

## PowerBuilder 主題文章：

# PowerBuilder 開發 .NET，所需要的 .NET 觀念

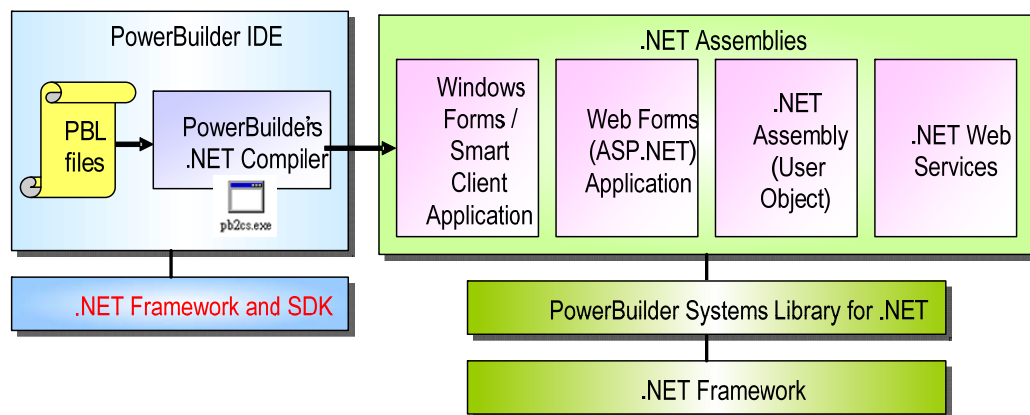
## 前言

如果您有使用 PowerBuilder11 開發 .NET 相關應用的需求，那麼除了 PB 本身的基礎之外，部分 PB 開發中用到的 .NET 觀念也必須有所了解，如此一來不但能夠增進程式開發時設計的寬度，也能在問題發生時讓我們釐清問題。本文將介紹 PB 開發 .NET 相關 Target 後，程式如何在 .NET framework 2.0 運作，並說明有哪些細節是 PB 程式設計師需要注意，以及有哪些工具能夠協助開發，引領 PB 程式技師能夠快速、有信心地開發第一個 .NET 應用。

## PB 開發環境設定與客戶端(Runtime)環境需求

打從 PB11 推出以來，就我們在客戶服務的經驗上，第一次開發 .NET 應用最常遭遇的問題往往都來自環境建置不完全，在過去的 PB 版本安裝上，通常就是執行 PB 原版光碟的安裝流程，將主程式安裝完畢就能開始開發，但 PB11 這個版本如果按照過去的經驗去做安裝，那麼您的 PB11 開發環境將只能撰寫傳統的 C/S 程式，而無法支援 .NET 應用的開發。為了避免這樣的問題，在第一次安裝 PB11 開發環境時，我們希望您能按照接下來的流程，一步步的完成所有程式安裝，假如您發現先前安裝的 PB11 並沒有按照我們建議的流程去建置，那最好也能全部移除後，從頭再進行一次標準的安裝流程，以減少日後不可預期狀況發生的可能性。

在安裝環境前，我們首要了解的是 .NET 應用在 PB11 中開發與部署的架構圖(如下圖)。



這張圖我們可以區分為左右兩個部份來看，左邊(淺藍色系區塊)為 PB IDE 開發環境的架構，右邊(淺綠色系區塊)為部署後的 Runtime 環境架構，由這張架構圖，我們可以了解到開發環境底層必須安裝 .NET Framework 2.0 與 .NET Framework 2.0 SDK，而 Runtime 環境的底層則必須安裝 .NET Framework 2.0 與 PB Systems Library for .NET。

因此，開發端環境安裝步驟如下：

0. IIS(非必要;但如果安裝必須是 IIS 5.0 以上的版本,因此只支援 IIS 4 的 windows 2000 作業系統無法保證.NET 應用的運作)
1. .NET Framework 2.0(必要安裝)
2. .NET Framework SDK 2.0(必要安裝)
3. Microsoft ASP.NET 2.0 AJAX Extensions 1.0
4. ASA10 資料庫(非必要)
5. PB11 主程式(必要安裝)
6. gs860w32(PB11 使用 Saveas()函式列印 PDF 檔案必須安裝)

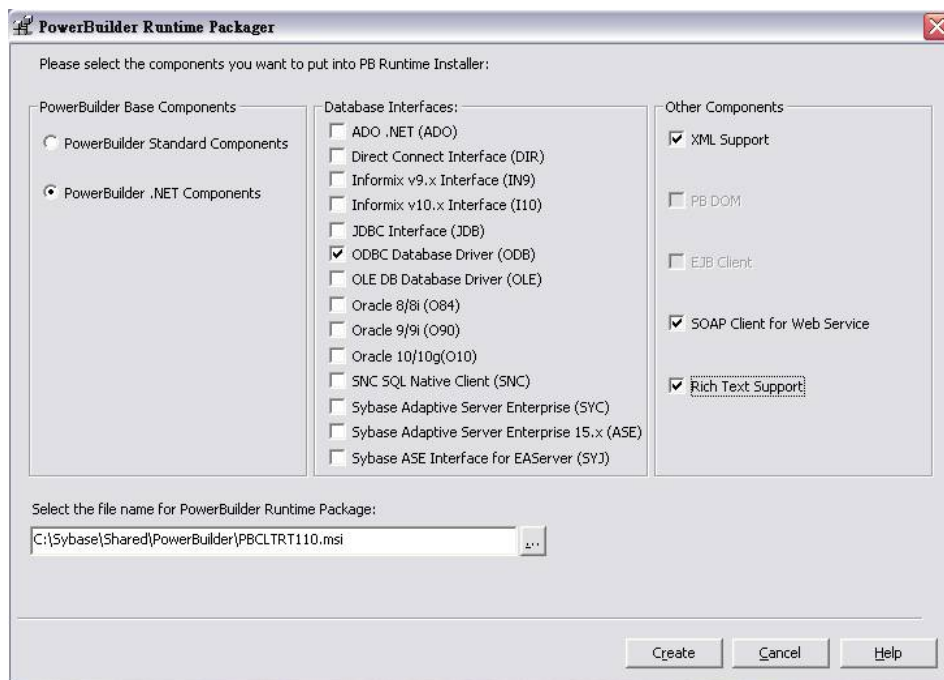
補充：

1. .NET Framework 2.0 與.NET Framework SDK 2.0 需上微軟網站下載安裝
2. Microsoft ASP.NET 2.0 AJAX Extensions 1.0 與 gs860w32 可上本公司網站下載安裝，下載點如下：

<http://www.mpinfo.com.tw/resource.php>;倍力資訊技術資源下載中心

**Runtime 環境**安裝步驟如下：

0. IIS(除非是充當伺服器的主機角色則非必要安裝，注意需僅支援 IIS 5.0 以上的版本，IIS 4 無法保證.NET 應用的運作)
1. .NET Framework 2.0(必要安裝)
2. Microsoft ASP.NET 2.0 AJAX Extensions 1.0
3. PB Systems Library for .NET(可利用 PowerBuilder Runtime Packager 來製作；如下圖)



*May 09 M-Power eNew*

本篇文章版權為倍力資訊股份有限公司所有，未經書面同意，嚴禁複製、轉載

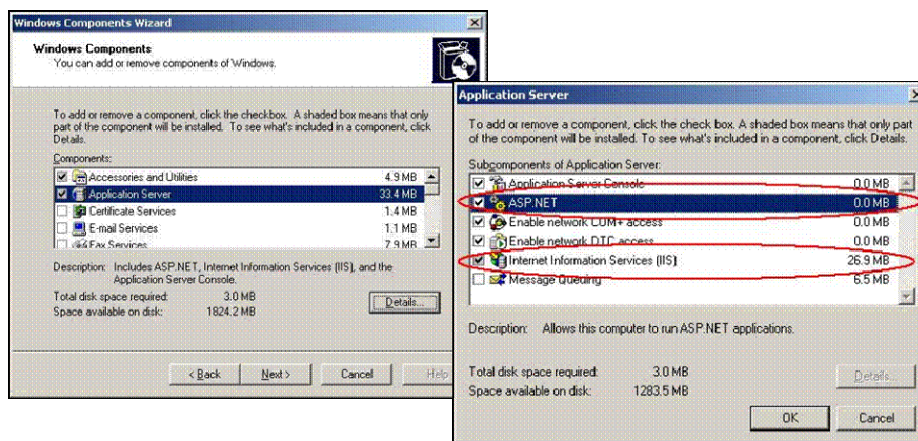
## 環境確認

環境建置完成後我們可以透過下列簡單的檢查去確認安裝是否完成。

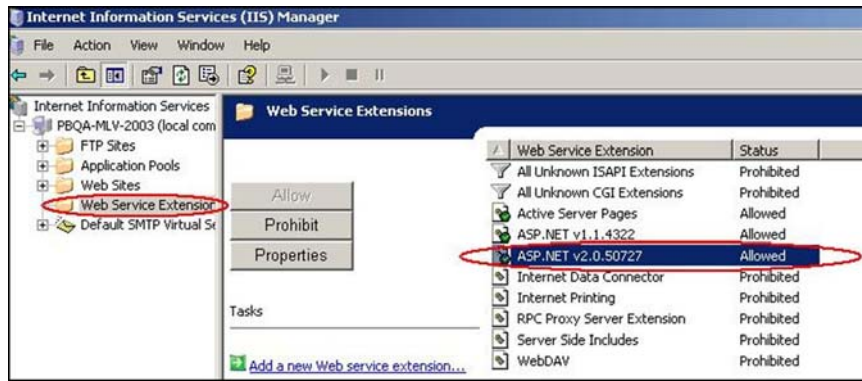
1. 若有安裝 IIS 請確認其中的 ASP.NET 屬性頁籤的版本是 2.0.50727 版，萬一這個頁籤不存在，或者 ASP.NET 版本設定錯誤，都會引發 PB 部署在 IIS 上的 .NET 程式無法正常運作。



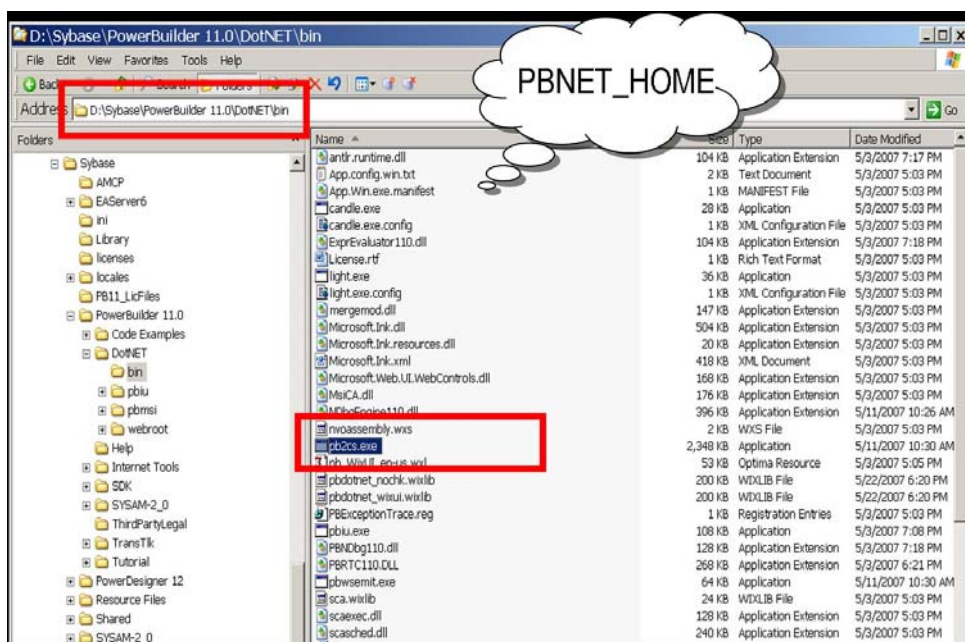
如果您的作業系統是 Windows 2003，則必須注意 IIS 安裝時所勾選的選項(如下圖所示)，請確認 ASP.NET 選項有被選取。



Windows 2003 的 IIS 安裝完成後還需要設定 Web Service Extension(網站服務延伸屬性)的 ASP.NET v2.0.50727 選項狀態為"允許"(如下圖所示)。



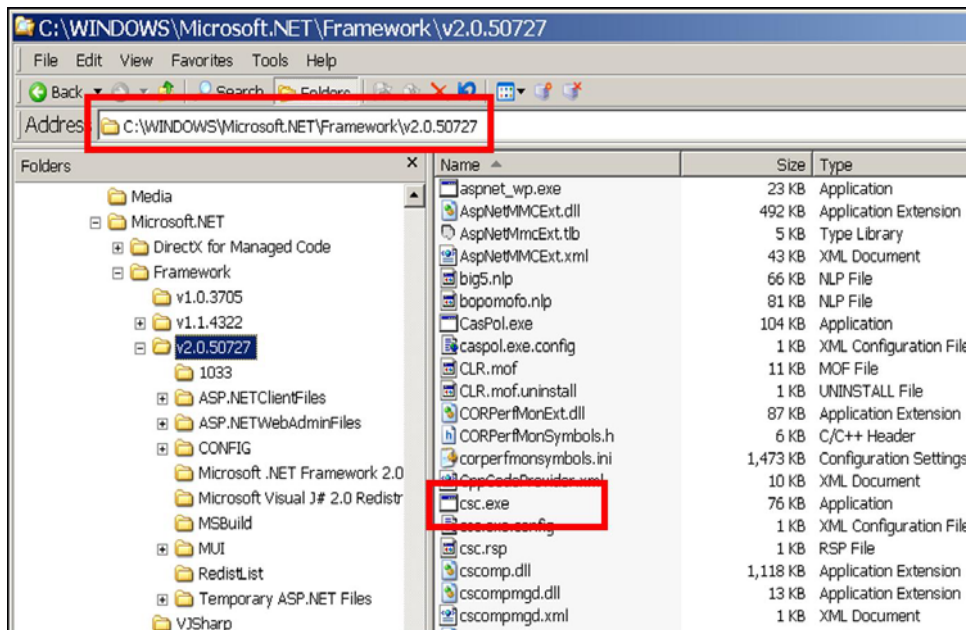
2. **PB2CS Compiler**：我們可以在\\Sybase\PowerBuilder 11.0\DotNET(如下圖)找到這個編譯器，它會負責將 PB scripts 轉譯為 C#的程式碼。



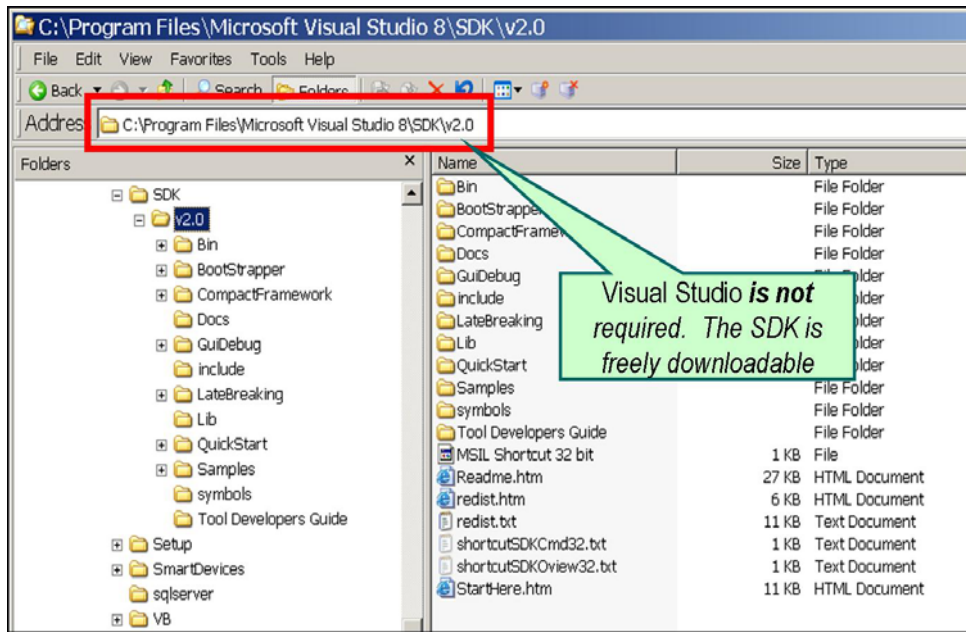
此外我們可以在作業系統的環境變數中看到一個新的變數“PBNET\_HOME”來記錄這個位置(如下圖)。



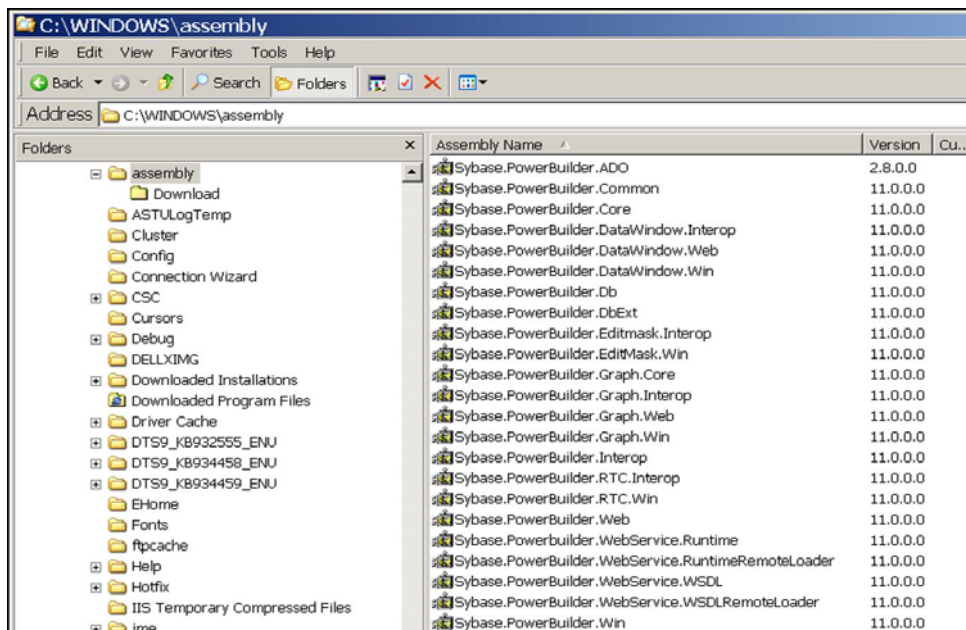
3. **C# compiler**：在 C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727 有一個 csc.exe 檔案，此為 C# 程式碼的編譯器，能夠將 C# 程式碼編譯為 MSIL (Microsoft Intermediate Language；微軟中介語言)



4. **.NET Framework SDK 2.0 目錄**：依使用者安裝的目錄而定，如：[\\Program Files\Microsoft Visual Studio 8\SDK\V2.0](#)，SDK 是 Software Development Kit 的縮寫，中文可以翻譯為軟體開發工具包，是一個能夠協助開發 .NET 應用的文件、程式範例和工具的集合。



5. **GAC 目錄**：GAC 是 Global Assembly Cache 的縮寫，這個目錄提供了所有 .NET 程式 Runtime 執行時所共需的 Libraries 檔案，這個目錄的設計能夠減少程式執行時必須到不同目錄去載入 Libraries 的機會，並有效提高程式執行效能，其中我們必須注意有部分 Sybase.PowerBuilder 開頭的檔案，這些檔案會隨著 PB11 的安裝或 PB Systems Library for .NET 的安裝而進入該目錄，但如果未完整安裝則會造成 PB 所開發的 .NET 相關應用執行時出現錯誤，此時就必須使用 GACutil 工具(在 \\Program Files\Microsoft Visual Studio 8\SDK\v2.0\bin 目錄下)來手動安裝(PB GAC 檔案安裝不完全的情形通常在 Vista 電腦才會發生)。



## .NET 觀念簡介

利用 PB11 開發 .NET 雖然不必真的碰觸到 .NET 的程式碼(如 C#)，但 PB 程式設計師如果能掌握 .NET 程式架構的精神，則開發上就能收事半功倍之效，接下來我們簡短的介紹一些相關的 .NET 常識。

### 1. 何謂 .NET??

一般來說，.NET 就是 .NET 應用的簡稱，而 .NET 應用必須運作在 .NET Framework 上，所謂的 .NET Framework 本身是共通的、標準的『程式庫』，這個標準的程式庫上能夠提供不同的程式語言設計師來開發各式各樣的應用，同時也是一個標準的應用程式運作環境(如下圖所示；請注意這些各式各樣的應用中除了 Web 類型程式通常被另外稱作 ASP .NET 外，其他能夠運作在 .NET 環境上的程式，都可統稱為 .NET 應用) 目前 PB 支援的 .NET Framework 版本為 2.0。

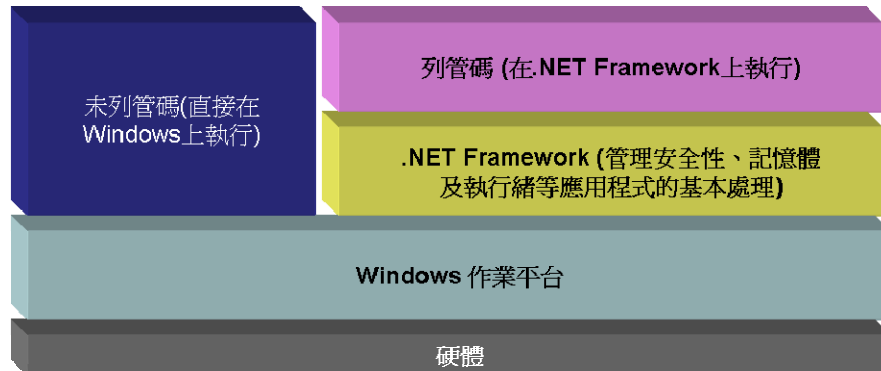


一個標準的程式庫能有什麼作用呢？它主要的功能是提供程式設計師應用程式開發的基礎，並讓程式能夠執行在一個豐富、穩定高效率的執行環境。 .NET 的程式設計師不必像以往開發 System32 的程式那樣，凡事都要自己來，透過 .NET Framework 所提供的標準類別庫，程式設計師無論在 UI 介面設計、檔案管理、資料庫存取上都只要呼叫、實作這些現成的標準類別就能達到目的，更重要的是因為這個 Framework 是全球統一的標準，一來它能解決過去程式部署時總要自行加入許多 .dll 檔問題，這些數量眾多的 dll 檔一旦有版本引用錯誤的情形，就會造成程式運作錯誤(被戲稱為 DLL Hell 問題)，二來，另一個好處是只要您的程式碼是運作在 .NET Framework 上，則站在程式碼重複使用與 Library 檔案交互使用上相容性是非常高的。

### 2. 何謂 Managed Code 與 Assembly??

- i. 過去 System32 的 \*.dll 檔或 \*.exe 執行檔，在 .NET 問世後被稱做為 unManaged

Code，而能夠在.NET Framework 上運作的程式碼則被賦予另一個名稱“Managed Code”(列管碼)，這樣的命名原理非常簡單，純粹以能不能被.NET Framework 託管來分類(如下圖所示)。

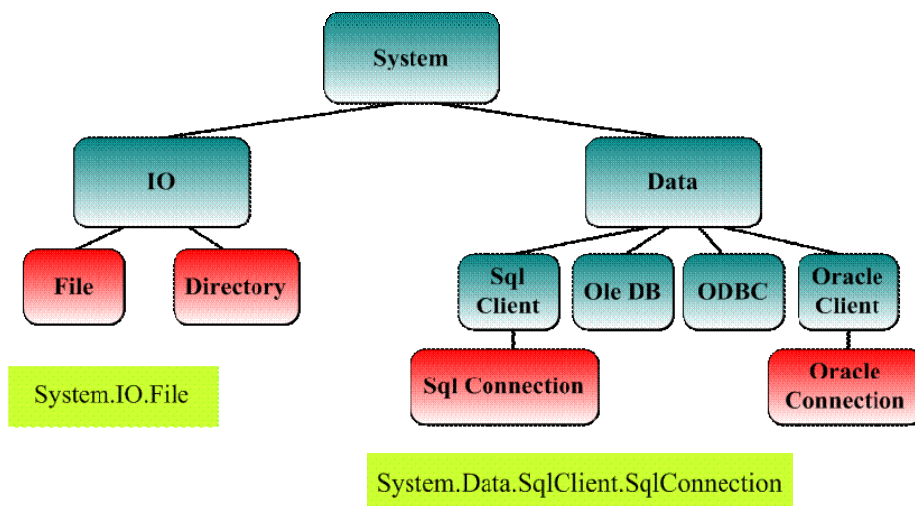



ii. Assembly(組件)，所有被編譯成\*.dll 或\*.exe 的.NET 檔案都叫做 Assembly。

### 3. 何謂 Namespace??

以往的 Win32 應用下，所有的類別、方法都散佈在不同的 dll 檔中，使用者使用時，面對的是一大堆的 API，既沒有依功能分類，更讓人不知道要完成某個需求時，需要先呼叫哪個函式再呼叫哪個函式，往往順序錯誤便造成系統當機。也不能透過繼承的方式加以修改，依專案需求進行改寫或擴充。基於以上缺點，.NET Framework 內部便使用階層式名稱空間(Namespace)將所有的類別分門別類。

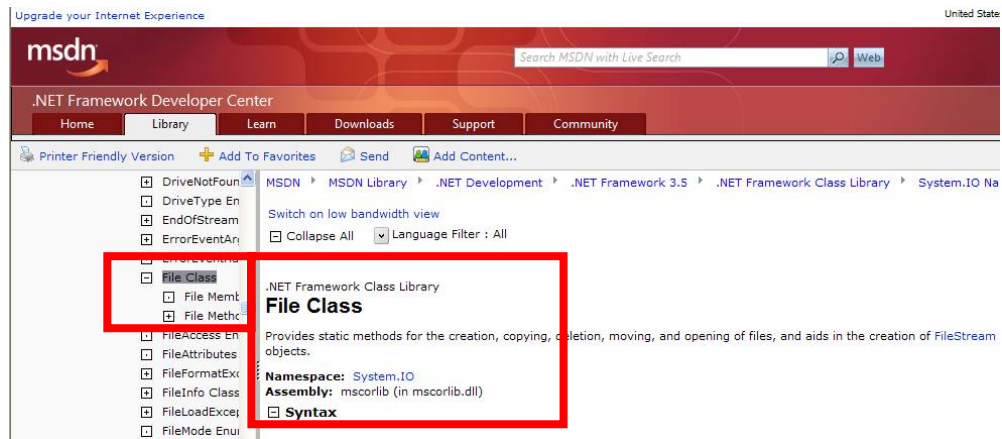
物件導向架構的.NET 中有一個很重要的觀念，.NET 中最基礎的單位是 Class，而整個命名空間的樹根是“System”，因此我們在引用某一個屬性或方法時，它必定是存在於某一個命名空間的某個類別之下，引用時必須以完整名稱來描述它，如：當我們要在程式中將文字輸出，則我們必須依照命名空間的樹狀階層架構規則，去呼叫 File class 下的方法，如：System.IO.File.OpenText(如下圖所示)。



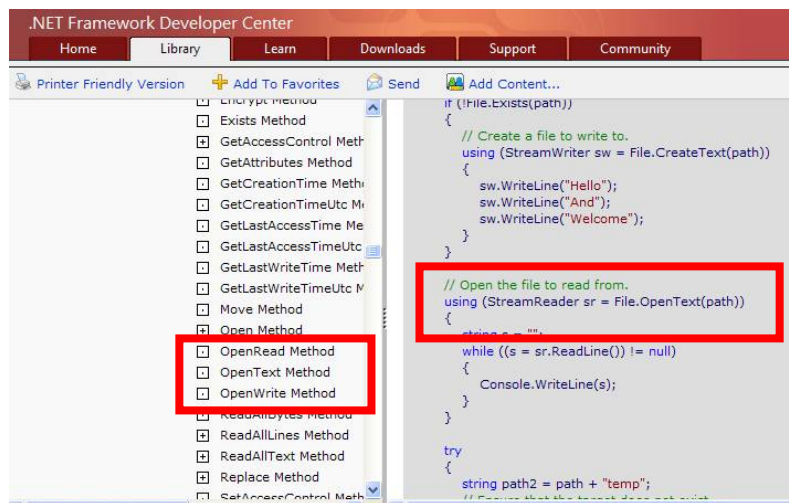
 :代表Class

.NET 基本單位：Class

但要如何得知這個類別被存放在哪一個 assembly 之中呢？我們可以查詢 MSDN 網站上的資訊(或上 google 以關鍵字查詢，如：System.IO.File)，我們可以在 MSDN 上找到 File Class 是屬於 System.IO 命名空間以及存放於 mscorlib.dll assembly 之中(如下圖右方紅色方框所標示)。



接下來我們可以展開 File Class 下的 method，並且觀察其中的程式碼範例，也許您會覺得奇怪，我們明明在說明 PB 的開發，為什麼要參考 MSDN 中的程式碼範例？這是因為當我們要在 PB 的 .NET 相關 Target 中去引用這些方法時，我們可以使用類似的方式去建立並使用這些 method 與屬性。



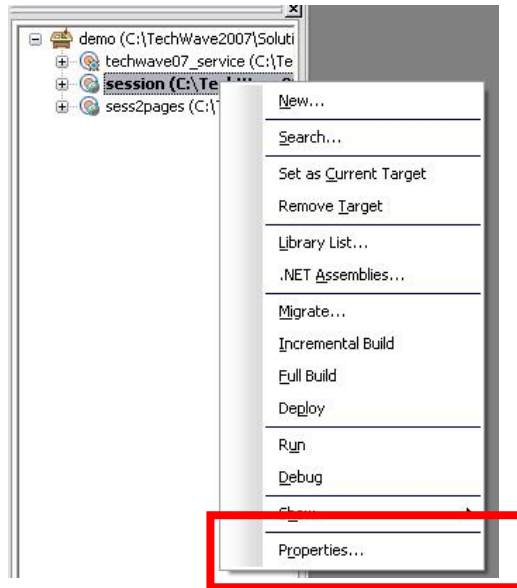
4. 綜合上述的觀念，現在我們使用一個簡單的範例來說明如何在 PB11 中引用這些 Assembly 檔，透過 .NET Framework 所提供的類別、函式、屬性來延伸 PB 的開發工作。

#### [範例說明]

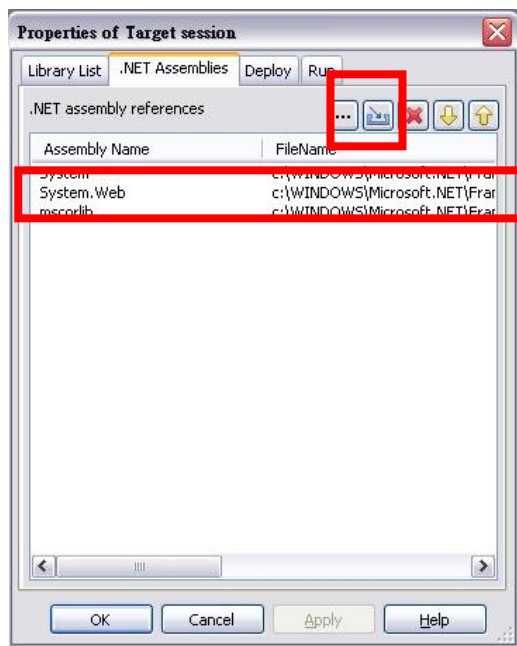
我們展示的範例是以 PB 開發一個 ASP .NET 網頁，當使用者開啓該頁面時可以取得當時的 SessionID 序號。

- i. 首先我們要加入我們可能會使用到的 Assembly：請確認關閉所有的工作視窗，

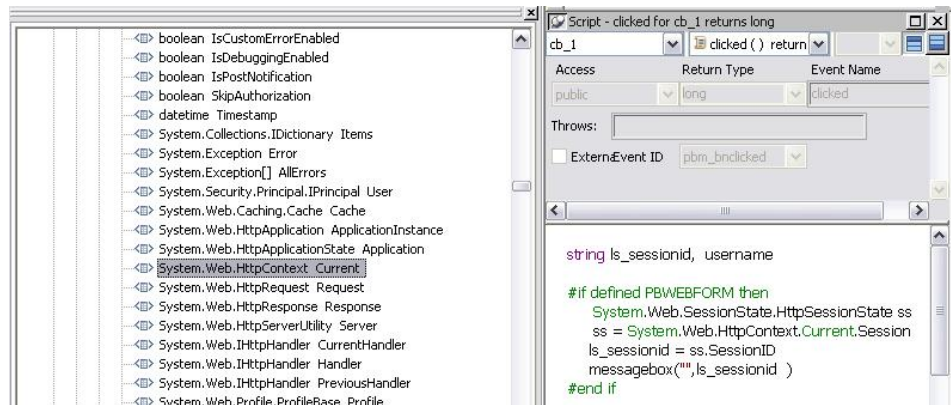
在沒有任何 painter 開啓的前提下，以滑鼠右鍵點選目標 Target(新增.NET WebForm Target)，並選得選單中的 Properties 屬性，如下圖紅框所示。



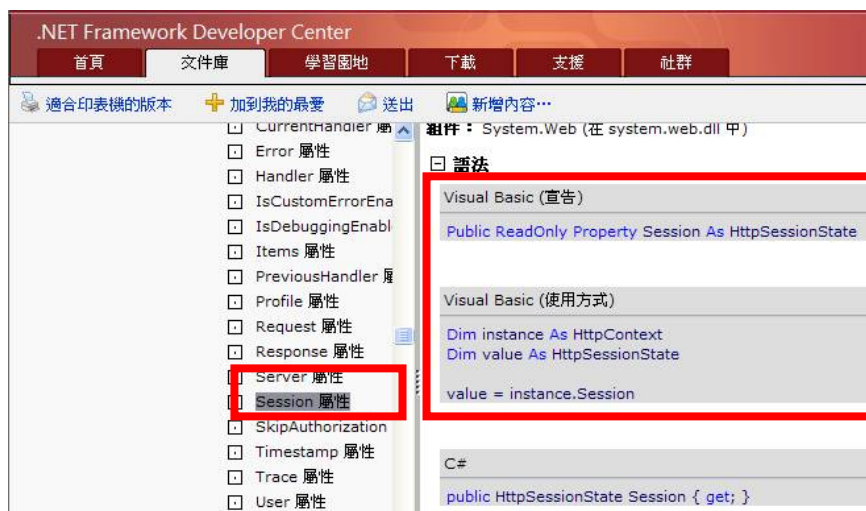
- ii. 設定參考的 Assembly：接下來依下圖紅框所標示的功能鈕，進入後選擇 System.Web.dll 檔。



- iii. 引用成功後，我們就能夠在 PB 的 system tree 中找到該 Assembly 物件，展開後我們便能看到該類別下的所有 methods 與屬性成員。



爲了瞭解如何使用這些方法與屬性，我們可以查詢 MSDN，.NET Framework 下的 `HttpContext.Session` 屬性與 `System.Web.SessionState` class 下的 `SessionID` 屬性，值得注意的是 MSDN 中所提供的 Visual Basic 程式碼範例會比較接近我們要寫在 PB condition compilation area 中的程式碼，因此我們可以參考 VB 的範例下去修改。



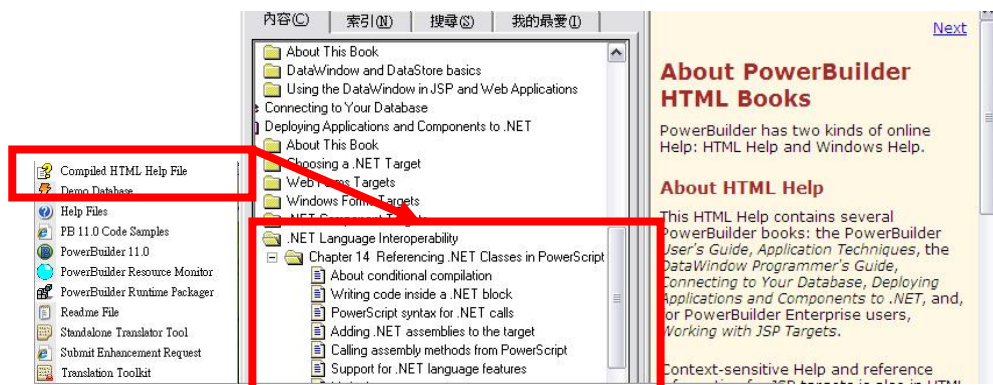


- iv. 我們可以快速新增一個 Window、一個 Command Button，並將程式碼寫在 Command Button 的 clicked event 之中，程式碼如下，程式儲存部署完成後便可以直接執行。

```

1. string ls_sessionid, username
2. #if defined PBWEBFORM then
3.     System.Web.SessionState.HttpSessionState ss
4.     ss = System.Web.HttpContext.Current.Session
5.     ls_sessionid = ss.SessionID
6.     messagebox("",ls_sessionid )
7. #end if
    
```

在此我們並不作 PB condition compilation area 中程式碼開發的詳細介紹，如果您有興趣可以查詢 Compiled HTML Help File 中的 .NET Language Interoperability 章節(如下圖所示)。

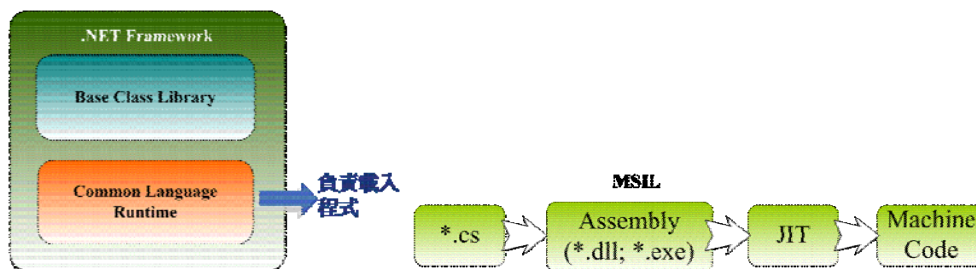


## 5. 何謂 CLR 與何謂 JIT??

在 .NET 的架構中，程式編譯完成後的 Assembly 檔並不能立刻拿來執行，因為此時的程式碼並非可執行的機械碼而是我們先前提到的中介語言格式(MSIL)，當第一次執

行時，.NET Framework 底層 CLR(標準語言執行環境；Common Language Runtime：如下左圖所示)中的 JIT(Just In Time 即時編譯器)才會動態載入檔案，進行編譯為機械碼的動作後，然後才真正開始執行。

這樣的設計主要是為因應跨平台的需求，當中介語言的 Assembly 檔案被移植到不同的作業系統上，只要透過相容的 CIL 動態編譯，程式就可以不經任何修改達到跨平台的目的，JIT 運作的機制如下右圖。



Common Language Runtime 除了提供 JIT 的編譯功能外，它主要的角色就是 Visual Machine(虛擬機器)，提供所有不同語言一個標準的執行環境，其他像是 Memory GC、例外捕捉及處理、型別檢查等也都由這個執行環境來提供。

但也因 JIT 機制，PB 程式設計師可能會發現開發 .NET WebForm (ASP .NET) 應用於第一次執行時有執行速度緩慢的問題，這是正常的現象，解決的方式可以用 .NET Framework 所提供的 Aspnet\_compiler.exe 來解決。

## 結語

有人曾提出疑問 PB11 去開發 .NET 應用實質上到底有什麼好處，我想最顯而易見的就是 .NET 對於許多公認標準支援的更新非常快速，因此 PB 開發出來的 .NET 程式碼也就能夠間接地對這些標準支援，在這個系統與系統間溝通的方式越來越多也越來越頻繁的時代，無法即時支援這些標準的系統往往也就表示將成爲一個無法與其他技術溝通的技術孤島，過去的 PB 在特定應用的開發上雖然快速，但在開發延伸性應用上往往爲人所詬病，現在回歸到 PB 工具快速開發的基本理念，加上 .NET 技術爲 PB 開發的延伸開啓了一扇大門，產生的威力不可言喻。

透過這篇文章的介紹，我們可以發現 PB11 開發 .NET 應用並沒有使用到多麼高深的 .NET 技術，但這些簡單的概念卻已足夠讓 PB 程式設計師開發出許多 .NET 延伸的應用來，如果您在您的工作上遇到了一些需要大量整合外部應用的功能需求，或者您正嘗試開發出一個支援多種標準協同運作極爲複雜的大型專案，不妨試試讓 PB 爲您減少曠時費日的開發工作。