PowerBuilder開發系列（1）

善用PowerBuilder開發 .NET程式

文／圖　黃智祥

本文將以一個不同的角度看待PowerBuilder for .NET，讓PowerBuilder開發者可以用在適當的地方應用，發揮最大效益。

自從PowerBuilder上市以來，演進的過程已經超過10年，能歷久而不衰，絕對有它獨特的地方，然而外在環境的變化，尤其是IT技術上的重大改變，似乎不少懷疑加諸在它的身上。其實身為PowerBuilder的推廣者，發現主要是因為不少使用者，並沒有深入了解其核心價值，進而產生不少誤解。因此，在開始討論技術方法之前，筆者將先帶領開發人員從另一個角度看待PowerBuilder，進而才能為企業產生效益。

IT 技術的分水嶺

從PowerBuilder 11以來支援 .NET技術，可說是PowerBuilder有史以來最重要的改變。筆者之所以強調這是一個非常重要的改變，原因是這也意味著將來PowerBuilder的發展方向，將強化 .NET的支援，絕對比想像的還多。

PowerBuilder 11支援 .NET技術包含：

- .NET WebForm：將寫好的PowerBuilder程式，轉換成Web Application。
- .NET WinForm：將寫好的PowerBuilder程式，轉換成以.NET為運作底層的Window Application。
- .NET Web Services：將撰寫PB NVO物件，部署成在.NET環境運作的Web Services。當然，也支援呼叫Web Services。
- .NET Assembly：將撰寫PB NVO物件，部署成.NET Assembly，供他人使用。當然，也支援匯入他人開發好的Assembly，呼叫使用。

上述這4個技術，全部歸納來說，它是將所有PowerScript的指令，全部轉換成C#的程式碼，然後將這些C#程式碼，再包裝成 .NET Assembly的形態來運作。

講到這裡，各位應該會想到，既然都是以C#為主，自然應該可以跟其他以C#為主要構成的程式，達到互通的效果，進而擴展原先PowerBuilder的領域，或是再次利用原本已經運作良好的C#元件，也就是程式碼的重複使用（reuse）。答案是正確的，這個功能，也就稱之為程式碼的互通性（interoperability）。

PowerBuilder要做到這樣的功能，最主要就是使用條件式編譯器「Conditional

compilation」的機制。條件式編譯器會根據所謂預定符號（Predefined symbols）的值，來決定你所界定範圍內的程式碼區塊，要使用哪一種編譯器來解讀程式。

預定符號的值將由你指定，例如預定符號的值若是「PBNATIVE」，後續的程式將以PowerScript的編譯器來編譯；假如值是「PBDOTNET」，則將以C#的編譯器來編譯。最重要的，就是位於C#區塊的程式碼，可以和PowerScript區塊的程式碼，其變數的值是可以互通的，所以就做到所謂互通的效果了。

程式1

```
variable  lv_var
#if defined PBDOTNET then
  /*action1*/
#end if
/*action2*/
```

讀者可以參考程式1，「#If…#end if」條件只有在 .NET類型的應用程式才會被編譯，其變數「lv_var」可以在action1和action2內都可以直接使用。

講到這裡，或許有讀者會問，對於.NET的世界，主要的功能、元件，PowerBuilder是否都可以拿來使用？沒錯，這個機制大幅增加PowerBuilder的擴展性，把 .NET環境的資源拿來運用。在PowerBuilder 11之後，幾乎可以稱之為PowerBuilder .NET，它是一個非常重大的改版，一個PowerBuilder歷年來版本中最大的分水嶺。

角色定位

既然PowerBuilder有這樣的改變，我們該如何發揮它的效益？之前的版本，除了Client/Server外，PowerBuilder主要針對Java的支援，這個版本仍然延續這樣的支援，但是重點放在元件的製作上，而不在前端的網頁。

▶ PowerBuilder Application Server Plug-in

我們可以從PowerBuilder 11.5來看，它把JSP Target這個功能移除掉，並且在產品的包裝上，新增加PowerBuilder Application Server Plug-in，就可看出一些彌端。PowerBuilder支援多層次架構程式的技術，前端可以使用PowerBuilder或是JSP，至於中間層的元件，則是使用PowerBuilder NVO（Nonvisual User Object）物件來開發，可以稱之為PB元件，對應到Java的領域就是EJB元件。

PB元件最後將被部署到中間層Application Server上，以往PB元件只能部署到Sybase的AP Server，也就是EAServer（Enterprise Application Server），對於其他產牌的AP Server則無法支援。PowerBuilder Application Server Plug-in這個產品，就可以讓你把PB元件部屬到其他廠牌的AP Server上，例如IBM WebSphere或是Oracle Weblogic，其最主要的目的就是在不同的領域中可以沿用PowerBuilder的技術。

▶ PocketBuilder

PowerBuilder 11.5還包含另一個新產品，就是PocketBuilder，它可以讓你開發Mobile Application，例如手機、PDA等行動裝置的應用程式。可在行動裝置上執行，並且離線處理資料，這些資料將暫時存放在這些裝置中，且由一個小型資料庫管理，就是SQL Anywhere。這些暫存的資料，稍後則會與企業內部的資料庫系統做同步。

雖然上述的2項產品實際上都已上市許久，但是這次都納入PowerBuilder 11.5，最重要是免費，不用再額外購買。如今，PowerBuilder的角色已漸漸成為一個綜合性的開發工具，也就是你只要一種語言：

PowerScript，使用同樣觀念，就可以開發數種不同的應用程式。相較於Java，更為單純，語法更為簡單，並且只要學習一次，就可以開發不少類型的應用程式，達到學習以及技術上的重覆使用性。

使用者需求為依歸

增加不少功能的PowerBuilder，要能帶給企業最大的效益，跟專案的需求有莫大的關係，例如End User操作的需求、對畫面的期待、企業文化、應用程式的特性等都可以算是。

舉個例來說明，由於PowerBuilder主要運作的操作介面是以Window Form的樣式為主，即便是產生網頁的Web Form應用程式，其特性也跟Window Form很類似。假如專案的需求是要能夠做到與HTML特性非常相關的網頁操作特性，那PowerBuilder可能無法滿足；但是假如專案經理依據專案需求的重要性，這個特性可以改成PowerBuilder既有風格的作法，並且說服專案的stateholder（利害關係人），那PowerBuilder就可以發揮非常大的效益，因為開發人員將不用花費昂貴的教育訓練成本，去學習HTML等相關技術，並且能重複使用既有PowerBuilder開發畫面的技術，即可產生網頁，這樣就可輕鬆達到開發上低成本和快速開發的目標。

另外，對於Web和Window應用程式的差異是必須要有所認知的。對於企業內部的資訊系統，往往有使用者會拿以往PowerBuilder既有Client/Server Window Form功能，來要求PowerBuilder .NET Web Form，假如是這樣的話，那絕對不是一件好事。很多功能在不同的技術上，原本就很難完全相同，硬要如此，則專案失敗的機率很高。

假如需求如此，最好回到原點思考更換系統的目的。其實對於End User，只會要求操作的便利性，以及資料處理的速度（他們可是急著下班阿），是不是Web他們一點都不介意。當然，使用者的要求也許和公司政策互有牴觸，這個時候，專案經理的重要性就會出來了。一個專案因為參與的人數眾多，每個人的角色和觀看事情的角度不同，所以有不同的專案需求，專案經理的重要性就是要協調這些需求，達到每個人都認可的程度。基本上，事先告知絕對比事後拒絕好辦事，尤其每個使用者通常會以菩薩心腸的態度，願意跟你討論他接受的替代方案，這樣對於專案的進行方便些，也不會花費心思對於PowerBuilder轉.NET WebForm，絞盡腦汁要完成跟Window Form一樣的功能。

善用特性，發揮效益

善用PowerBuilder的語言特性，儘量在它的特性下完成專案所需的功能，才能發揮最大的效益。讀者應該要有所認知，雖然PowerBuilder可以做到許多功能，但也並不是所有專案都適合，所以事前的評估很重要，評估後跟stateholder協調也很重要。

一般而言，internet應用程式也許畫面比較花俏，才能吸引使用者注意。但是對於intranet的應用程式，效能和便利性往往大於畫面的美觀。既然如此，intranet的應用程式一定要是Web化的網頁程式嗎？也就是每個程式都要轉成Web嗎？網頁的應用程式通常執行效能和操作的便利在intranet內運作，往往比不上傳統Client/Server 4GL的應用程式，所以這是一個直得思考的問題。是不是落入為了Web化而Web？

假如不是為了Web，而是前端程式的管理在Client/Server下過於複雜，所以不得已選擇Web。若遇到這樣的情況，筆者非常推薦讀者使用PowerBuilder .NET Window Form的方式來建構系統。因為對於支援度而言，PowerBuilder .NET Window Form和既有Native PowerBuilder支援度最高，轉換效果幾乎不變，而且支援Smart Client，可以幫你控制前端版本的問題，大幅降低前端管理的煩瑣，就這一點是和Web應用程式是相同的。最重要的，是因為它是.NET的應用程式，所以比起Native PowerBuilder，有更大的彈性可以擴增應用程式的功能，長遠來看，會是較適當的選擇。

至於Smart Client的架構可參考圖1，開發人員將最新的程式公佈在Server端，前端End User可以下載執行，並且定期檢查Server是否有新版，而提醒End User是否下載新版執行。

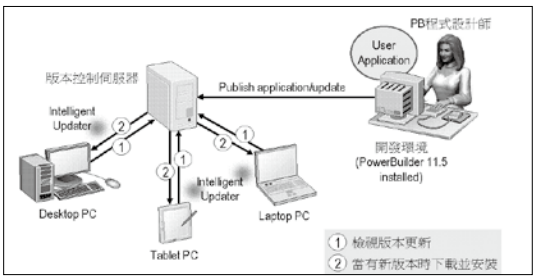

圖1　Smart Client運作架構。

學習之路

自PowerBuilder 11以來，很明確的往 .NET的方向前進，這對PowerBuilder的開發人員，確實造成不小的影響。也許有人質疑，既然如此，就直接選擇 .NET的技術就可以了。關於這個問題，筆者認為，如何才能發揮「效益」，這才是真正要考慮的重點。你必須發揮PowerBuilder語言的特色，對於你的專案，才會有對大的效益。別忘了，PowerBuilder還有一個重要的特色，那就是DataWindow，也就是DataWindow Anywhere。筆者了解 .NET的技術，以及一位資深的Java高手曾經告訴筆者，目前還沒有其他技術可以做到像DataWindow這樣特性的物件。

那麼，PowerBuilder開發人員應該要建立什麼樣的 .NET技術呢？其實不用擔心，儘管PowerBuilder可以支援 .NET，但是 .NET對PowerBuilder而言，並不是先決條件，而是一種加分的效果。技術層面的知識知道的愈多，對於專案將更有助益。基本上只要先了解 .NET的架構和觀念即可，下述主題就已足夠：

- 何謂.NET Framework。
- 何謂.NET Framework SDK。
- Assembly在.NET的角色。
- 初級IIS的特性和管理。

較深入的，就是學習C#語言，這是達到互通性最重要的部份。

小結

只要善用產品的特性與優點，發揮在對的地方，對個人或企業就能帶來更多的好處和競爭力。

責任編輯／洪羿漣

作者介紹

黃智祥

目前服務於倍力資訊加值服務部經理，負責PowerBuilder系列產品講師。從早期的PowerBuilder 4.0開發人員，到現在負責PowerBuilder的技術推廣。曾在TPUG等多項活動中發表專題演說，撰寫過PowerBuilder 8.0分散式進階應用一書。專長為程式語言、物件導向技術，擁有SCJP、SCWCD以及IBM Rational等認證。