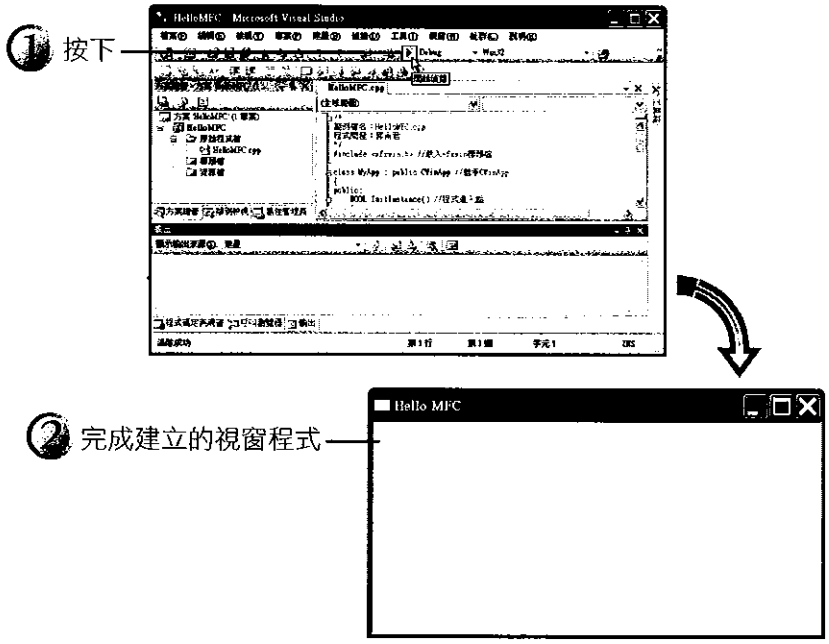
Step 6 將檔案加入專案中

按下工具列的**全部儲存**按鈕，完成方案的儲存。



Step 7 編譯並執行程式

按下開始偵錯 ▶ 按鈕，執行程式的編譯與執行。



HelloMFC 程式範例建立出的視窗程式相當簡單，只是一個空白的視窗框架，透過這個程式範例將介紹以下重點：

1. CWinApp 類別與程式進入點（2-2-2 節）
2. 視窗框架物件的建立（2-2-3 節）

2-2-2 CWinApp 類別與程式進入點

自訂應用程式類別

欲自訂應用程式類別所需要完成以下工作：

1. 繼承 CWinApp 類別
2. 重載 CWinApp::InitInstance()函數，此函數為視窗程式的進入點，回傳值的型態為 BOOL。

```
'摘自 HelloMFC\HelloMFC.cpp
7       class MyApp : public CWinApp //繼承 CWinApp
8       {
9       public:
10         BOOL InitInstance() //程式進入點 <-- 程式進入點
...     { ... }
20      };
```

CWinApp::m_pMainWnd 屬性

自訂應用程式類別除了必須重新定義繼承於 CWinApp 類別的 InitInstance()函數外，還繼承 CWinApp 類別的 m_pMainWnd 屬性（實際上此屬性是由 CWinApp 類別繼承自 CThreadWnd 類別），此屬性將指向**應用程式物件**使用的**視窗框架物件**。CWinApp::InitInstance()函數必須設定該屬性，以下將說明程式進入點 – CWinApp::InitInstance()函數的執行過程。

程式的進入點 – CWinApp::InitInstance()函數

當建立繼承於 CWinApp 類別的應用程式類別時，必須重新定義 CWinApp::InitInstance()函數，使之成為視窗程式的進入點。通常在該函數中，將完成以下工作：

1. 產生視窗框架物件（第 12 行）
2. 將視窗框架物件的位址設定給 CWinApp::m_pMainWnd 屬性（第 13 行）
3. 建立視窗框架（第 15 行）
4. 在螢幕中，顯示視窗框架物件。（第 16 行）

以下為重新定義之 InitInstance()函數的程式片段。

```
'摘自 HelloMFC\HelloMFC.cpp
10      BOOL InitInstance() //程式進入點 <-- 程式進入點
11      {
12          CFrameWnd *Frame = new CFrameWnd();
            //建立 CFrameWnd 物件
13          m_pMainWnd = Frame;
            //將 m_pMainWnd 指向視窗框架物件 Frame
14
15          Frame->Create(NULL,"Hello MFC");  //建立視窗
16          Frame->ShowWindow(SW_SHOW);  //顯示視窗
17
18          return true;
19      }
```

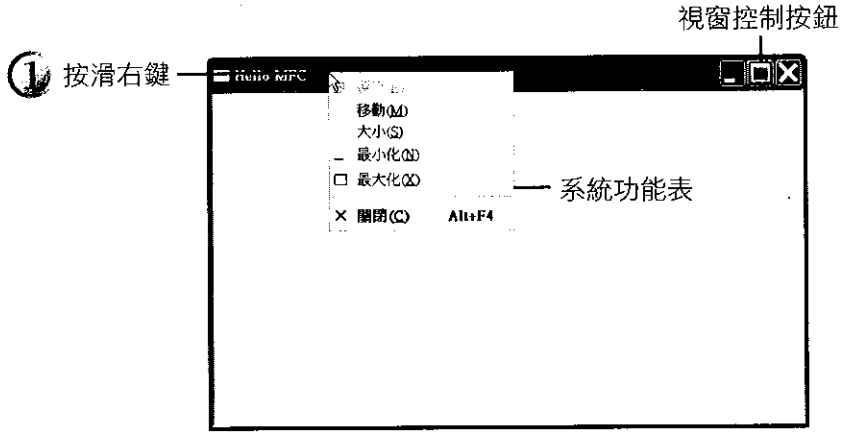
建立應用程式物件

建立自訂應用程式類別後，必須利用該類別建立一個應用程式物件。當程式建立應用程式物件時，將呼叫 InitInstance()函數執行程式。

```
'摘自 HelloMFC\HelloMFC.cpp
22    MyApp a_app; //建立一個應用程式物件，MyApp 爲自訂的應用程式類別
```

2-2-3 視窗框架物件

視窗框架物件就是您在電腦螢幕上看到的視窗，HelloMFC 程式範例將直接利用 CFrameWnd 類別，建立供應用程式物件建立的視窗框架物件。由 CFrameWnd 類別建立的視窗框架物件，將提供一般視窗所應具備的基本功能。所以，即使您不作任何的修改，產生出來的視窗框架，都將如下圖的視窗般，具有系統功能表以及右上角的三個視窗控制按鈕。



產生、建立、顯示視窗框架物件

請注意！**產生**一個視窗框架物件與**建立**、**顯示**視窗，並不是同一件事，因為在視窗程式裡，產生一個視窗框架物件（第 12 行）是指在程式裡建立了一個視窗框架物件。當程式欲操作視窗框架時，可以直接操作該物件。而建立一個視窗框架物件，則是指利用 CFrameWnd::Create()函數（第 15 行），向作業系統註冊這個視窗框架。經過產生與建立視窗框架的過程後，才能利用 CFrameWnd::ShowWindow()函數（第 16 行），將視窗顯示在螢幕上。

```
'摘自 HelloMFC\HelloMFC.cpp
10      BOOL InitInstance() //程式進入點 <-- 程式進入點
11      {
12          CFrameWnd *Frame = new CFrameWnd();
            //建立 CFrameWnd 物件
13          m_pMainWnd = Frame; //將 m_pMainWnd 設定爲 Frame
14
15          Frame->Create(NULL,"Hello MFC");  //建立視窗
16          Frame->ShowWindow(SW_SHOW);   //顯示視窗
17
18          return true;
19      }
```

以下為 Hello MFC 程式範例所使用函數的說明。

```
BOOL virtual CFrameWnd::Create(
    LPCTSTR lpszClassName, LPCTSTR lpszWindowName,
    DWORD dwStyle = WS_OVERLAPPEDWINDOW,
    const RECT& rect = rectDefault, CWnd* pParentWnd = NULL,
    LPCTSTR lpszMenuName = NULL, DWORD dwExStyle = 0,
    CCreateContext* pContext = NULL)
```

□ 函數說明

建立視窗框架物件。

◆ 參數說明

✓ LPCTSTR lpszClassName
lpszClassName 指向一個用於儲存視窗類別名稱的資料結構。在 HelloMFC 中，這個參數將設定為 NULL，表示使用 MFC 預設的視窗框架產生一個標準的視窗。

✓ LPCTSTR lpszWindowName
視窗標題欄內的文字。

✓ DWORD dwStyle = WS_OVERLAPPEDWINDOW
以特性常數指定視窗形式，預設為 WS_OVERLAPPEDWINDOW。

特性常數	說　明
WS_BORDER	有外框
WS_CAPTION	有標題欄， 但不可與 WS_DLGFRAME 樣式並用。
WS_CHILD	此視窗為子視窗，但不能與 WS_POPUP 並用。
WS_CLIPCHILDREN	當在父視窗中繪圖時，將會避開被子視窗遮蓋的部份不畫。
WS_CLIPSIBLINGS	當子視窗收到一個重繪訊息時，將不會重繪被其他子視窗覆蓋的部份。
WS_DISABLED	建立一個視窗時，即宣告該視窗不可接收任何訊息。
WS_DLGFRAME	建立雙外框，但沒有標題的視窗。
WS_GROUP	將視窗設定為第一個控制項，通常用於由控制項群組所組成的對話盒視窗應用程式，並可利用方向鍵在控制項間移動。
WS_HSCROLL	建立一個具有水平捲軸的視窗
WS_MAXIMIZE	建立一個起始大小為最大化的視窗
WS_MAXIMIZEBOX	建立一個有最大化按鈕的視窗
WS_MINIMIZE	建立一個起始大小為最小化的視窗
WS_MINIMIZEBOX	建立一個有最小化按鈕的視窗
WS_OVERLAPPED	建立一個可重疊的視窗
WS_OVERLAPPEDWINDOW	建立一個具有下列特性的視窗 WS_OVERLAPPED, WS_CAPTION、WS_SYSMENU、WS_THICKFRAME、WS_MINIMIZEBOX、WS_MAXIMIZEBOX
WS_POPUP	建立一個蹦現式的視窗（不可與 WS_CHILD 常數並用）
WS_POPUPWINDOW	建立一個蹦現式視窗，並具有 WS_BORDER、WS_POPUP、WS_SYSMENU 特性。
WS_SYSMENU	建立一個左上角可拉出系統功能表的視窗
WS_TABSTOP	設定使用者可以利用 Tab 鍵在視窗的控制項中移動。
WS_THICKFRAME	建立一個可用滑鼠調整大小的細外框視窗
WS_VISIBLE	設定視窗一開始即顯示在螢幕上
WS_VSCROLL	建立一個擁有垂直捲軸的視窗

✓ const RECT& rect = rectDefault
設定視窗的大小及位置，其中位置座標以左上角為視窗原點。

✓ CWnd* pParentWnd = NULL
指向父視窗的指標，若視窗已為最上層的視窗則設定為 NULL。

✓ LPCTSTR lpszMenuName = NULL
指向一個定義在資源檔（RC 檔）的功能表名稱。如果您以識別子的方式代表功能表時，必須利用『MAKEINTRESOURCE（識別子）』傳入此選項。

✓ DWORD dwExStyle = 0
指定延伸視窗形式，下表將說明可使用的常數。

常　數	說　明
WS_EX_ACCEPTFILES	設定視窗接受檔案以拖曳方式開啟
WS_EX_CLIENTEDGE	建立具有 3D 立體感的視窗
WS_EX_CONTEXTHELP	建立一個在標題欄具有問號按鈕的視窗，當使用者按下該按鈕時，視窗將會收到 WM_HELP 訊息。
WS_EX_CONTROLPARENT	允許使用者利用 Tab 鍵操縱子視窗
WS_EX_DLGMODALFRAME	建立一個雙外框，且具有標題欄的視窗
WS_EX_LEFT	讓視窗具有向左對齊的特性，此為預設值
WS_EX_LEFTSCROLLBAR	在視窗的客戶區左方放置一個捲軸。
WS_EX_LTRREADING	以左向右的方式顯示視窗文字，此為預設值。
WS_EX_MDICHILD	建立一個多文件的子視窗
WS_EX_NOPARENTNOTIFY	設定子視窗不必於被建立或消滅時，傳出 WM_PARENTNOTIFY 訊息給父視窗。
WS_EX_OVERLAPPEDWINDOW	= WS_EX_CLIENTEDGE | WS_EX_WINDOWEDGE
WS_EX_PALETTEWINDOW	= WS_EX_WINDOWEDGE | WS_EX_TOPMOST
WS_EX_RIGHT	讓視窗具有向右對齊的特性
WS_EX_RIGHTSCROLLBAR	在視窗的客戶區右方放置一個捲軸，此為預設值。
WS_EX_RTLREADING	以右向左的方式顯示視窗文字
WS_EX_STATICEDGE	建立一個 3D 外框的視窗，並且不接受使用者之資料輸入。
WS_EX_TOOLWINDOW	建立一個工具視窗，該視窗被當成一個浮動的工具列使用。
WS_EX_TOPMOST	設定為此形式之視窗將被置於所有不具 WS_EX_TOPMOST 特性視窗之上，即使該視窗並不處於有效狀態。
WS_EX_TRANSPARENT	建立一個透明視窗
WS_EX_WINDOWEDGE	建立一個具有浮雕外框的視窗

✓ CCreateContext* pContext = NULL
如視窗具備 Document/View 架構時，才須指定，若非則採預設值即可。

```
BOOL CWnd::ShowWindow( int nCmdShow )
```

□ 函數說明

顯示視窗框架物件。

◆ 參數說明

✓ int nCmdShow

控制顯示視窗方式的常數。

常　　數	說　　明
SW_HIDE	隱藏視窗，並將控制權交給其他視窗。
SW_MINIMIZE	將視窗最小化，並將控制權交給目前作業系統中最上層的視窗。
SW_RESTORE	顯示視窗，如果視窗處於最大化或者最小化，則還原到預設的大小。
SW_SHOW	以預設大小，顯示視窗。
SW_SHOWMAXIMIZED	以最大化方式顯示視窗。
SW_SHOWMINIMIZED	以最小化方式顯示視窗。
SW_SHOWMINNOACTIVE	將視窗最小化，縮成一個 icon。
SW_SHOWNA	以目前的狀態顯示視窗
SW_SHOWNOACTIVATE	以上一次顯示的大小與位置顯示視窗。
SW_SHOWNORMAL	顯示視窗，並將其設定為活動視窗。如果視窗為最小化或最大化，則還原至預設的位置與大小。

Slow and steady wins the race.

-Robert Lloyd

慢而穩，獲勝券。

-羅勃．萊德

3

Visual Studio 的操作(一)
- 建立專案與程式編譯/除錯

本章導讀

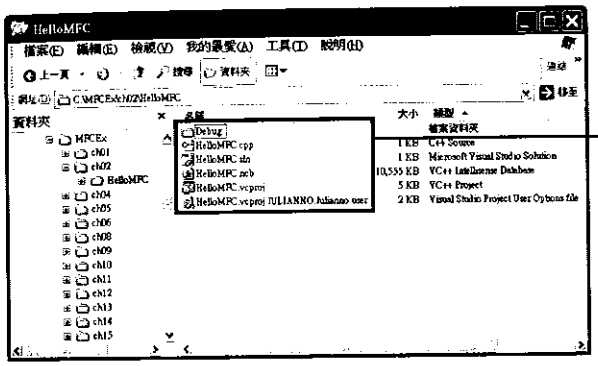
Visual Studio 是一個功能完備的程式開發環境，但也因如此，使得整個視窗介面的操作變得複雜。在本書裡，將以循序的方式，在您學習程式設計的過程裡，穿插介紹 Visual Studio 的使用。這一章將告訴您如何操作專案與檔案，並帶領您使用 Visual Studio 提供的除錯工具，並在最後一小節，介紹 Visual Studio 的線上說明功能，讓您能在即將被龐大的程式碼淹沒時，找到自救的方法。

各節標題

3-1 專案的介紹

3-1-1 Visual Studio 使用的檔案

2-2 節將以 Step by Step 的方式，告訴各位如何建立您的第一個應用程式。Visual Studio 建立應用程式時，將為應用程式建立專案檔（.vcproj）與方案檔（.sln）。在一個方案檔中，將包含數個專案檔。不過，對於一般使用者而言，方案檔通常僅包含一個專案檔。而 Visual Studio 裡，一次僅能開啟一個方案檔。以下將為您介紹 Visual Studio 使用的各種檔案。



HelloMFC 專案資料夾的相關檔案

下表將介紹專案內，幾個較為重要的檔案。

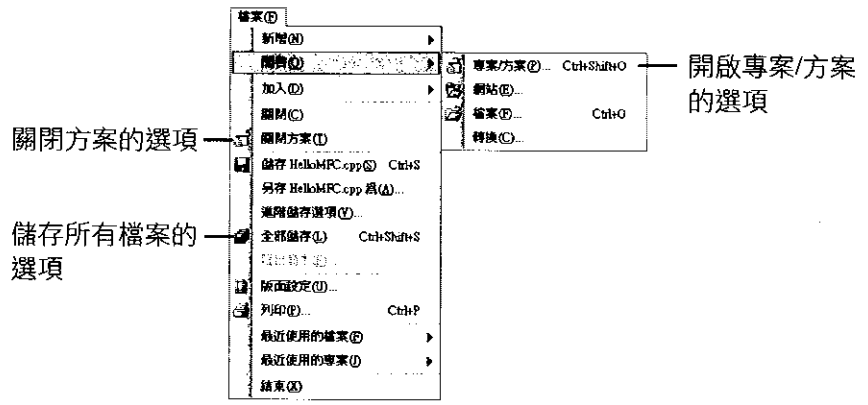
檔案類型	說明
.vcproj 檔	Visual Studio 用於儲存所建立專案資訊的檔案。
.sln 檔	Visual Studio 內用於來儲存方案特定設定值的檔案，將提供方案總管顯示管理檔案的圖形介面所需資訊。建立方案時，將包含專案檔。開啟專案時，請直接開啟此類型檔案。
.cpp 檔	C++程式的原始碼
.h 檔	標頭檔
.rc 檔	資源檔

檔案類型	說明
Debug 資料夾	該資料夾下將儲存編譯過後的可執行檔、機器碼檔，以及程式除錯用的相關檔案。若您設定專案所產生的程式是用於發行用時，則 Release 目錄將會取代此目錄。

3-1-2 專案/方案的開啟、關閉、儲存

第 2 章曾示範過如何建立一個應用程式的專案與方案，在這一節中，則將介紹專案/方案的操作。

檔案功能表的 **[開啟/專案/方案]** 將可選取欲開啟之應用程式方案。欲關閉方案時，則請選取**關閉方案**選項。若選取**關閉**選項，將關閉**編輯區**開啟的檔案。按下**全部儲存**選項將儲存方案內，所有已經修改的檔案。

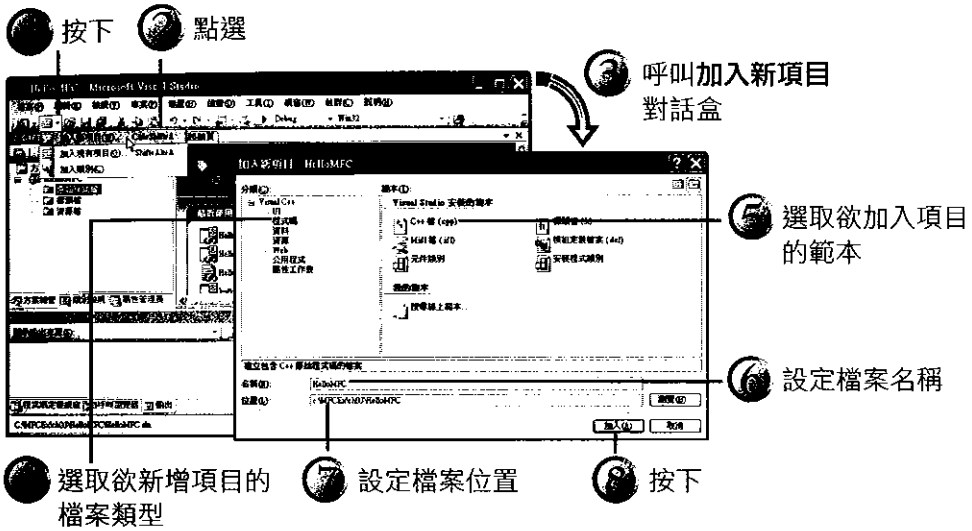


其中**開啟**子功能表的**檔案**選項，將可開啟專案檔的各類型檔案。但請注意！在 Visual Studio 裡，一次僅能開啟一個方案。

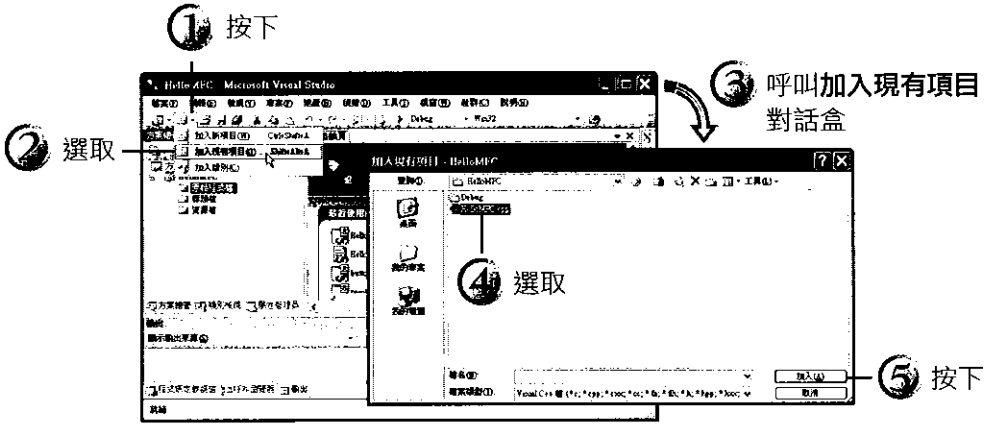
3-1-3 專案中檔案的操作

將檔案新增至專案

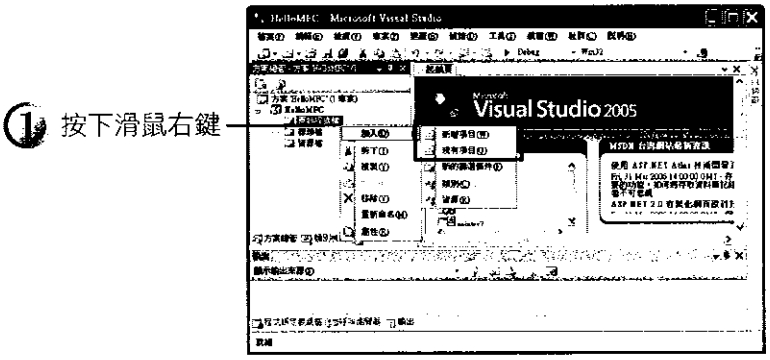
Visual Studio 的檔案操作功能，大致上都在**檔案**功能表與工具列上。欲新增檔案時，可以按下工具列的**加入新項目** 按鈕，並點選**加入新項目**選項，呼叫**加入新項目**對話盒，選取欲新增的項目。這個對話盒可以新增許多類型的檔案，常用的有 **C++檔**（.cpp）、**標頭檔**（.h 檔）、**資源檔**（.rc）...等，如下圖所示。



欲加入已經完成建立的檔案時，請點選**加入現有項目**選項，呼叫**加入現有項目**對話盒。

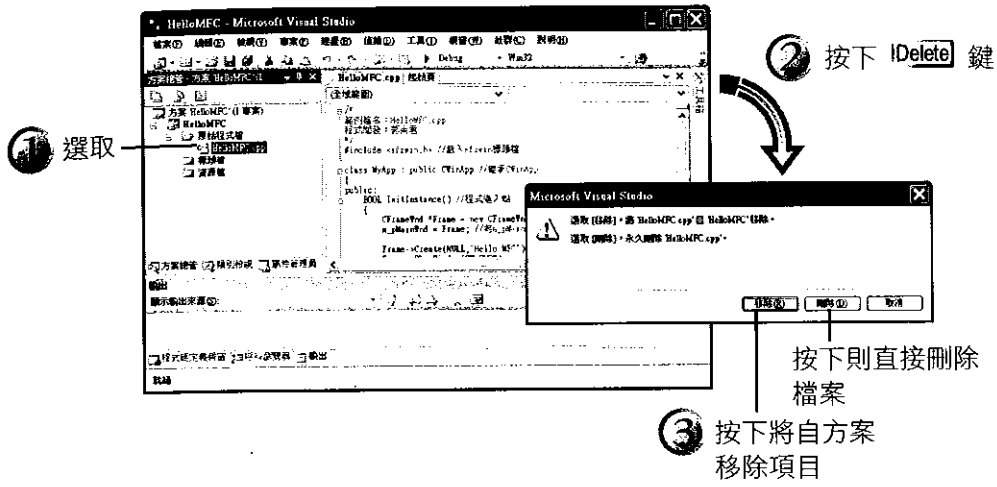


您亦可直接在欲加入項目的資料夾上，按下滑鼠右鍵，選取欲執行動作。



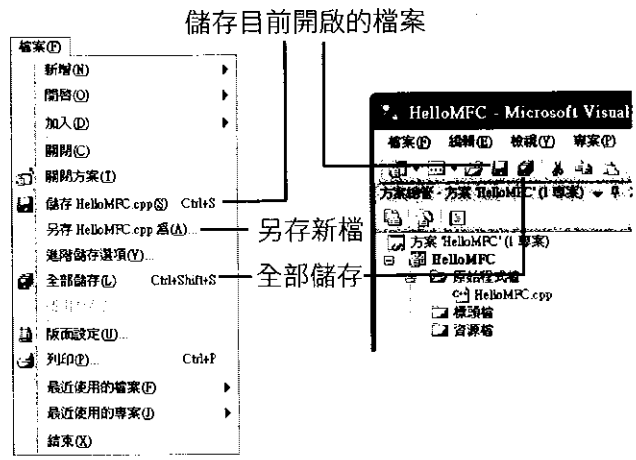
刪除檔案

相較於新增檔案的複雜，將某檔案從專案中刪除的動作就相當簡單，只要將**方案資訊區**切換到**方案總管**視窗下，選取欲刪除檔案後，按下鍵盤的 Delete 按鍵即可刪除檔案。



儲存檔案

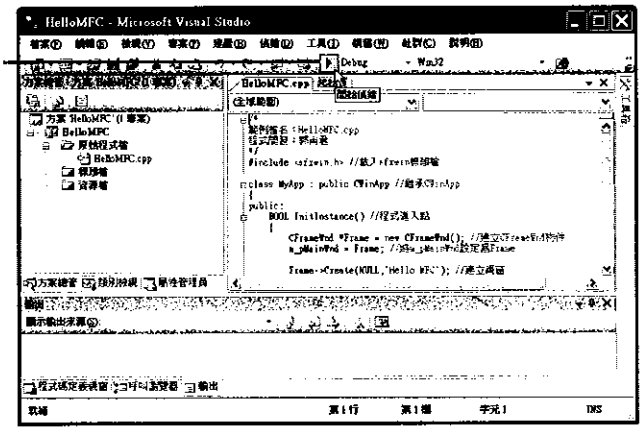
Visual Studio 儲存檔案的方式有三種，一是**儲存單一檔案**，用於儲存目前開啟於編輯區的檔案。二是**另存新檔**，將目前開啟在編輯區的檔案儲存成另一個檔案。三是**儲存所有檔案**，這個命令則將儲存目前所有開啟在 Visual Studio 下的檔案。



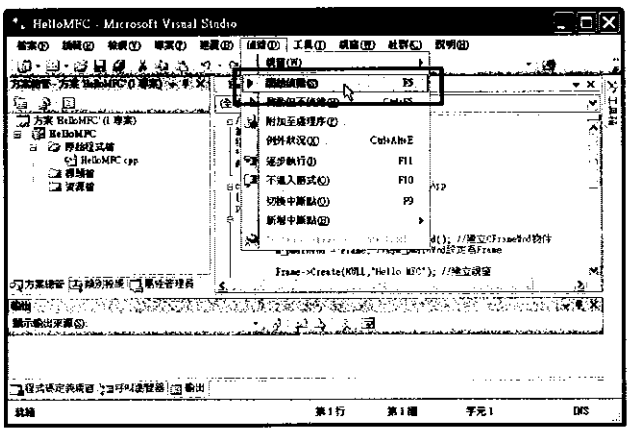
3-2 編譯程式

完成程式的建立後，接著，必須將程式編譯成可執行檔。在 Visual Studio 裡，編譯程式並不難，只要按下工具列的**開始偵錯** ▶ 按鈕即可。

按下執行程式的編譯



或者點選 [偵錯/開始偵錯] 選項，亦可直接按下 F5 按鈕。



若點選**偵錯**功能表的**啟動但不偵錯**選項，則將執行應用程式，但不執行程式的重新編譯與偵錯。

3-3 蟲蟲危機 - 程式的除錯

3-3-1 程式的除錯

天不從人願，是人生的常事。而寫程式時抓抓蟲（除錯），也是很正常的。幸好 Visual Studio 具備相當優越的除錯功能，讓您抓蟲工作能更加順利。

錯誤的種類

撰寫程式時，通常會發生的錯誤，可分為兩類，一是**語法錯誤**，二是**邏輯錯誤**。

所謂**語法錯誤**，就是您用錯的程式語法，像是 if 後面接了兩個 else，或者每行最後忘了加上『;』號...等。這類錯誤在編譯時，Visual Studio 就會把產生錯誤的程式行號，以及錯誤原因顯示在除錯區裡，您只要用滑鼠在錯誤敍述點兩下，**編輯區**即會切換到發生程式錯誤的程式碼。

下圖顯示出 HelloMFC.cpp 檔的第 19 行，出現語法錯誤，您只要在該行用滑鼠左鍵快速點兩下，即可切換至 HelloMFC.cpp 檔的第 19 行檢視該行。

另一種**邏輯錯誤**，則是在執行程式後，發現執行結果不正確。此時，表示程式語法並沒有錯誤，錯誤發生在**演算邏輯**，使得程式的執行結果不如預期，比如：錯誤的邏輯判斷、錯誤的運算過程...等。

相較於語法錯誤，邏輯錯誤的除錯較不容易。除錯方式通常是觀察程式執行的過程，以及執行過程的變數值變化。從變數值的變化，找出錯誤的程式敘述。

因此，以往沒有 Visual Studio 這類程式開發環境時，最常用的除錯方法，就是在程式裡，加入許多輸出函數，輸出程式執行過程中變數的值。藉由觀察變數值的變化，找出程式發生錯誤的原因，這種方式稱為**『插旗子』**。但是有了 Visual Studio 提供的強大除錯功能，並不需要運用這種麻煩的除錯方法。

除錯工具的介紹

對於語法錯誤的除錯，只要依照 Visual Studio 顯示於除錯區的資訊，找出錯誤發生的程式碼，並參考顯示的錯誤訊息修正錯誤，通常大部份的語法錯誤都能輕鬆解決。

對於邏輯錯誤的偵錯，接下來則要告訴您如何利用 Visual Studio 提供的協助工具完成。這部份可以分成兩個方面來說，一個**暫停程式的執行**，二是**除錯環境的使用**。前者可以在程式執行的過程中，將程式的執行**暫停**在程式的某行。後者則是在**暫停程式執行**時，利用 Visual Studio 提供的除錯環境，觀察當時的變數值。請參考以下兩小節的說明。

3-3-2 暫停程式的執行

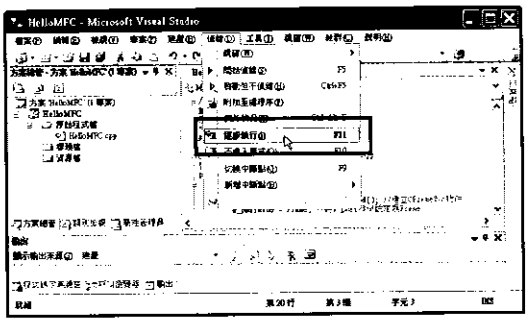
欲瞭解程式執行流程是否如預期時，可以運用 Visual Studio 所提供三種中斷程式執行的方法，檢視程式執行過程。當程式暫停執行欲繼續執行程式，只要再度執行 [**偵錯/開始偵錯**] 或按下**開始偵錯** ▶ 按鈕。

補給小站 暫停程式後，可利用 3-3-3 節的方法，檢視當時程式各變數的值，這些種方法介紹於後。

以下是三種暫停程式執行方法的介紹。

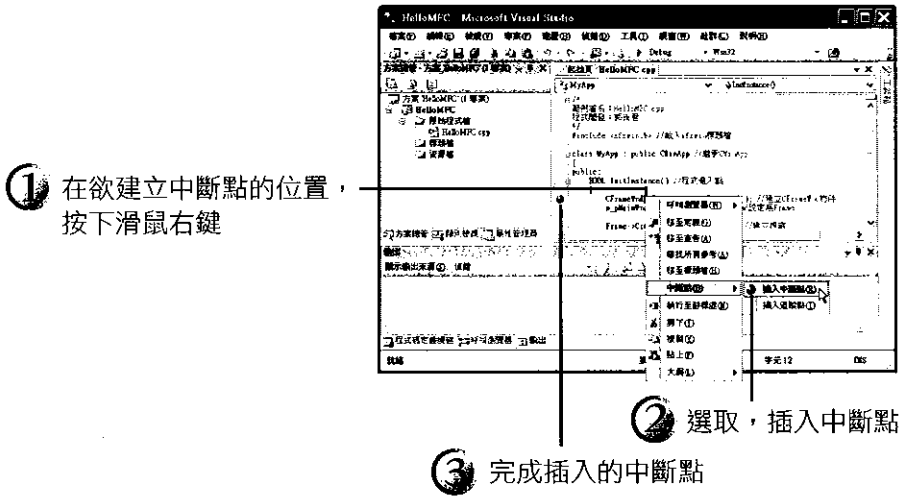
一、逐步執行

逐步執行將提供使用者控制程式的執行過程，協助瞭解整個程式的執行流程。當欲使用這種程式除錯方式時，只要執行 [**偵錯/逐步執行**] 或直接按下鍵盤的 F11 鍵。



二、設定程式中斷點

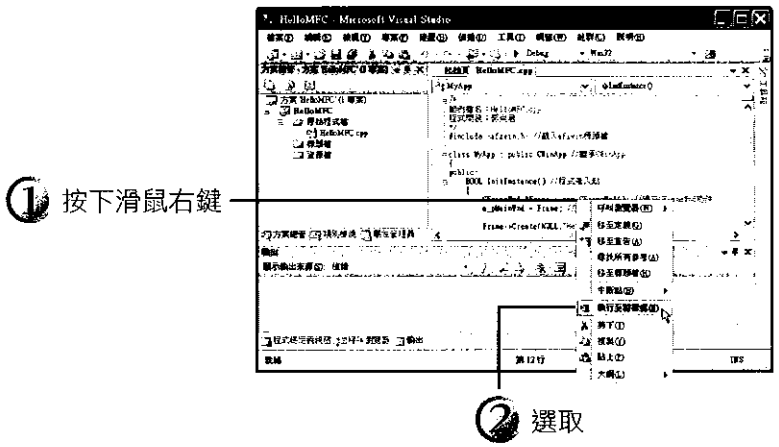
另一種可以瞭解程式執行流程的方式為設定**中斷點**，您可以在程式內幾個比較關鍵的程式碼上，設定程式中斷點，且可設定不止一個中斷點。當程式執行到該中斷點時，程式就會暫停。請在欲建立中斷點的位置上，按下滑鼠右鍵，點選快顯功能表的 [**中斷點/插入中斷點**] 選項，將在游標所在位置的程式碼前，出現代表中斷點的 ●。



標定中斷點後，執行 **[偵錯/開始偵錯]** 或按下工具列的**開始偵錯** ▶ 按鈕後，程式執行至中斷點將暫停。

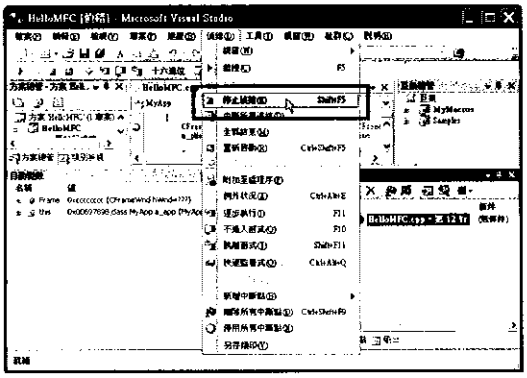
三、執行至游標處

若欲控制程式執行至游標所在位置，請在游標位置按下滑鼠右鍵，點選快顯功能表的**執行至游標處**選項。



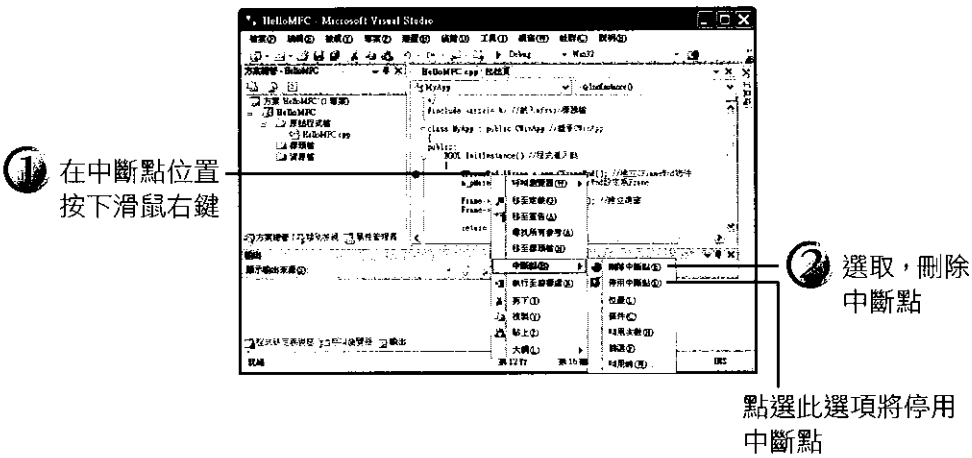
終止除錯

若欲終止除錯時，請點選 [偵錯/停止偵錯] 。



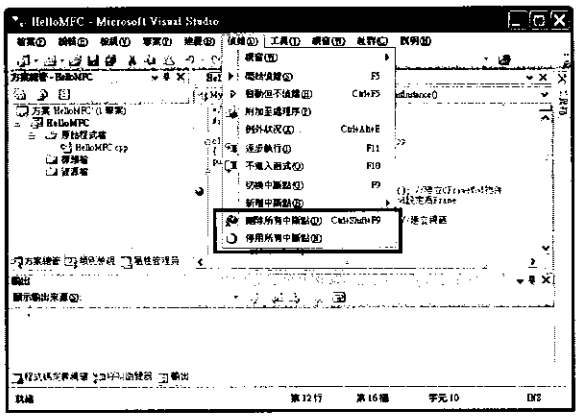
中斷點的刪除與停用

欲清除特定位置的中斷點時，請將滑鼠游標移至中斷點的位置，然後按下右鍵，選取 [中斷點/刪除中斷點] 指令。



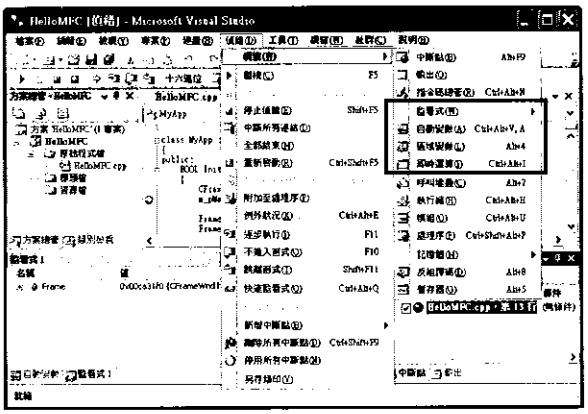
選取**停用中斷點**選項，則將停用中斷點。此時，中斷點的圖示將顯示為 ○。

執行 [偵錯/刪除所有中斷點] 將可刪除程式內設定的中斷點，若執行 [偵錯/停用所有中斷點] 將可停用中斷點。

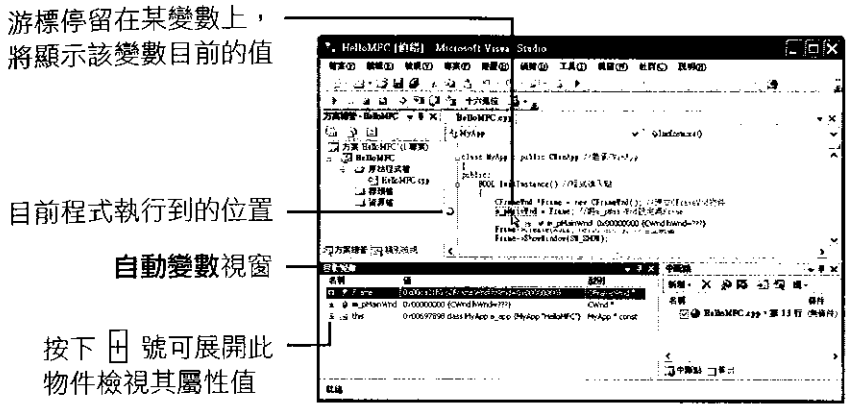


3-3-3 變數值的監看

中斷程式執行時，Visual Studio 將進入程式除錯畫面。在這個畫面下，透過各種偵錯視窗，可以得到程式執行至該行程式碼時，變數值的狀況。欲開啟偵錯視窗時，請點選 [偵錯/視窗] 內的相關選項。

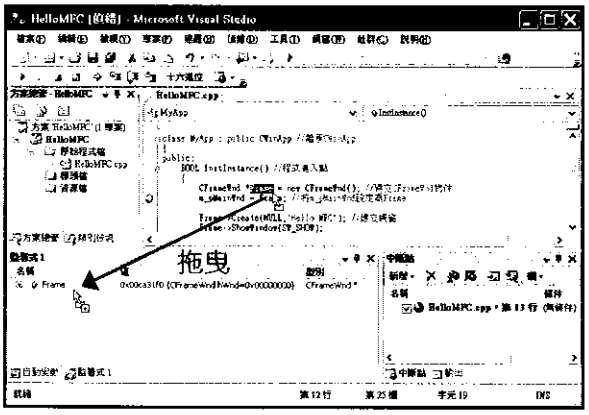


其中**自動變數**視窗將協助我們，得到程式執行至該處時，所有**變數值**。下圖為暫停程式執行時，Visual Studio 的除錯畫面。

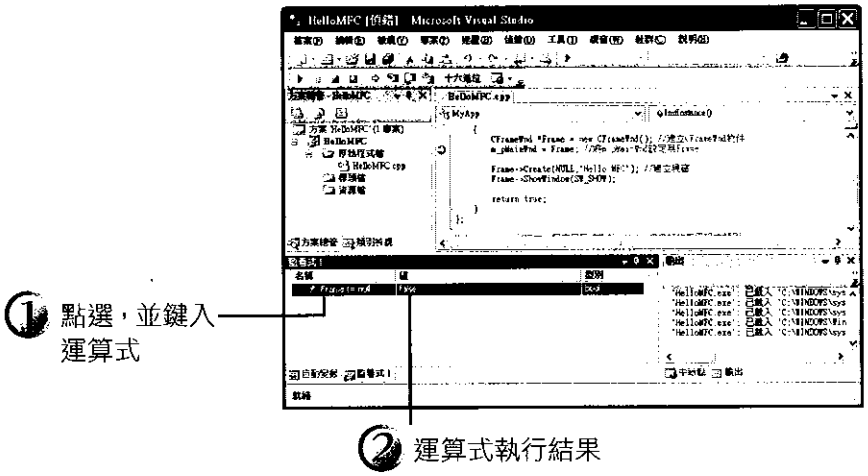


除了透過偵錯視窗檢視程式內變數的值外，您還可以將游標移至任何一個變數上方，停留一下子後，便將顯示該變數目前的值。除錯畫面的上半部，將有一個箭頭顯示目前程式執行到的位置。

若想要監看特定變數或者運算式時，可以點選 [**偵錯/視窗/監看**] 內的選項，開啟**監看式**視窗，在中斷模式下，再用滑鼠選取變數或運算式，拖曳至視窗內。



監看式視窗將不止可以監看變數，還獨監看變數運算後的結果。建立運算式時，請直接用滑鼠點選**監看式**視窗的**名稱欄**，然後編輯欲監看的運算式。

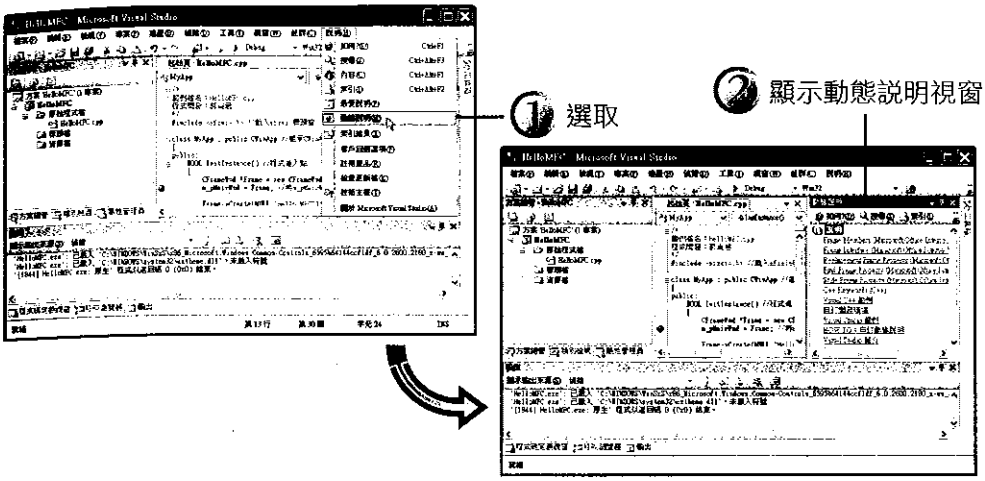


3-4 如何利用線上說明文件

Visual Studio 提供**動態說明**功能，使用者可透過以下三種方式，取得說明資料。

- **內容 – 以主題方式顯示說明文件。**
- **索引 – 透過關鍵字索引的方式，尋找說明文件。**
- **搜尋 – 以使用者輸入的文字，執行搜尋。**

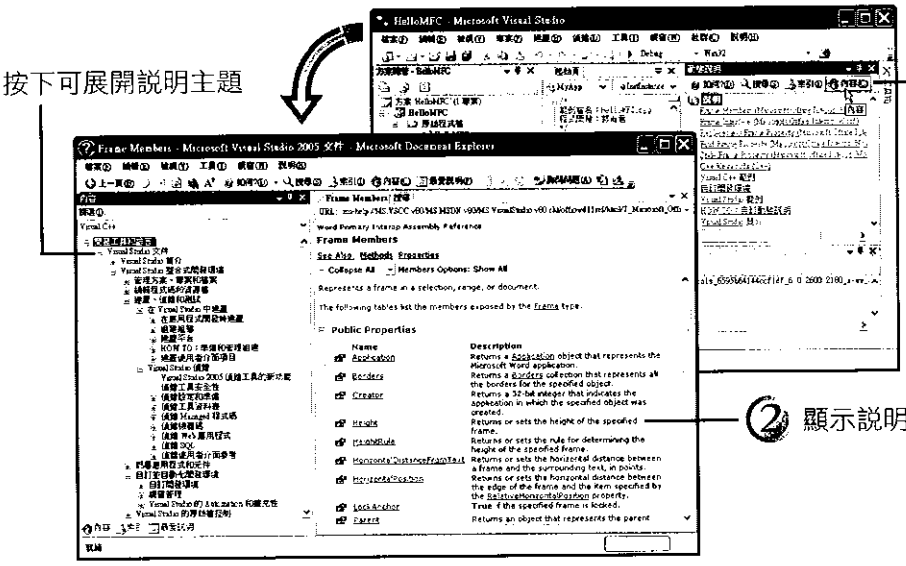
欲尋求動態說明協助時，請點選 **[說明/動態說明]** 選項，Visual Studio 畫面的右半部，將顯示動態說明畫面，點選欲檢視的主題後，將呼叫 Microsoft Document Explorer 視窗顯示說明文件的內容。



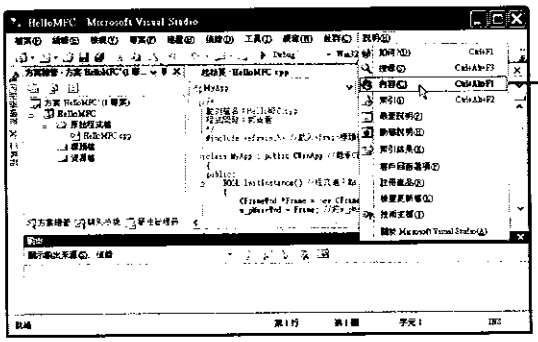
動態說明視窗上方提供三個按鈕，可讓使用者透過三種方式，取得說明文件。

方法一、內容

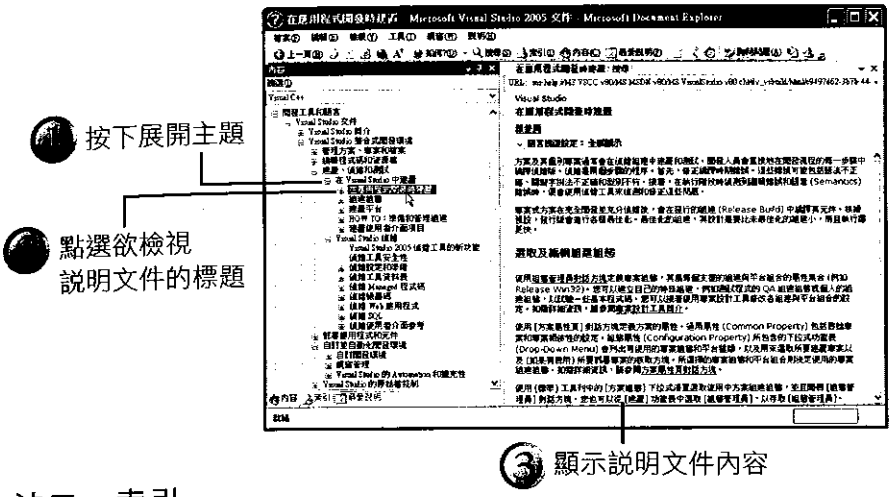
欲以主題分類方式，檢視 Visual Studio 的說明文件時，可按下**動態說明**視窗的**內容** 按鈕，將呼叫 Microsoft Document Explorer 視窗顯示說明文件的視窗。



您亦可點選 [說明/內容] 選項直接呼叫 Microsoft Document Explorer 視窗。

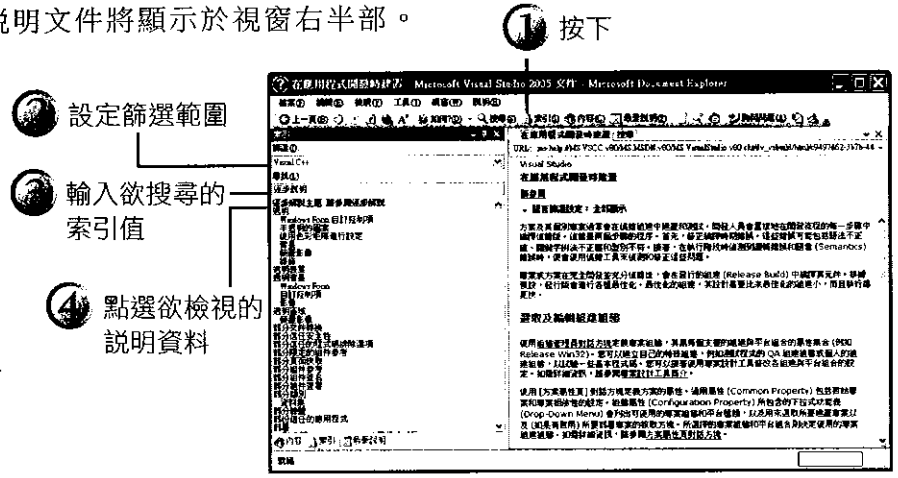


欲檢視某特定說明主題時，請在 Microsoft Document Explorer 視窗左半部的內容子視窗展開說明主題，點選欲檢視說明文件之標題即可。



方法二、索引

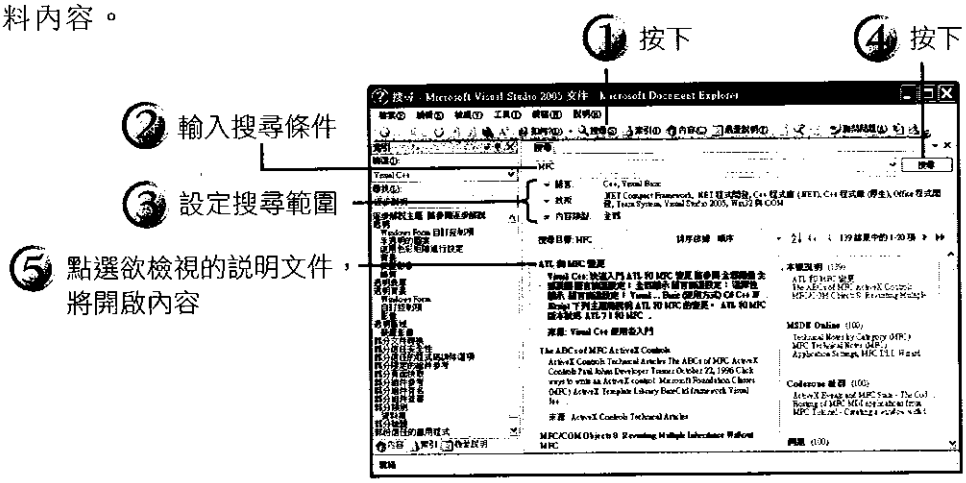
欲透過索引方式檢視說明文件時，請於**動態說明**視窗或 Microsoft Document Explorer 視窗按下**索引** 按鈕。Microsoft Document Explorer 視窗左半部將出現**索引**視窗，即可在**篩選**欄輸入搜尋的資料範圍，於**尋找**欄鍵入欲尋找的索引項目，然後點選尋找到欲檢視的索引資料內容，說明文件將顯示於視窗右半部。



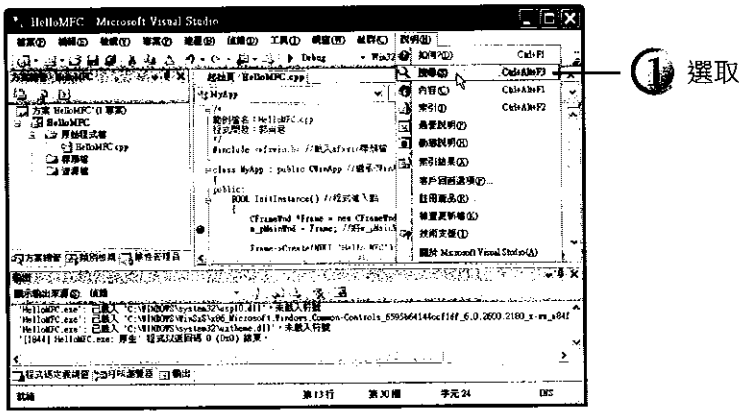
點選 [**說明/索引**] 選項，亦可直接進入 Microsoft Document Explorer 視窗的索引畫面。

方法三、搜尋

欲以特定關鍵字搜尋說明文件內的資料時，請於**動態說明**視窗或 Microsoft Document Explorer 視窗按下**搜尋** 按鈕。Microsoft Document Explorer 視窗的右半部將出現**搜尋**標籤，請設定欲尋找的資料範圍，再於**尋找**欄鍵入欲尋找的索引項目，再點選尋找到欲檢視的索引資料內容。

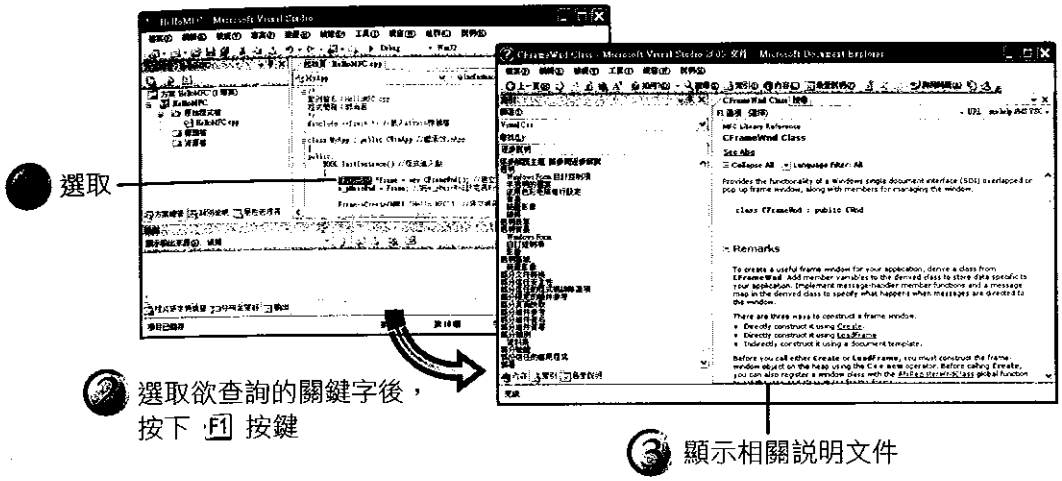


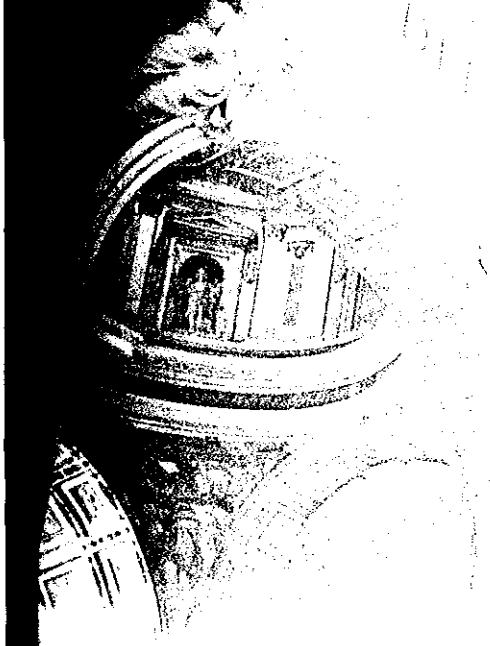
亦可點選 **[說明/搜尋]** ，呼叫搜尋視窗。



快速查詢

若對程式中某一函數、類別或語法有疑問時，可以直接以滑鼠選取該函數、類別或語法，然後按下鍵盤的 F1 按鍵，Visual Studio 將自動尋找出相關說明文件。





Speaking without thinking is shooting without aiming.

–Spanish proverb

不思而言，無的放矢。

–西班牙諺語

4

自訂視窗框架與資源檔的運用

本章導讀

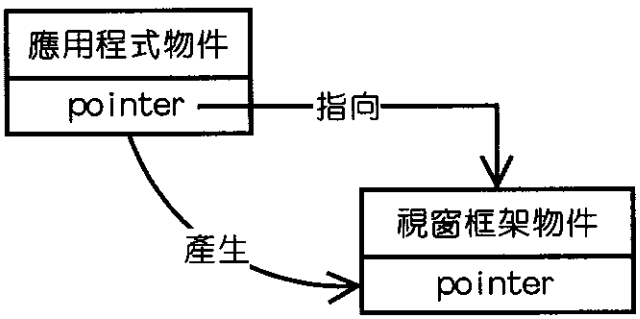
第 1 章曾提到視窗程式設計的資源觀念，但第 2 章的 Hello MFC 程式範例裡，僅建立了一個沒有任何功能的視窗，並沒有使用資源檔。這一章的 MyFrame 程式範例裡，將示範如何運用資源檔，為視窗程式建立一個功能表。

各節標題

4-1 自訂視窗框架物件

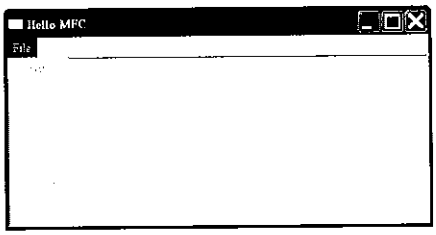
第 2 章的 HelloMFC 程式範例中，僅簡單介紹如何利用 CFrameWnd 類別產生空白的視窗框架，與您一般所看到，具有功能表、工具列...等視窗元件的視窗介面相差甚遠。如何才能建立出具有視窗元件的視窗框架呢？這就必須繼承 CFrameWnd 類別，建立一個自訂的視窗框架類別，並在資源檔定義該視窗框架所需使用的元件，再由視窗框架物件載入這些資源。

這一章的程式範例裡，將建立一個 MyFrame 類別，並在資源檔裡，建立功能表資源，做為視窗框架的功能表。整個 MyFrame 應用程式，在架構上與 HelloMFC 並沒有不同，只是由於欲建立的是自訂視窗框架，所以，並不直接利用 CFrameWnd 類別建立視窗框架物件，而改以繼承的方式，由 CFrameWnd 類別衍生出自訂的 MyFrame 類別，以便修改出所希望建立的視窗框架類別。下圖是 MyFrame 應用程式的架構。



4-2 MyFrame 程式範例

以下為 MyFrame 程式範例的執行結果與程式內容。



請依照第 2 章的方式，運用以下程式碼建立出 MyFrame 程式方案，但請先不要執行，因為還沒有為 MyFrame 建立資源檔。

MyFrame 程式範例 - 自訂視窗框架

檔案位置：MyFrame\MyFrame.cpp

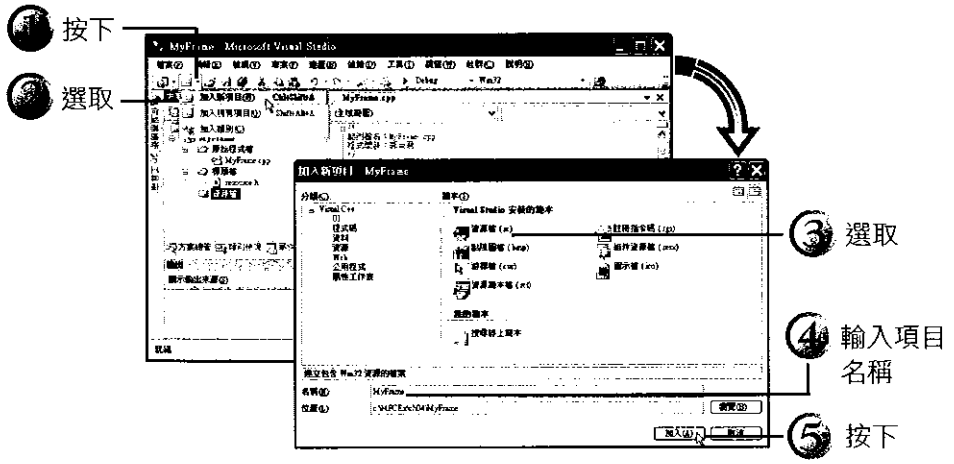
```
1   /*
2   範例檔名：MyFrame.cpp
3   程式開發：郭尚君
4   */
5   #include <afxwin.h>
6   #include "resource.h"  //由資源編輯器所產生的標頭檔
7
8   class MyFrame : public CFrameWnd   //繼承 CFrameWnd 類別
9   {
10  private:
11     CMenu *FMenu;
12  public:
13     MyFrame()
14     {
15        Create(NULL,"Hello MFC");     //建立視窗
16        FMenu = new CMenu;  //產生功能表物件
17        FMenu->LoadMenu(IDR_MENU1);  //載入功能表
18        SetMenu(FMenu);  //設定視窗所使用的功能表
19     }
```

```
20    };
21
22    class MyApp : public CWinApp
23    {
24    public:
25       BOOL InitInstance()  <-- 程式進入點
26       {
27          CFrameWnd *Frame = new MyFrame;    //產生視窗
28          m_pMainWnd = Frame; //將視窗物件設定給應用程式
29
30          Frame->ShowWindow(SW_SHOW);        //顯示視窗
31
32          return true;
33       }
34    };
35
36    MyApp a_app;   //建立應用程式物件
```

接著，請按照以下步驟，為 MyFrame 程式專案建立資源檔。

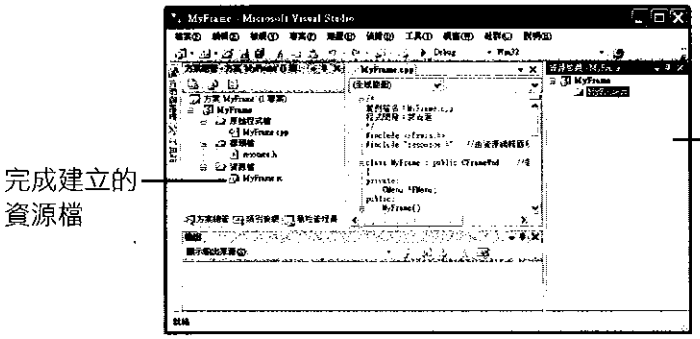
Step 1 新增資源檔

按下工具列的**加入新項目** 按鈕，選取**加入新項目**選項，呼叫**加入新項目**對話盒，於**範本**欄點選**資源檔(.rc)**選項，並於**名稱**欄輸入資源檔名稱為 MyFrame，最後按下 加入 按鈕。



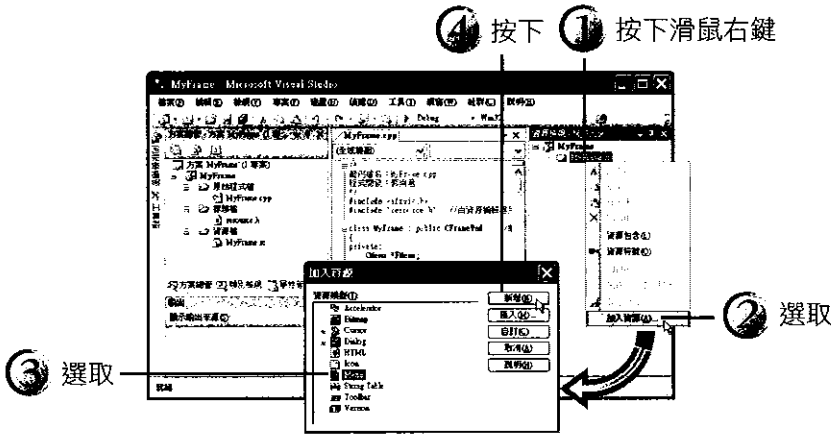
Step 2 完成資源檔的建立

建立資源檔時，將同時產生資源標頭檔。該檔將定義資源檔裡資源物件所使用的識別子...等資料，預設的資源檔檔名為 resource.h。



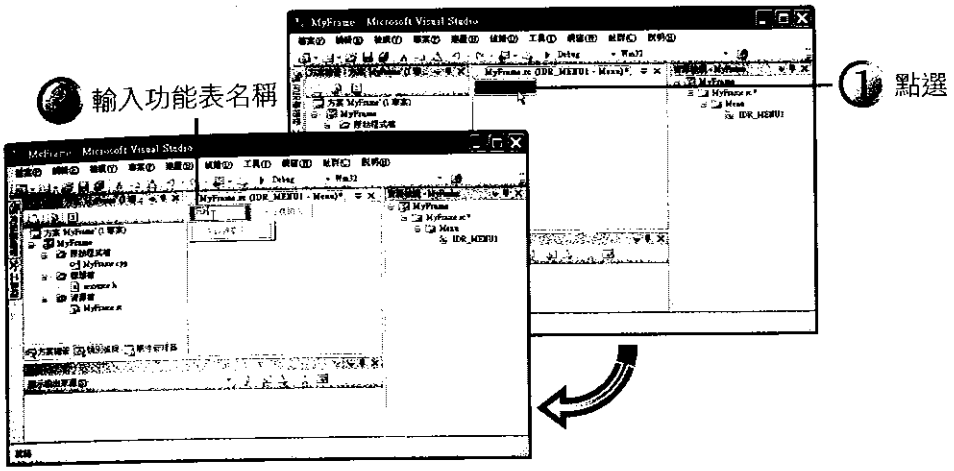
Step 3 插入功能表列物件

於**編輯區**的 MyFrame.rc 圖示上，按下滑鼠右鍵，點選快顯功能表的**加入資源**選項，呼叫**加入資源**對話盒，在**資源類型**欄選取 Menu 選項，按下 新增 按鈕。



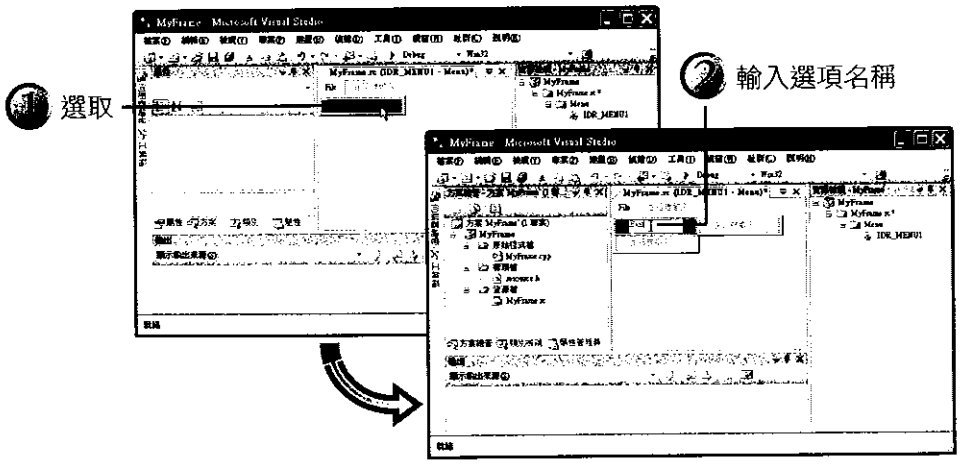
Step 4 建立功能表

加入功能表資源後，編輯區將自動切換為**資源編輯區**，點選空白功能表將可鍵入功能表的名稱 - File。



Step 5 建立選項

點選 File 功能表下的空白選項，輸入 Exit 選項。



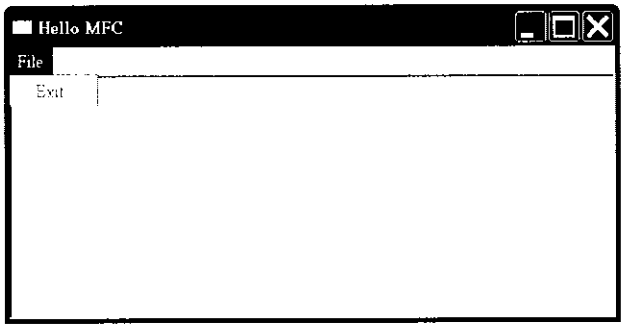
Step 6 選項屬性的設定

在欲檢視屬性的選項上按下滑鼠右鍵，點選快顯功能表的**屬性**選項，並於視窗左半部的**方案資訊區**顯示**屬性**視窗設定 ID 欄為 ID_FILE_EXIT。

補給小站 選項的 ID 欄為當使用者按下選項時，代表傳出值的識別字。

Step 7 編譯程式

按下工具列的**開始偵錯** ▶ 按鈕，即可完成具有功能表的視窗程式。



4-3 自訂視窗框架類別與資源檔

4-3-1 視窗框架物件的建立時機

MyFrame 程式範例雖然是以自訂類別建立視窗框架物件，但 MyFrame 程式範例的架構與 HelloMFC 程式範例其實是一樣的。兩者在程式碼的部份，不同處僅在於建立視窗框架的時機。

HelloMFC 是在 MyApp::InitInstance()函數裡建立，而 MyFrame 程式範例則是將 CFrameWnd::Create()函數改到 MyFrame 類別的建構子中執行。不過，不論執行於 MyApp::InitInstance()函數或 MyFrame 類別的建構子，其實都是一樣的。

此外，MyFrame 類別的建構子除執行 CFrameWnd::Create()建立視窗外，還建立視窗使用的功能表，關於這部份的說明請參考 4-3-3 節。

4-3-2 自訂視窗框架與資源檔的建立

視窗框架與資源檔

由於視窗框架類別負責建立應用程式物件所使用的視窗框架物件，也就是應用程式的視窗介面。而視窗框架物件使用的元件，將定義在資源檔裡。當利用 Visual Studio 開發視窗程式時，可以藉由資源編輯器的協助完成資源的建立，並自動產生定義識別子的標頭檔。在本例中，資源標頭檔的名稱為 resource.h。

定義識別子的標頭檔

程式範例的第 6 行裡，載入的 resource.h 便是由資源編輯器自動產生之標頭檔，該檔由 MyFrame 方案的資源檔（MyFrame.rc）所使用，該標頭檔的設定與產生方式，請參考 4-2 節建立方案的 Step 2 之說明。

```
'摘自 MyFrame\MyFrame.cpp 檔
6     #include "resource.h"  //由資源編輯器所產生的標頭檔
```

resource.h 內將定義功能表資源的識別子，如：代表功能表的 IDR_MENU1，以及功能表選項的 ID_Exit。以下為由 Visual Studio 自動產生的 resource.h 標頭檔內容。

MyFrame 程式範例 - resource.h 檔

檔案位置：MyFrame\resource.h

```
1     //{{NO_DEPENDENCIES}}
2     // Microsoft Visual C++ generated include file.
3     // Used by MyFrame.rc
4     //
5     #define IDR_MENU1                       101
6     #define ID_FILE_EXIT                    102
7
8     // Next default values for new objects
9     //
10    #ifdef APSTUDIO_INVOKED
11    #ifndef APSTUDIO_READONLY_SYMBOLS
12    #define _APS_NEXT_RESOURCE_VALUE        103
13    #define _APS_NEXT_COMMAND_VALUE         40001
14    #define _APS_NEXT_CONTROL_VALUE         1001
15    #define _APS_NEXT_SYMED_VALUE           101
16    #endif
17    #endif
```

4-3-3 於視窗框架建立功能表

當資源檔建立功能表時，要如何將功能表加入視窗框架裡呢？在解答這個問題前，先來談談如何在程式內，操作資源檔的資源。在程式中欲操作資源檔的資源物件時，第一件事就是要在程式裡建立對應的物件，以該物件代表資源檔的資源。當在程式中，欲操作資源檔的資源時，便可透過該物件達到目的。

所以，想要將功能表加到視窗框架時，必須先在程式內建立對應於功能表資源的物件。因此，MyFrame 類別的建構子裡，將建立一個 CMenu 功能表物件，並以 CMenu::LoadMenu()函數為該物件載入功能表資源，再利用 CWnd::SetMenu()函數將功能表物件設定給視窗框架物件。

摘自 MyFrame\MyFrame.cpp 檔

```
16        FMenu = new CMenu;  //產生功能表
17        FMenu->LoadMenu(IDR_MENU1);  //載入功能表
18        SetMenu(FMenu);//設定視窗所使用的功能表
```

BOOL CMenu::LoadMenu(LPCTSTR lpszResourceName)

BOOL CMenu::LoadMenu(UINT nIDResource)

■ **函數說明**

該函數有兩種形式，一為傳入功能表資源名稱，另一為傳入功能表識別子。該函數如果成功載入功能表資源，將傳回非零值，失敗則傳回零值。

◆ **參數說明**

✓ LPCTSTR lpszResourceName
功能表名稱。

✓ UINT nIDResource
功能表的識別子。