十分鐘了解.NET

這一篇將介紹.NET 的觀念，期許不了解.NET 的 PowerBuilder 開發人員，能有快速的了解。內容包含如下：

- .NET Framework架構
- CLS、MSIL和CLR
- Managed和Unmanaged code
- CTS(Common Type System)
- JIT Compiler、Assembly和Metadata

.NET Framework 架構

我們先用宏觀的角度來介紹.NET Framework 架構，然後以此概念爲基礎，再說明每個專有名詞，相信大家對.NET Framework 有更明確的概念。在這裡筆者所談，先以.NET Framework 2.0 內容爲主。

首先，就以程式語言的角度說明，.NET Framework 包含三大類別程式庫，分別是 ADO.NET 類別、基本類別和 Framework 類別。這三種類別各有其職責，如下表所示：

類別	功能
基本類別	所有程式必要的**基本功能**，例如 IO、Security、記憶體、Collections…等等的類別。
ADO.NET 類別	職司**資料處理**的程式庫，例如資料庫或是 XML 資料。
Framework 類別	提供應用程式**使用者介面**或是**網路服務**等類別，例如 WinForm、WebForm、Web Service…等等。

表 1.NET Framework 2.0 的類別程式庫

本篇文章版權爲倍力資訊股份有限公司所有，未經書面同意，嚴禁複製、轉載

以圖形表示而言，大致就如圖 1 所示：

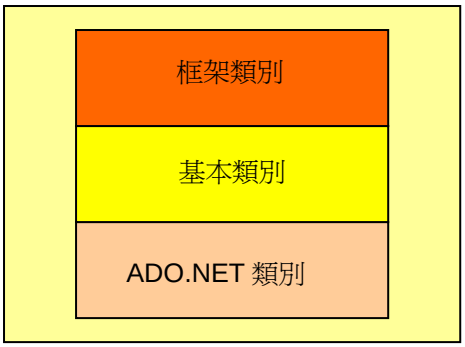

圖 1 .NET Framework 類別庫

圖 1 是以程式類別庫的架構來說明，因爲這是程式設計師最直接遇到.NET Framework 的第一線。對 PB 的開發人員，了解這些程式庫將有實質上的幫助，假如你的程式需要一些 PB 無法做到的功能，那這些程式庫也許可協助你完成。對於完全沒碰過.NET Framework 的 PB 開發人員也不用太擔心，因爲就大部分系統而言，PB 的功能即可滿足，所以了解這些程式庫就不具急迫性了。

CLS 和 MSIL 和 CLR

.NET 號稱可以跨語言，是有其技術背景的。就好像如 Java 號稱可以跨平台，其技術背景是因爲 Sun 提供各種平台的 Java 虛擬機器(JVM，Java Virtual Machine)，Java 語言運作在虛擬機器上，虛擬機器再運行在硬體平台上。這樣 Java 就隔絕了硬體，假如硬體有任何問題，都由 Java 虛擬機器來應付即可。

所謂 Java 虛擬機器，就是指使用軟體來模擬硬體機器的行爲；對 Java 而言，Java 虛擬機器就像是真實的機器，提供了 Java 執行的環境。

那跨語言呢？由於各種語言有其自己的定義和特性，所以在整合上有其困難度。例如，以 PowerBuilder 而言，一個 Integer 資料型態，長度是兩個 bytes，至於 C++，光是整數資料型態的類型就有數種，最高甚至可支援到 64 個位元；又例如有的程式支援物件導向的特性，有的程式無法支援；有的程式可以支援例外處理的機制，有的程式不行。諸如這些相異點，都是造成程式彼此間呼叫和共享的最大障礙，所以很多的程式碼無法重複使用(Reuse)，只能被同類型程式呼叫。

本篇文章版權爲倍力資訊股份有限公司所有，未經書面同意，嚴禁複製、轉載

基於這樣的理由，要讓不同的程式碼可以有機會共享，首先大家要有共同的標準，只要有了標準，就革除了障礙，如同 XML 語言般，因爲資料交換格式的標準，促進資料共容易地交換。而.NET 的架構，針對語言的標準，所提出來的就是 CLS(Common Language Specification)。

CLS 是一種語言的規範，依據微軟.NET 架構的精神，只要語言能夠符合 CLS 的規範，彼此就可以互相叫用，就做到所謂跨語言的目的。問題是要如何產出符合 CLS 的程式呢？以 VB 而言，它的語法顯然不支援.NET(因爲 VB 比.NET 老早就出現在市面上了)，ASP 同樣的也不支援。那答案是什麼呢?就是需要符合 CLS 規範的語言編譯器。

凡是符合 CLS 的語言編譯器，都會把程式編譯成另一種語言，叫做 MSIL(Microsoft InterMediate Language)，它算是一種中介語言，因爲 MSIL 最終還是需要一個環境執行，MSIL 本身不是原生碼，機器看不懂他，所以 MSIL 只是暫時的中介語言。讓我們想一下，假如所有的語言，經過編譯之後，全部都產生 MSIL 語言，那她們就有了共同的標準，可以互相呼叫，這就做到了跨語言。所以，把 VB 語言經過符合 CLS 的編譯器來編譯，這樣的語言就稱之爲 VB.NET，把 ASP 語言經過符合 CLS 的編譯器來編譯，這樣的語言就稱之爲 ASP.NET。當然這樣的說法只是爲了描述概念，事實上以程式而言，VB 和 VB.NET 程式還是有所不同，不是把 VB 加上 CLS 編譯器就是 VB.NET 那樣的簡單。例如 VB.NET 程式本身增進更多功能，更符合物件導向語言，光是這點就與 VB 差異很多了。

前段說到 MSIL，它是 CLS 編譯器產生的結果，是一種中介語言，無法執行。那問題來了，MSIL 到底要如何執行？答案就是 MSIL 必須要在一個虛擬機器上執行，那個虛擬機器就叫做 CLR(Common Language Runtime)。CLR 就類似 Java 的 JVM，它隔絕了 MSIL 和硬體，能夠將 MSIL 轉成機器看的懂的原生碼，然後在此機器上執行。除此之外，CLR 亦提供執行中的程式一些管理機制，例如 GC(Garbage Collection 圾垃回收)、CTS(Common Type System)、例外處理(Exception Handling)、執行緒的支援(Thread)…等等，這些特性都是爲了提供 MSIL 一個穩定的執行環境，讓程式可以順利運行。

本篇文章版權爲倍力資訊股份有限公司所有，未經書面同意，嚴禁複製、轉載

目前爲止，我們可以把圖 1 再加以擴展來表示，表達.NET Framework 架構，各位可以參考圖 2 所示：

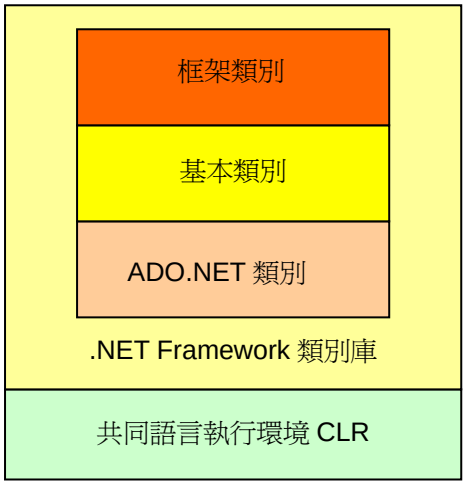

圖 2-2 .NET Framework 架構

Managed 和 Unmanaged Code

簡單而言，凡是在 CLR 所運作中的程式，都是 Managed Code，不是在 CLR 運作中的程式，就不是 Managed Code，也就是 Unmanaged Code。有些人以是否在.NET Framework 中運作來判定，可說是一樣的。例如 VB 的程式是 Unmanaged Code，VB.NET 是 Managed Code，C 是 Unmanaged Code，C#是 Managed code.(事實上，C#和 C 嚴格來說沒有關係，只是語言名稱類似；C#可說是微軟版的 Java 語言)。

假如是 Managed Code，就可以在.NET Framework 架構中，納入 CLR 的管理，包括前述的 GC(Garbage Collection 圾垃回收)、CTS(Common Type System)、例外處理(Exception Handling)、執行緒的支援(Thread)…等等。假如是 Unmanaged Code，就無法享受前述的管理機制了。

本篇文章版權爲倍力資訊股份有限公司所有，未經書面同意，嚴禁複製、轉載

CTS(Common Type System)

CTS，共通型態系統，主要是所有.NET 語言所共通使用的資料型態，這也是跨語言的重要關鍵。不管是哪一種語言，原本資料型態的定義可能不盡相同，但是經過符合 CLS 的編譯器編譯後，所產生的結果，包括 MSIL 和 Metadata(詳見後述)，都要符合 CTS 的型態，這樣程式的互通性就達到了。例如，C#的整數型態爲 int，Visual Basic 的整數資料型態爲 integer，將來只要經過符合 CLS 的編譯器編譯後，其整數資料型態將會是 int32，因爲 int32 是 CTS 標準的資料型態。當大家最終有相同的型態，就可促進跨語言的機制。

JIT Compiler、Assembly 和 Metadata

在介紹 JIT Compiler 之前，我們必須先要了解整個.NET 程式從編譯到執行的步驟。如前所述，程式碼首先要經過符合 CLS 規範的編譯器編譯程式，其結果會產生 IL 程式碼，而這些程式碼將會被包含在 Assembly 檔案內。Assembly 檔案除了 IL 程式外，還包括了 Metadata 資料，Metadata 資料是用來定義此 Assembly 執行時所需要的環境或是資源，以及 Assembly 本身的一些資訊，所以 Assembly 具備可再使用、可版本修訂和自我描述的功能，不需要如 COM 元件在作業系統註冊，這些都是 Metadata 的功能和特性。

Assembly=IL+metadata

Assembly 的檔案型態，可以是 exe 或是 DLL 檔案，但這些都是無法執行的，它需要第二次的編譯，將裡面的 IL 程式碼(別忘了，IL 是一種中介檔案，無法執行的，是所有.NET 程式語言經過 CLS 編譯器編譯的結果)，同時參考 Metadata 資料，轉換成機器可以執行的原生碼(Native code)，這樣才可以執行，而後者就是 JIT Compiler 的動作。

JIT Compiler 執行上的時機，是在需要執行 Assembly 時候才會有動作，它會將 Assembly 載入，然後進行編譯的動作，並不是老早就編譯好，所以顧名思義 JIT 的涵義就是 Just In Time，需要時才動作。JIT Compiler 包含在 CLR 內，提供.NET 程式能夠執行最主要的功能。

本篇文章版權爲倍力資訊股份有限公司所有，未經書面同意，嚴禁複製、轉載

我們可以持續擴展圖 3 的功能，將整個.NET 架構比較完整的勾勒出來。

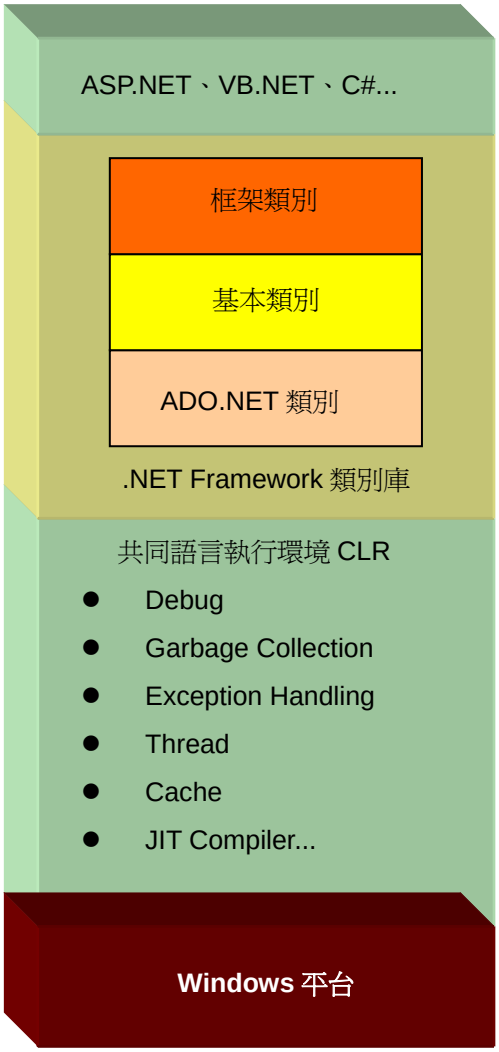

圖 3 更完整的.NET Framework 架構

前述都是.NET Framework 整體概念最重要的部份，對 PowerBuilder 開發人員而言，初步的概念都已包含，在開發上也幾乎能夠應付，所以讀者若對於.NET Framework 的不了解，因而擔心 PowerBuilder 在.NET 上的開發，這就多慮了。當然，不可諱言，假如希望要能更加對於 PB 和.NET 的緊密結合，就有賴於讀者對.NET 的深入了解，對於 C#語言、JavaScript 語言的了解、IIS Web Server 的管理特性，你能了解的愈多，對於開發過程上將會更有助益。 ###完

本篇文章版權爲倍力資訊股份有限公司所有，未經書面同意，嚴禁複製、轉載