

如何使用 PowerBuilder11 實作出最佳的.NET 專案

PowerBuilder 11 開發.NET 應用，使用的語言主要是 PowerScripts，關鍵在於透過編譯器的自動轉譯功能，產出標準.NET IL(Intermediate Language)的執行碼，運作於.NET Framework 環境上。這樣的開發機制，理論上雖說可以完全使用 PB 既有的開發經驗來開發新的.NET 應用，但實際上由於.NET 與 PowerScripts 這兩種語言間先天存在著一定的差異，例如在語法的結構上，PowerScripts 是比較鬆散、兼容的，相較之下，.NET 所使用的語言(如 C#)，則顯得比較嚴謹、沒有什麼彈性，這使得許多在 PowerBuilder 中能過通過編譯的語法，在.NET 編譯器上卻不一定能行的通，因此 PB 程式設計師在開發上需要注意一些細節，才能讓.NET 應用的開發流程更順暢，產出合乎期待的程式碼。為此，本期文章特地整理了 PB 開發.NET Target 應用須注意的事項，以便幫助程式設計師實作出最佳的.NET 應用。

兩個必須避免的函式寫法

- 避免使用 GoTo 語法

由於 PB 中沒有範圍(Scope)這樣的觀念，因此 GoTo 能夠在傳統的 PB 程式中使用，不過在.NET 中有範圍的觀念，雖然 GoTo 的語法能夠使用，但如果 GoTo 的 label 被標示在不同的範圍裡，那編譯時就會出錯，範例如下：

在這個範例中，PB 程式碼(左下)會被轉譯成.NET 程式碼(右下)，由於.NET 語法的特性 label 的部份會形成另一個新的範圍，造成編譯錯誤！

| 原 PB 程式碼 | 轉譯後的 C# 程式碼(會造成編譯錯誤！) |
|--|---|
| <pre> if b = 0 then label: ... else ... end if goto label </pre> | <pre> if (b == 0) { // opening a new scope label: ... } else { ... } goto label; </pre> |

- 避免以間接的方式執行一個物件的祖先事件：

例如有三個 Window，W_1、W_2、W_3，W_2 繼承自 W_1，W_3 又繼承自 W_2，在三個 Window 中都有一個 Clicked event，傳統 PB 中如果在 W_3 的 clicked event 以如左下方式呼叫執行 W_1 的 clicked event 是可行的，程式的意義是以間接的方式去呼叫執行祖先 W_1 的 clicked event，不過這樣的語法轉換成 C#的程式碼後，由於 C#並不支援如 PB 般的非直接呼叫

event 語法，這意味著在 C# 程式中邏輯會變成直接執行 W_3 自己的 clicked event(如右下)，雖不會編譯錯誤，卻會造成非預期的執行結果，解決的辦法是將祖先 event 中的程式碼移到 function 中，然後子孫再呼叫該祖先的 function。

| | |
|--------------|--------------------------------|
| 原 PB 程式碼 | 轉譯後的 C# 程式碼(不會造成譯錯誤，但會造成語意錯誤！) |
| W_1::clicked | base.clicked(); |

一些語法使用上需要注意的細節

- 1、不要在 Global function 中使用 this 這個關鍵字：
C# 的編譯器不支援這樣的語法使用方式。
- 2、不要改變 event 的簽名(signature)：
所謂 event 的簽名是指 event 的參數與回傳值，一旦祖先類別定義了該 event 的簽名，C# 語法是不允許在繼承的子物件中去更改該 event 簽名的，因此雖然 Powerscript 中沒有這樣的限制，但在開發 .NET 應用時則要避免這樣的程式設計方式。
- 3、不要改變一個繼承下來 function 的函式封裝範圍為 public：
一個祖先的 class 中，如果有某個 function 的函式封裝(也就是函式的存取權)被設定為 protected 或 private，繼承下來的 class 就不能再將該 function 的封裝設定修改成 public，C# 的編譯器不支援這樣的程式設計方式。
- 4、不要在 try-catch-finally-end-try 語法的 finally 語句中使用 return 敘述：
儘管 PB 允許在 finally 區塊中撰寫 return 敘述，但這樣的語法在 C# 中並不支援，會造成編譯錯誤。
- 5、不要在沒有直接繼承關係的 classes 間做 class type 轉型：
在 PB 中，在沒有直接繼承關係的 classes 間，是可以做轉型的，例如 nvo_2 和 nvo_3 都繼承自 nvo_1，在 PB 中 nvo_2 與 nvo_3 之間是可以轉型的，但這在 C# 中是不被允許的。

使用 External functions 所需注意的事項

- **PB 與 C# 在 external function 中傳遞 reference structure 參數用法上的差異**

宣告一個 external function 且傳入一個 reference 的 structure 變數，在 PB 中這個 structure 變數可以是繼承自 structure 標準類別的自訂的資料型態(例如 LogInfo 是繼承自 structure 的自訂類別)，這樣的語法是可行的，但轉成 C# 程式碼卻會出錯，程式範例如下：

```
Subroutine CopyMemory(ref structure s, int size) library "abc.dll"
LogInfo li // li is a descendant of structure
CopyMemory( ref li, 20) ←C#編譯器編譯時會產生 error，因為 C#編譯器認定 li 與原先
```

September 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

定義的 **structure** 類別是不同類別

由於 C# 本身語法上沒有這樣的支援，上述的程式必須改為如下語法才能通過 C# 編譯器的編譯，以 **reference** 方式傳入的 **structure** 類別參數，需改為往後具體要傳入的 **LogInfo** 類別，如此一來才能順利通過編譯：

```
Subroutine CopyMemory(ref LogInfo li, int size) library "abc.dll"  
LogInfo li           // li is a descendant of structure  
CopyMemory( ref li, 20) ←順利通過 C#編譯器
```

- 在 .NET 應用中以 **structure** 類別為參數的 **external function** 作法

延續上一個議題，在 .NET 應用中如果有 **External function** 以 **structure** 類別當作參數時，參數傳遞的方式必須用 **pass by reference** 而不是 **pass by value**，以 Powerscript 呼叫一個位於 **kernel32.dll** 名為 **SystemTimeToFileTime** 的 **External function** 為例，程式宣告方式如下(值得注意的是該 **external function** 需傳入兩個 **structure** 參數，第一個以 **pass by value** 方式傳遞，另一個以 **pass by reference** 傳遞)

```
Function boolean SystemTimeToFileTime(os_systemtime lpSystemTime, ref  
os_filedatetime lpFileTime) library "KERNEL32.DLL"
```

但在 .NET 的應用中，上面的 **external function** 傳入兩個參數都必須被宣告成 **pass by reference**，修改如下：

```
Function boolean SystemTimeToFileTime(ref os_systemtime lpSystemTime, ref  
os_filedatetime lpFileTime) library "KERNEL32.DLL"
```

之所以要特別提及 **SystemTimeToFileTime** 這個 **function** 是因為它被宣告為 **local external function** 並出現在 **pfncst_filesrvunicode**、**pfncst_filesrvwin32** 以及其他與作業系統相關的類別中，放置在如 **pfncst_filesrv.pbl** 的 **PFC** 類別庫底下，因此，如果在建構 .NET Window Form 或 .NET Web Form 時有使用到 **PFC** 類別庫，請注意要按上述方式做修改。

- C# 使用 **External function** 時，若須要以 **reference** 方式傳遞 **string** 參數所需注意的地方

在傳統 **PB** 中若以 **pass by reference** 的方式傳遞 **string** 參數給 **external function**，必須先呼叫 **Space()** 函式分配一塊記憶體給這個 **string** 參數，若是這個 **string** 參數因故變成了空字串(通常是因字串的長度超過所分配的記憶體空間所致)，在 **PB** 環境下通常還是可以正常運作，這是因為 **PBVM** 分配給這個 **string** 變數的記憶體空間並不會那麼快釋放掉。

但相同的情況在 .NET 應用就不一樣了，這樣的情況會造成 **external function** 存取該字串時發生 **exception**，唯一的解決之道就是盡量在傳遞 **reference string** 參數之前賦予足夠長度的記憶體空間。

就設計角度而言所需注意的事項

在這一個章節中，介紹的是一些編譯器不會視之為錯誤的程式設計方式，但這些設計方式

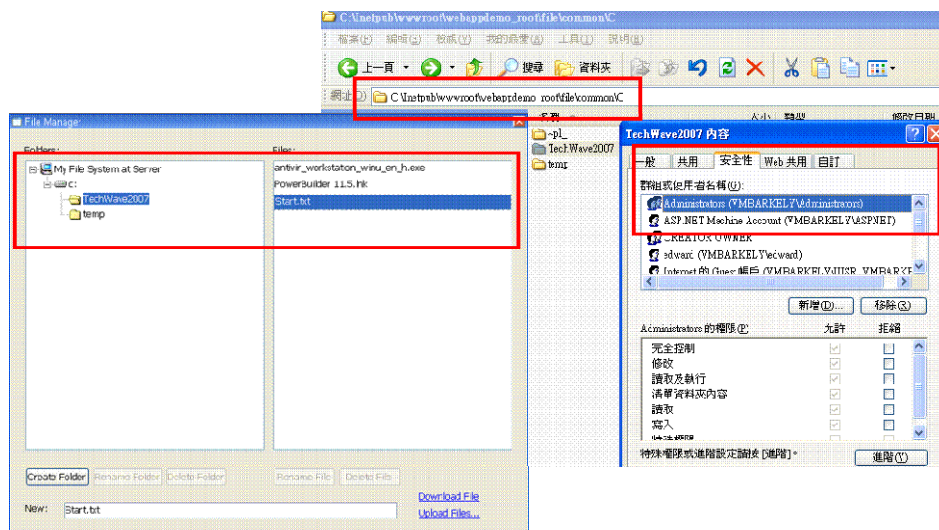
September 09 M-Power eNews

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

卻因為程式轉換為不同的應用架構(如轉成 Web 應用)，因應用架構特性的限制導致功能無法達成或錯誤，因此設計上建議遵守以下的規則，以便實作出良好的.NET 應用。

1、 盡量使用 PowerBuilder 所提供的 system function 來達成各種功能要求：

- 某些 PB 提供的 system 函式(如與檔案處理相關的函式)，在實作上因為已考慮到 Window Form 與 Web Form 架構特性不同而在程式部署上有著未言明的不同處理方式，因此 PB 程式設計師如果總是使用 PB 所提供的 system 函式，則便無須理會不同架構下所需的函式實作差異。
- 舉例來說，某些傳統的 PB 應用程式會呼叫外部的 DLL 檔來取得目前的資料夾目錄位址，但這在 PB .NET Web Form 下維持原作法會有麻煩，因為 PB .NET Web Form 應用使用的是虛擬檔案系統來模擬真實檔案目錄路徑，最重要的是這些目錄全部都要被賦予適當權限才能存取(如下圖一所示，使用者由瀏覽器上看到的 c:\TechWave2007 資料夾，實際上是已被賦予 ASP.NET Machine Account 權限的虛擬資料夾)，這時使用 GetCurrentDirectory 這樣的 PB system 函式會比呼叫外部 DLL 函式來的方便，因為資料夾目錄權限設定細節的部份 PB 會自動完成，程式設計師無須再操心 IIS 存取權限問題。



圖一、PB .NET Web Form 中使用者所看到的目錄結構實際上都是虛擬路徑，而該目錄需要被賦予適當的 IIS 帳戶存取權，使用者才能透過網頁去存取該目錄下的資料

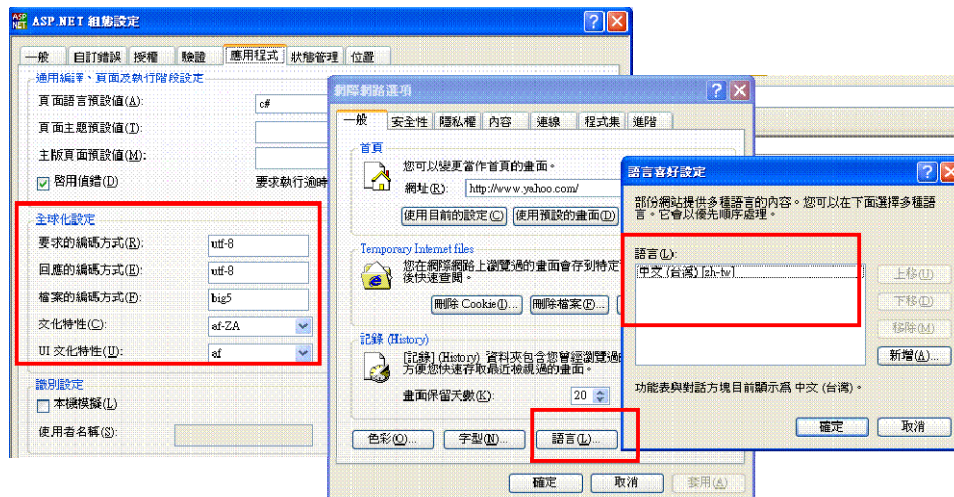
2、 使用 destroy 語法：

- .NET 的 garbage collection 服務無法觸發 PowerBuilder nonvisual object 中的 Destructor event，所以一旦有需要觸發該物件的 Destructor event，則必須明確的以 destroy 語法來進行。

3、 利用 PBCultureSource global 屬性來規範.NET Web Form 的區域性設定(regional)：

- IIS 每個網站的組態設定檔中都有一個名為全球化的屬性設定，這個屬性設定影響的是

使用者端瀏覽器上的語言喜好設定(如下圖二所示)，關於這個屬性，PB .NET Web Form Project 提供了一個相應的屬性來控制—“PBCultureSource”，這個屬性值有兩個選項：“Client”與“Server”，當該屬性被設定為 Client 時，網頁會依客戶端瀏覽器的語言喜好設定列表中第一順位的語言來作為區域性設定的參數，但萬一 Client 的語系偏好設定中沒有任何喜好的語言，則還是會以 Server 端所指定的全球化設定網頁編碼方式進行編碼。



圖二、IIS 中全球化設定與使用者端瀏覽器的語言喜好設定

- 事實上區域性屬性設定還會影響以下的項目，藉由這些項目，便可滿足世界上不同區域的格式設定需求：
 - i. numeric separators(數值的分隔符號選用：decimals / commas)
 - ii. number of digits per group to the left of a separator(在一個分隔符號左方的數字將以幾位數為一個群組，如：000,000.0)
 - iii. currency symbol location when a specific EditMask is not used(當欄位的編輯形式未指定為 EditMask 時，貨幣符號所顯示的預設位置)
 - iv. date and time values(日期及時間的格式)
- 此外，PB 中有以下列出的物件、控制項、函式會受到區域性設定的影響(包括某些特殊型態的 DataWindow Column、Controls 和 function)：
 - i. 當 DatePicker 控制項中使用到 DtfLongDate!、DtfShortDate!或 DtfTime! Format 時
 - ii. 當 EditMask 控制項包含有如[DATE]、[TIME]、[CURRENCY [{digit number}]] format 時
 - iii. 使用 MonthCalendar 控制項時
 - iv. 當函式 String (v) 其中的參數資料型態是 Date、Time 或 DateTime (而當

它們的 formats 設定分別為[SHORTDATE]、[TIME]或[SHORTDATE][TIME] 時)

- v. 當函式 String (v, f)當其中傳入參數的 format 是 [SHORTDATE]、[LONGDATE]、[DATE]、[TIME]或[DATETIME] 時

4、利用.NET 中所定義的中介字符(|)來測試一個字串是否滿足於一組以上的文數字組合樣式：

- 程式中如果要測試一個 string 字串中是否有某些特定的英數字組合樣式，在傳統 PB 程式中我們會使用 Match()函式來檢查，但該函式一次只能檢查一種文數字組合樣式。不過在.NET 相關的條件式編譯區塊(conditional compilation)下，同樣的函式卻可以用來檢查多種不同的文數字組合樣式，方法是使用.NET 承認的中介字符(|)將欲測式的英數字組合隔開即可，如下：

```
#IF Defined PBDOTNET Then
    Match(sle_ID.Text, "[^0-9]" | "[0-9][0-9][A-Za-z]$")
#End IF
```

上述的語法將測試 sle_ID control 中的字串是否帶有單一個 0-9 的數字或一組二位數數字後面緊接著一個大寫或小寫的英文字這樣的字組。

5、一些.NET 中不支援所以需要改變的程式寫作語法：

- 避免使用 Handle：某些應用會用 handle()函式取得某個 window 的控制權並將之交給某個 external function 繼續處理，但這在 PB .NET Web Form 中沒作用。
- 注意 unsupported events 的影響：雖然 unsupported events 在.NET 應用中不會被觸發，但如果這些 event 中存在某些重要的變數是正常 event 會用到的，就會影響程式的運作，需注意！
- 避免命名稱衝突(conflicts)：在傳統 PB 中物件或控制項的命名可以用相同的名稱，前提是只要類別不同就行了(如：structure 物件、static text control 和 nonvisual object 三個不同種類的物件都可被命名為 s_address，不會造成錯誤)，但在.NET 應用中是絕禁止兩個類別擁有相同名稱的，因此.NET 應用設計上必須避免類別命名相同的問題。
- 避免在繼承的物件中去使用 local structure 變數：在繼承下來的物件中使用 local structure 變數會造成.NET 應用的部署失敗，解決方案是將其中所有 local structure 變數改為 global structures 來讓繼承的物件使用。
- 在 PB .NET Web Form 中使用 AcceptText()函式是多餘的：在 PB .NET Web Form 中，所有 datawindow 在 loss focus 的情況下都會自動呼叫 AcceptText()進行資料的檢驗，因此自行另外撰寫程式呼叫 AcceptText()是沒有必要的多餘行為，但並不影響程式運作！

6、避免使用可能會妨礙程式效能的語法：

以下列出的物件或函式，雖然在.NET 應用上都支援，但卻有可能造成效能不彰的問題，須謹慎地考慮是否使用，或儘可能少使用。

- Response windows 和 message boxes: 在 PB .NET Web Form 中 Response windows 和 message boxes 都是非常消耗 Server 端資源的物件，此外，以 hind 的方式讓 Response windows 暫時隱藏可能會造成程式運作異常，因此如果使用 Response windows 的話，最好確保使用者會將之正常關閉。
- Yield : PB .NET Web Form 中 Yield()函式會消耗額外的 Server 端資源。
- Timers : 雖然 timer()函式在 PB .NET Web Form 中是支援的，但由於其週期性地驅動網頁的 postback 機制，這期間會影響某些資料的操作，因此如果不得不使用該函式時，需注意避免在 timer()函式使用期間進行資料的操作，並盡量以適當的 Client 端 Scripts(JavaScripts)來延遲網頁的 postback 動作。
- PFC : PFC 中有許多 datawindow event 處理的模組，其中某些 datawindow event 可能會隨著使用者點選滑鼠，每一次都去驅動網頁的 postback 動作，造成效能嚴重的下降，解決的辦法是將 datawindow object 的 paging method 屬性設定為 XMLClient!，該屬性設定能夠確保該 datawindow object 只有在執行 update()指令時才會驅動網頁的 postback 機制。

其他限制與注意事項：

除了上述的說明外，PB .NET Web Form 的 Project 中還有一些 global configuration 屬性以及在 datawindow control 中有十個 events 可編寫成在 Client 端執行的 JavaScripts，以上兩種作法都能夠對 PB .NET Web Form 的效能有所幫助，但我們這裡並不做詳述，讀者們可自行參照 Compiled HTML Help File 下的 Take advantage of global configuration properties 以及 Take advantage of global configuration properties 這兩個小節的敘述。

結論

本次 PB 主題文章介紹了一些 .NET 應用開發上需注意的設計規則與語法限制，掌握這些訊息可以幫助 PB 程式設計師在設計 .NET 應用時得到比較好的效能及避免掉一些不必要的小問題，希望能帶給大家一點幫助。