PowerBuilder開發系列（2）

PB .NET Web Forms快速建構

文／圖　黃智祥

本篇將從 .NET Web Forms開始，它主要使用ASP.NET 2.0的技術，將PowerBuilder程式編譯成以ASP.NET和Assembly為主體的Web Application，最終在.NET Common Language Runtime（CLR）環境下執行。步驟並不複雜，讀者看完此篇，應可快速完成一個網站的建置。

上期筆者提到從PowerBuilder 11開始，加入了.NET的支援，主要技術包括產生.NET Web Forms、.NET Windows Forms、.NET Assembly以及.NET Web Service這四種類型的程式。接下來筆者將針對這4個主題做探討，希望能協助讀者有更深入的了解和體會。

這裡將以PowerBuilder 11.5的版本為例，要產生.NET Web Forms應用程式，環境上需要準備較多的服務和元件，各位可以參考下面各點，同時亦建議按此順序安裝：

- IE Server 5.0、6.0或7.0：作為產出ASP.NET Web Application運作的Web Server。
- .NET 2.0、3.0或3.5：提供IIS端執行環境和前端PB部署程式所需。
- .NET 2.0 SDK（3.0或3.5）：主要提供前端PB開發所需。
- AJAX延伸套件：IIS端、PB開發和部署所需，用來增進運作效能。
- GhostScript元件：用來產生PDF報表，主要在IIS端需要安裝，PB端安裝則非必要條件。所有版本皆可，目前最新為8.64。

前述所謂開發、部署和執行，是不同的環境，讀者應按照情況，分別裝在不同的機器上。一般而言，開發和部署是在同一台機器，IIS Server運作執行則是在另一台機器。假如只是測試，可以全部安裝在同一台，這樣就不用傷腦筋了。

按照前述的步驟安裝完後，即可開始安裝PowerBuilder 11.5，筆者建議你在安裝過程中，也一併安裝SQL Anywhere 11，它提供許多範例程式執行所需要的資料庫。最後，你所需的就是準備一個PowerBuilder Client/Server應用程式，準備部署成Web Application。

我們以PowerBuilder 11.5所附的範例程

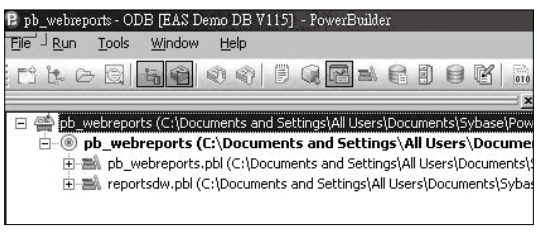

圖1　開啟「pb_webreporst」。

式：pb_webreports target來測試，讀者只要從開始-->程式集-->Sybase-->PowerBuilder 11.5-->Code Samples，然後選擇「Web Reports」即可開啟，如圖1。

使用.NET Web Forms Application Project物件來部署Web Application

要將PowerBuilder Client/Server的程式部署到Web上，就是使用.NET Web Forms Application Project物件來部署。我們可以利用新增一個target，透過Wizard過程，填入相關的資料，即可新增此project。

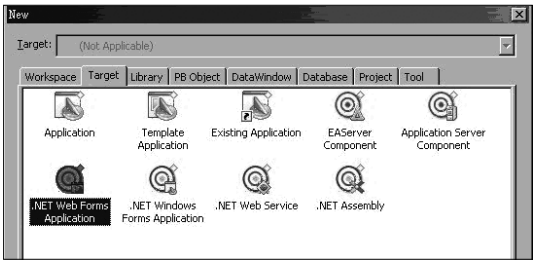

圖2　新增.NET Web Forms Application target。

對於以往PowerBuilder程式要compiler成執行檔，都是利用project物件，而事實上，PowerBuilder具有不同類型的物件，供你將程式部署成不同類型。前述PowerBuilder對.NET支援的主要4個特性，也是利用不同的project來幫助完成部署，讀者可以利用「New」視窗，切換到「Project」，即可看到不同的project物件種類。

接下來我們開始新增一個.NET Web Forms Application target，表示這是一個Web應用程式，參考圖2。之後的步驟，讀者可以參考表1說明其設定的意義。

前述步驟完成後，將會在Workspace看到新的pb_webreports_webform target，從名稱可以看出這是一個Web程式。接下來即可開始部署，請開啟此target下的project物件：p_webreports_webform，開啟後的畫面將如圖3，點選上方Toolbar上的Deploy Project按鈕，部署動作就開始了。

表1　.NET Web Forms Application Wizard屬性設定說明

視窗	設定
Create the application	選擇「Use the library list and application object from an existing target」，表示此Web application的程式來源。
Choose a Target	選擇要參考的target。本文選擇pb_webreports。
Specify Target File	指名target的檔案名稱。接受預設值：p_pb_webreports_webform。
Specify Project Information	用來指名此.NET Web Forms Application物件的名稱，以及位在哪個PBL程式庫檔案內。選擇pb_webreports.pbl。
Specify settings for .NET Web Forms Application	用來設定此Web Application的名稱，將決定你的網址URL。接受預設值：pb_webreports。
Specify Resource Files/ Directories	將此Web Application會用到的任何資源納入進來，例如圖檔、INI檔、dll檔案等。要注意的是雖然可以指名PBR檔案，但是它只是將PBR內所列出的檔案幫你加入進來，PowerBuiloder .NET Web Forms本身並不接受PBR的設定，PBR檔案只有PB Client/Server的程式才認得。我們選擇加入一個目錄：C:\Documents and Settings\All Users\Documents\Sybase\PowerBuilder 11.5\Code Examples\Web Reports\res，此目錄包含此程式會用到的所有圖檔。
Specify W32 Dynamic Library Files	加入程式所呼叫的外部DLL檔案。由於此範例沒有使用，所以可直接跳至下一步。
Specify JavaScript Files	PowerBuilder .NET Web Forms Application可以接受使用JavaScripts，假如有使用的話，必須在這裡加入進來。這個範例沒有使用，所以可直接跳至下一步。
Specify Deployment Options	選擇部署的時候，是產生安裝程式.msi檔案，還是直接部署到IIS Web Server。選擇.msi檔案通常表示PB無法直接部署到IIS，必須將此安裝檔拿到IIS所在機器上安裝。假如目前登入作業系統的使用者可以連線到IIS，且有寫入的權限，則可以選擇直接安裝。這部分請讀者自行設定。

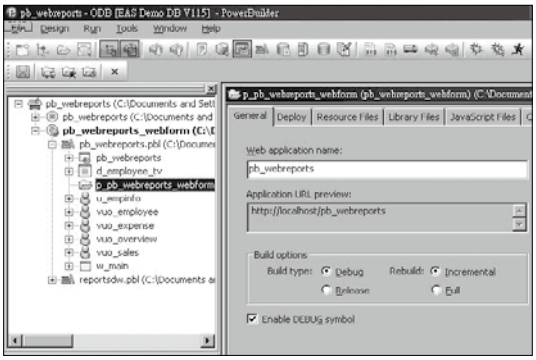

圖3　部署Web Application。

需注意，假如是Vista的作業系統，必須以系統管理員身分啟動PowerBuilder，然後才可進行部署的動作，否則這個步驟將會有錯誤。要以系統管理員身分執行，只要在啟動PowerBuilder 11.5的圖示上按滑鼠右鍵，選擇「以系統管理員身分執行」即可。

我們解析一下部署後在Output視窗所出現的訊息，這裡加註一些說明，讓讀者知道它的動作：

```
---------- Deploy: Deploy of p_pb_webreports_webform （下午 11:27:11）
Checking IIS server... //檢查IIS是否運作，權限是否可以寫入
Generating .NET assembly file... //將PowerScript轉換成Assembly檔案
Generating PBD files... //將包含datawindow的pbl產生對應的pbd檔案
Copying to IIS server... //將產生的所有檔案複製到IIS上，並在IIS建置一個Web Application
Deploy succeeded.
---------- Finished Deploy of p_pb_webreports_webform （下午 11:28:11）
```

部署完成後要能夠正常執行，還需再留意一個地方，也就是這些Assembly和PBD檔案能夠執行是不夠的，仍需要參考PB所提供在.NET運作的系統程式庫，也就是這些檔案的執行環境，這有點類似以往所謂的PowerBuilder Virtual Machine。

欲安裝上述的執行環境，可以利用PowerBuilder所提供的：Runtime Packager，將執行環境所需要的檔案，包裝成一個安裝程式，以便於安裝。只要從開始-->程式集-->Sybase-->PowerBuilder 11.5，選擇「PowerBuilder Runtime Packager」即可。

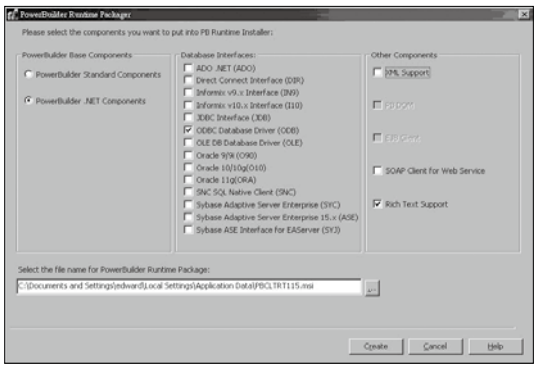

圖4　製作執行環境的安裝程式。

由於此範例使用SQL Anywhere 11的資料庫，使用ODBC連線，同時又是部署成.NET的應用程式，所以讀者可參考圖4設定，製作一個PBCLTRT115.msi的安裝程式，之後，再將這個安裝程式複製到IIS上安裝即可。

接下來必須啟動此範例所使用的資料庫，我們可以藉由PowerBuilder DB Profile功能，在ODBC的分類下，啟動EAS Demo DB V115即可。讀者可以將原本PowerBuilder Client/Server版本程式和部署後的.NET Web Forms Application版本做比較，彼此相似度非常高，參考圖5 Web版本。

圖5　pb_webreports網站。

截至目前為止，這些步驟並不複雜，但對於Web上的應用程式，有不少需要注意的地

方。所以接下來，筆者將探討幾個需要且常用的重要功能，這些功能在轉移到Web上行為已經改變，開發人員必須知道這些改變，甚至需要修改程式，以符合Web化所需。

報表應用環境

關於Web應用程式，有關報表的解決方案，大都是產生PDF檔案，然後在Browser上呈現。很大的原因是因為HTML所呈現的畫面會受許多方面的影響而有誤差，為了避免誤差，所以採用PDF檔案，同時PDF也是Internet上文件輸出的標準格式，在相容性和支援度上也是最高的。

PowerBuilder .NET Web Forms的報表功能，也是藉由產生PDF檔案來完成，而你所需要做的就是安裝GhostScript軟體。事實上就產生PDF的過程中，PowerBuilder會先產生一份PostScript文件， PostScript是由Adobe和蘋果電腦公司所合力開發的描述性語言，裡面詳述產生檔案的格式，而我們安裝GhostScript軟體的目的，就是它會讀取PostScript文件，參考文件內容來產生PDF檔案。

需注意GhostScript要安裝在IIS Server端，而不是Client端，所以產生的PDF檔案也是在Server端，前端使用者只是下載該PDF檔案，然後在Browser內顯示，假設讀者要在PowerBuilder開發環境下也要能夠產生PDF檔案，則也要安裝GhostScript。另外，完成安裝後，還需要在作業系統的path環境變數，加入安裝目錄下的bin 子目錄，以便讀取相關功能。

安裝完GhostScript後，接下來我們需要一個PDF印表機，這個印表機會把PowerBuilder所交付列印的內容和工作，產生PostScript檔案，然後再交由GhostScript轉換成PDF檔，所有完整程序藉此告一段落。PowerBuilder已提供此印表機驅動程式，通常位於\Shared\PowerBuilder\drivers\目錄，檔案名稱為ADIST5.INF。在安裝印表機你所選擇的連接埠必須是「FILE:（列印至檔案）」，同時產生的印表機名稱必須為「Sybase DataWindow PS」，絕不要打錯字了，否則無法產生成功。

讓我們再次執行此Web程式，由於我們要產生報表，所以將畫面切換到Employee頁面，利用此頁面右上方的列印功能按鈕，藉此產生報表。列印按鈕按下去後，網頁的右上方將會出現一個icon，這是所謂的Print Manager，PowerBuilder藉由此Print Manager來管理我們所產生的報表，可參考圖6。

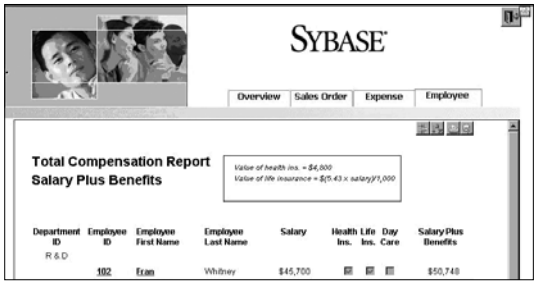

圖6　執行列印功能。

列印管理環境

Print Manager介面提供報表的管理功能，裡面顯示所有你列印出來的報表檔案，這些檔案都是存放在IIS Server端，並不是在前端使用者的電腦上。延續前面的動作，點選Print Manager後，將如圖7顯示目前所有的報表清單。

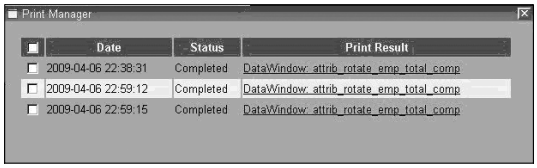

圖7　顯示目前所有的報表清單。

從圖7可以看出，具備簡易的管理功能，可以刪除報表、顯示產出時間，每個報表也

提供連結，以開啟該報表檔案。如圖8產出的PDF檔案，其內容和Browser所顯示的完全相同，唯一的差異是Browser所顯示的資料，已經提供分頁技術，避免在browser內顯示太多資料，降低網頁呈現的速度。但是PDF的報表檔案，則是顯示完整的資料內容。

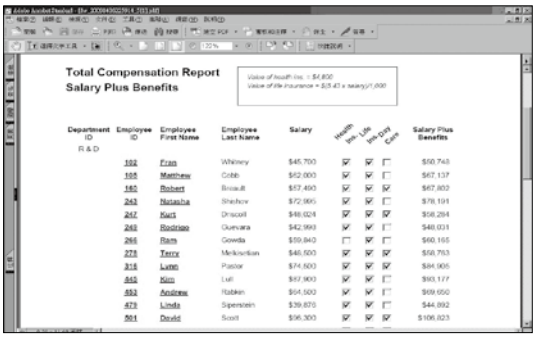

圖8　產出的PDF報表。

Print Manager所顯示的報表是以一個Session為範圍，也就是從你進入此網站，系統建立一個新的Session開始，所有在此Session期間所產出的最新報表，都會顯示在這裡。假如你關閉browser，Session將會消失，此時系統會自動將你的報表從Server端刪除，所以你不用擔心你所產出的報表會愈來愈多。另外，這些報表都是屬於你的檔案，你不會看到別人產出的報表檔案，你也不用擔心你的報表檔案被他人看出。

除了Print Manager，PowerBuilder還提供相關的管理介面，用來提供一些Web上常會用到的功能，同時也讓你不用再自行撰寫這些程式。這些包括：

- File Manager：用在Web程式中，需要下載或是上傳檔案的管理介面。例如，我們可以將DataWindow物件，利用呼叫SaveAs（）函數，將DataWindow儲存成各種不同類型的檔案，此時我們就會利用File Manager來指定儲存的目錄，這些目錄與檔案都是在IIS Server端，與Print Manager的機制非常類似。當然，我們也可利用File Manager來下載程式，對於常用的公用檔案或程式，若是Web應用程式所需，就可藉由File Manager來下載。
- Mail Manager：則是提供儲存寄發email的各種設定值，不同的設定可能包含使用不同的Mail Server來寄發，藉由這些設定值，不但撰寫程式將會更為輕鬆，同時更有彈性決定使用哪一種方式來寄發。
- Theme Manager：主要是更改程式的Theme主題，假如前端使用者是不同的作業系統，此時只要藉由Theme Manager，選擇適當的Theme主題，整個程式的風格就可以立刻轉變成符合該作業系統風格的Theme主題，而這部分也不用撰寫程式來做到，非常的方便。

IT 效能的增進

PowerBuilder支援AJAX的特性，可以讓網頁的呈現速度更加順暢，在使用上也可以避免整個網頁被重新讀取，只會讀取異動的資料範圍。至於這個範圍的大小，則是程式設計師需要注意的地方，以PowerBuilder對於AJAX更新的範圍，是以更新面板（UpdatePanels）為主，凡是更新面板內有所異動，AJAX只會重整更新面板內的資料和範圍，不會牽涉到所有的網頁。

更新面板主要由container control以及DataWindow control為構成，其中Container control可能是Windows物件、Tabs物件或是Visual UserObjects物件都算是。基於這些規則，程式設計師應該就資料異動的部分，分割成適當的更新面板，同時每個更新面板所包含的物件，可以儘量減少，這樣才可以減少更新範圍。例如針對一個Master/Detail DataWindow的設計，我

們可以使用Nested或是Composite DataWindow來做到，這樣是包含在一個更新面板內；或是我們分別將Master和Detail DataWindow，放置在各自的DataWindow control內，這樣則是兩個更新面板。就畫面異動的整體效能而言，後者將會比較好，因為每個更新樣板範圍最小，反應將會更為快速。

另一個增進效能的方式，就是針對所產生的Web Application，先做Compiler的動作，讓使用者直接執行機器碼，加速程式的執行。

由於原本PowerBuilder產生.NET Web Forms Application，程式碼為中介語言（IL；Intermedia Language），這要到使用者讀取的時候，才使用JIT（Just In Time）編譯器，將中介語言編譯成機器碼後執行，在時間上將會延誤許多。假如我們先做編譯，自然可省下這樣的時間，讓網頁直接執行，加速網頁的呈現。

先行Compiler的命令是使用.NET Framework所提供的編譯程式aspnet_compiler.exe，其語法讀者可參考微軟網站http://msdn.microsoft.com/zh-tw/library/ms229863（VS.80）.aspx。這裡節錄此程式的參數，請讀者參考：

```
aspnet_compiler  [-?]
 [-m metabasePath | -v virtualPath [-p physicalPath]]
 [[-u] [-f] [-d] [-fixednames] targetDir]
 [-c]
 [-errorstack]
 [-nologo]
 [-keyfile file | -keycontainer container ] [-aptca] [-delaysign]]
```

就預設而言，.NET Framework會安裝在[C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\]目錄，這也是aspnet_compiler.exe檔案所在的目錄。另外，PowerBuilder部署.NET Web Forms，在IIS上將會新增一個虛擬目錄，類型則為應用程式，名稱就是p_webreports_webform物件所定義的Web application name，以這個範例就叫做pb_webreports。此虛擬目錄就是我們所建立的Web Application，所有程式都建置在這裡。參考它所產生的實體目錄，將會在[C:\Inetpub\wwwroot\pb_webreports]目錄下。

知道的關連後，接下來我們就可以執行以下的程式，將此網站先行編譯，並且將編譯後的程式放在[C:\pb_webreports\ok]的目錄下：

```
C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\
aspnet_compiler.exe  -v  /pb_webreports  -p  C:\Inetpub\
wwwroot\pb_webreports  C:\pb_webreports\ok
```

「-v」的參數表示虛擬目錄，後面則是虛擬目錄名稱，也就是「/pb_webreports」；「-p」的參數則是程式碼所在目錄，也就是PowerBuilder編譯後，複製到IIS的目錄，這裡就是[C:\Inetpub\wwwroot\pb_webreports]。最後則是產出的目錄，我們將所有先編譯過的程式放置在[C:\pb_webreports\ok]的目錄下，但是要注意，執行此程式前，必須先把此目錄所有內容清空，否則無法在此產出編譯過的程式。

前述動作完成後，在[C:\pb_webreports\ok]目錄內就會有編譯過的程式碼，接下來你只要將此目錄下的所有檔案和子目錄，再複製貼回[C:\Inetpub\wwwroot\pb_webreports]即可。你可以再重新執行此程式，看看網頁的反應速度是否已經增加，就第一次執行的反應來看，反應時間將會縮短快50%。

總結

從本文的介紹可以發現，將PowerBuilder Client/Server程式，轉換成Web Application，步驟並沒有很繁雜，就程式撰寫的角度而言，並沒有對於.NET有所著墨，你還是可以使用既有PowerBuilder的語法和觀念來開發程式。

責任編輯／洪羿漣