PowerBuilder開發系列（3）

PB .NET Windows Forms Application

文／黃智祥

繼PB（PowerBuilder）快速開發.NET Web Form後，接著將介紹.NET Windows Form的開發，而.NET Windows Form究竟是什麼應用，它又有何特別之處呢？本文將其配合Smart Client機制，使得程式設計師能夠省去傳統C/S程式手動部署的麻煩。另外，還包括該如何引用.NET Assembly物件擴充程式的功能，以及最後針對PowerBuilder 11.5進階的CAS機制（Code Access Security），應用於解決程式在客戶端的安全性問題。

基本上 .NET Windows Form是.NET Framework支援的眾多應用類型之一，它就是一個桌面應用程式，必須被部署到客戶端的主機上才能運作，當這個類型的應用中必須與遠端資料庫連線協調運作時則特別稱之為C/S（Client /Server）應用。

Smart Client有何用？

那Smart Client又是什麼呢？它是一個被用來解決桌面類型應用程式部署問題的技術，長久以來無論是桌面或C/S應用，最大的問題都是必須花費大量的時間與人力在客戶端程式的部署、維護上。

Smart Client透過內建於.NET 2.0中的「Intelligent Updater」技術，大大的降低客戶端程式安裝所需的步驟及技術門檻。該技術能夠協助客戶無障礙的安裝程式，其中最主要的關鍵就是Click Once技術，讓主程式安裝步驟能夠完全以精靈方式引導，全自動設定、安裝，理論上客戶端的使用者只要按下一個鍵，同意安裝的按鍵，程式本身就能完成自動部署且可立即使用。

但Smart Client並不僅僅只能應付第一次安裝，他同樣也支援客戶端應用程式的版本控管，當有新版本發佈，客戶端的Smart Client機制會自動地與伺服器連線確認，並在客戶端使用者授意的情況下自動進行新版程式下載、安裝的動作，由於整個過程彷彿有一個聰明的客戶端小幫手做完所有該做的事而得名。

不過也因為Click Once是.NET 2.0中所包含的其中一項技術，因此如果程式要支援Smart Client這樣的機制，就必須在.NET 2.0上運作，傳統的PB Win32程式並不支援。

更進一步說明Smart Client機制的細節，可以用圖1來做簡單的介紹，這個機制主要的精神在於程式設計師完成程式開發後，編譯後的程式碼將部署到所謂的「程式碼託管伺

服器」上，接下來，各客戶端電腦將主動與伺服器聯絡、下載，並自動完成安裝。無論是第一次還是往後的版本更新，基本上不脫這樣的運作模式。

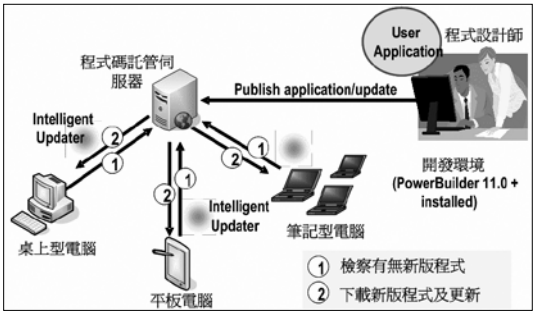

圖1　Smart Client運作機制示意圖。

Smart Client是包含於.NET Windows Form中的一個配套機制，因此想要運用Smart Client，那首先也必須擁有一個.NET Windows Form應用，不然就必須自行重頭開發。這一點對於熟悉PB C/S應用開發的程式設計師來說，一點都不困難，因為所有的步驟、流程包括使用的語法都是傳統的PB開發方式，新增.NET Windows Form Target便能向Smart Client應用跨出第一步（圖2）。

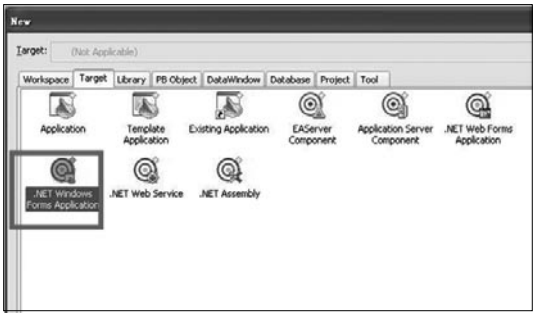

圖2　.NET Widows Forms Application。

在開發.NET Windows Form上，對於已經擁有PB應用的既有用戶來說有一個最大的優勢，那就是程式碼可直接轉換為.NET Windows Form應用。由於轉換的支援度非常高，需要改動的項目不多，如果讀者原先就有正在運作的PB C/S程式，那直接透過Project部署的功能，便能立即完成程式的轉換。

開發觀念與技巧

那麼開發Smart Client應用需要什麼樣的前置準備工作？程式設計師又要學習哪些語法與觀念？答案是完全不用，所有的工作都只要透過Project的設定就能完成了。Project是一個專門管理部署工作的物件，透過改變其中的屬性設定，程式設計師可以很快的變更部署的結果，.NET Windows Form下的Project物件如圖3所示，以滑鼠點選物件，則Project的編輯視窗就會在右方的編輯區展開來，如果程式設計師打算將應用部署成Smart Client，則只要完成Project中相關的屬性設定，就能讓Project自動完成Smart Client應用部署。

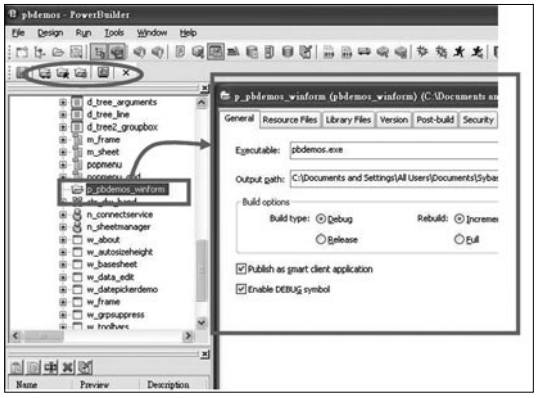

圖3　.NET Windows Forms Project。

Project的設計上，為了方便負責程式部署人員搜尋及設定，所有與部署相關的屬性會分門別類，歸到不同的頁籤中。其中與Smart Client機制相關的屬性頁有：Publish、Install/Update與Prerequisites。其中，Publish屬性頁（圖4）主要是設定程式碼託管伺服器的資訊，以及程式發佈到託管伺服器後是否要連帶自動產生「應用程式安裝導引頁」。如果同

意產生，這個引導頁面預設的名稱為publish.htm，它能夠導引客戶端使用者透過它所提供的超連結去下載、自動安裝主程式。

發佈程式到託管主機可分3種類型，程式設計師可依主程式檔案大小及環境架構需求來選擇最適合自身專案的類型進行主程式發佈：

1. Web site：將主程式發佈到IIS主機上，程式會被發佈到IIS伺服器的Inetpub\wwwroot\指定的目錄名稱下。
2. File path：程式會被發佈到domain中的共用磁碟區。
3. FTP site：程式被發佈到指定的FTP站台。

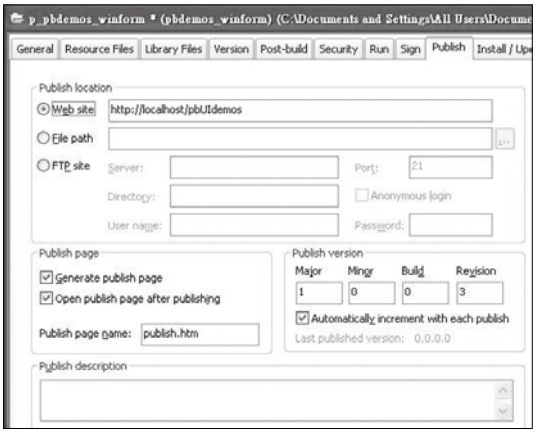

圖4　Publish屬性頁。

Install/Update屬性頁負責控制主程式在客戶端的行為（圖5），包括：

- 啟動模式（Running mode）。
- 程式版本託管規則（Mandatory update）。
- 程式版本更新模式（Update mode）。
- 主程式安裝及啟動路徑（How application will be installed or launched）。
- 程式版本更新位置（Update location）。

上述的啟動模式是這個屬性群組的關鍵，當啟動模式被設定為「Run from browser only」時，程式的表現將類似Web頁面的行為，啟動主程式的方式是依靠Url位址的導引，

圖5　Install/Update屬性頁。

而不是安裝在客戶端桌面的執行檔捷徑，在這個選項之下，主程式執行時僅暫時被下載到客戶端某個登錄帳戶的快取之中，當程式被關閉，這些檔案也就被移除，非常適合某些需要執行效能但又不能安裝在客戶端的應用。

如果啟動模式設定為「Run from browser or Start menu」，則程式會被永久安裝到客戶端電腦主機，且啟動主程式的捷徑會被設定在開始→程式集，換言之，使用者可以由客戶端主機啟動該程式，且主程式安裝完成後，也會被自動加入到Windows作業系統的「新增或移除程式」集合中。再者，當這個模式被設定後，接著上述屬性項目2~5也就必須連帶設定，包括設定程式安裝的位置、更新的規則、頻率、檢查程式版本時機以及下載更新程式的位置等。

最後一個我們要介紹的屬性頁籤為Prerequisites，這一頁主要的功能是負責管控客戶端環境建置，其實C/S程式要能正常運作，光只安裝有主程式是不夠的，主程式運作時周邊的應用軟體也需要一併建置才行，而這往往才是C/S架構真正的致命傷。

所幸這個問題在Smart Client中已有了解決的方式，透過將所需的軟體包裝成標準規格的MSI檔（Microsoft Windows Installer; 程式封裝

檔），如此一來當使用者在安裝主程式之前若被偵測到沒有適當的環境，便可下載這些檔案來安裝，我們可由圖6中看到，頁籤中已預設準備了兩個檔案，分別是.NET 2.0及Sybase PowerBuilder .NET Runtime，以及另一個筆者自行加入的SQL Anywhere 10客戶端軟體，選項若被勾選，當主程式被發佈，項目中的軟體也就會連帶被發佈到程式碼託管伺服器，讓使用者選擇安裝。

圖6　Prerequisites屬性頁。

著手部署

當所有選項都完成設定後，我們就可以開始部署並發佈Smart Client應用了，方式如圖7的圖說文字所述，至此，讀者們應該可以感受到整個過程並不困難。

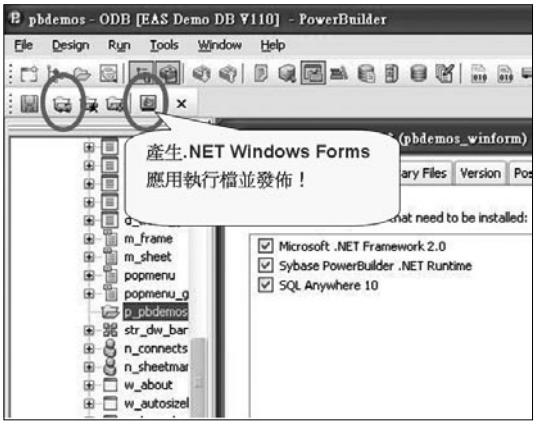

圖7　發佈程式為Smart Client應用。

接下來為了方便說明，我們將分為2個簡單的劇本來說明整個Smart Client機制運作的實用性。

▶劇本1：主程式暫時安裝至客戶端

這個方案適合替代部分運作負載較重的Web頁面，發佈完成後如果有設定自動產生「應用程式安裝導引頁」，PB就會自動為我們產生一個圖8的頁面，這個頁面主要負責說明應用程式的名稱、版本以及所需安裝環境等資訊。若使用者跳過環境安裝的選項，直接執行主程式，Smart Client中也會自動偵測客戶端環境是否完全，進而提示安裝，以上所述都是由頁面中所提供的超連結來導引。

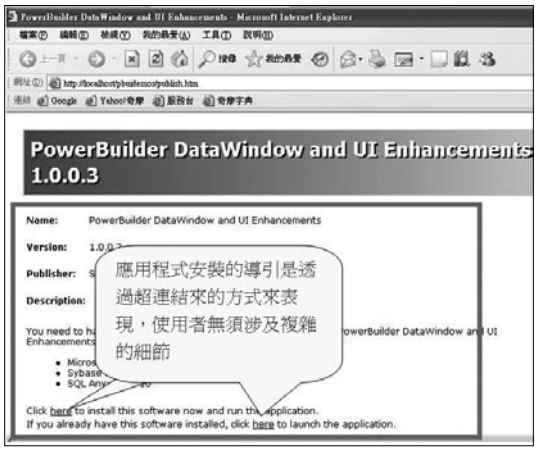

圖8　應用程式安裝導引頁。

點選超連結選項，結果我們可以看到如圖9的彈跳視窗，其中1.視窗為環境建置的確認對話框，2.視窗為主程式執行確認視窗。在這個劇本中，因主程式並非常駐在客戶端，雖然整個應用是桌面形式的應用，但其特性卻非

圖9　環境及主程式建置、啟動確認對話視窗。

常類似Web，每次執行都必須透過超連結來啟動，如本例中的啟動超連結為：http://localhost/pbuidemos/pbdemos.application，也因此沒有客戶端程式版本控管的問題，執行結果如圖10。

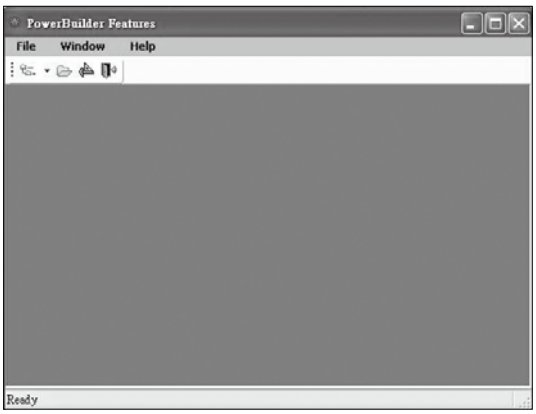

圖10　劇本一執行結果。

▶劇本2：主程式常駐於客戶端

這個方案會將主程式永久安裝在客戶端，部署後產生引導安裝的頁面與操作基本上與劇本一雷同，最大的差異是在於主程式必須被安裝到客戶端（如圖11所示），但安裝的過程是全自動進行。

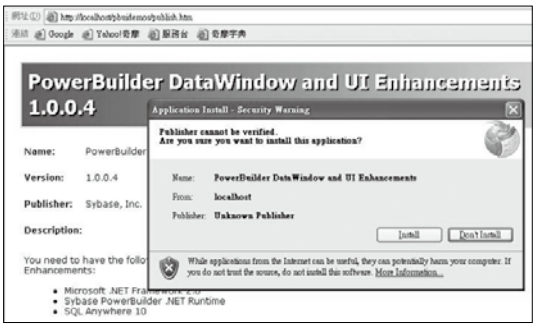

圖11　安裝程式在客戶端的確認視窗。

安裝完成後我們可以由Windows作業系統的開始/程式集去啟動該應用，而當應用程式執行時，位於螢幕右下角的常駐程式區也會有一個客戶端程式版本軟體管理告示器，能夠讓使用者主動或定期與伺服器連線，做程式版本更新的管控（如圖12）。

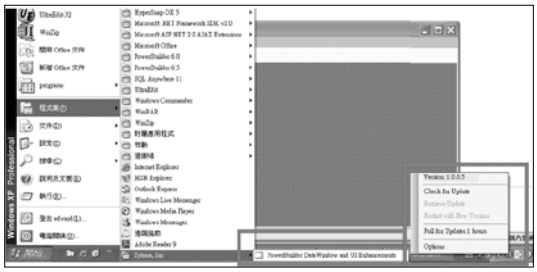

圖12　劇本2客戶端的程式啟動與版本控管機制。

在劇本2中，除了客戶端程式版本能及時更新外，最特別之處是當應用程式更新後若發現新版本有Bug，也能立刻做版本回復的動作。另一方式是由客戶端作業系統的「新增或移除程式」功能來完成版本回復動作（如圖13），這個方式能夠將程式回復到前一個版本。

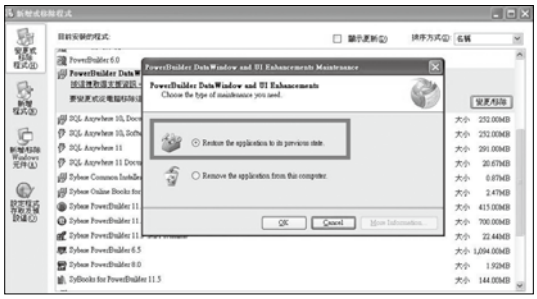

圖13　客戶端驅動的版本回復功能。

此外，還可由伺服器端的版本識別檔（manifest）來驅動，該檔案會紀錄每一次程式發佈所關連到的程式實體以及資訊。由圖14的紅框標示的檔案群中我們可以看到，PB每一次發佈程式都會連帶產生一個版本識別檔來記錄程式版本，管理者只要透過變更這些檔案，則所有客戶端就會在下一次與伺服器主機連線時回復到管理者指定的版本，值得注意的是這樣的機制並不限於向前回復一個版本，管理者可任意指定回復到先前的任一版本，而且這個機制會影響每一個客戶端，當程式出現重大更版問題時，便可以透過這個機制來全面修正，由此可知Smart Client機制非常完整、穩固。

圖14　位於伺服器端的版本識別檔。

在.NET Windows Form中引用Assembly

.NET Windows Form另一個有價值之處是PB程式設計師能夠在應用中去引用Assembly，最顯著的好處就是PB程式設技師能夠使用.NET 2.0底層中龐大的類別庫資源，或者使用由.NET廣大社群所開發出來的程式來擴充自身應用的觸角。

在過去PB要使用外部的程式方式無非是藉由OLE不然就是使用External Functions，但這些應用方式的文件很少公開流通，常迫使PB程式設計師必須憑經驗，用Try Error的方式盲目開發，過程中吃足了苦頭。

但如今在.NET應用中PB程式設機師可以用更簡單的方式來擴充應用本身，如圖15所示，只要是在PB11 .NET相關Target下，開啟屬性（Properties）選項都可以發現一個.NET Assemblies參考的設定視窗，引用過程非常簡單，只要在該視窗中去指定欲引用的Assembly檔案就算完成，且被引用的Assemblies會在PB的System tree中顯示其完整樹狀結構，包括屬性及方法。

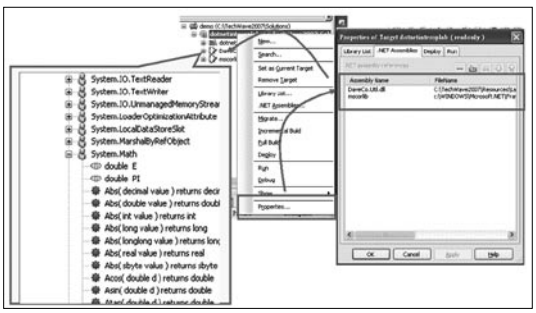

圖15　在.NET Windows Form中引用Assembly。

在這個例子中，筆者在程式中引用.NET 2.0底層的mscorlib.dll Assembly，並使用集合命名空間（System Namespace），在這個命名空間下有System.Collections.ArrayList類別及System.Collections.IEnumerator介面，能夠接收、暫存來自於資料庫的資料，最後再以.NET的Messagebox.Show（）方法將集合陣列中的資料項輸出，程式碼如圖16所示。

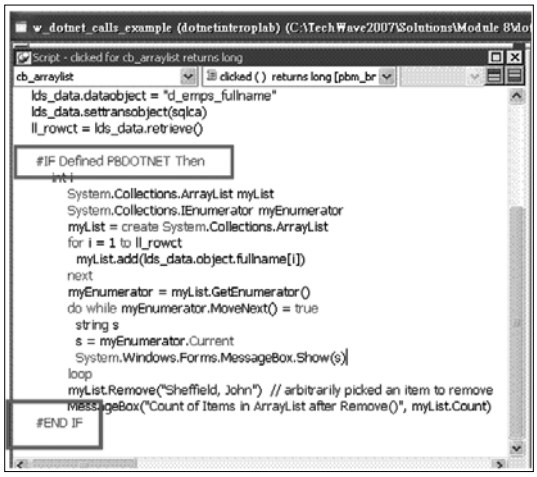

圖16　以.NET的方法將集合陣列中的資料項輸出。

值得注意的是，程式設計師必須把來自於Assembly的類別及方法寫在Conditional Compiler區域中，在PB11的各.NET Target中加入了以「#IF Defined PBDOTNET Then」開頭，並以「#END IF」結尾這樣的語法來容納非PB原生的程式碼，不過在這個區塊之外PB程式設計師還是可以寫傳統的PB Scripts，有趣的是無論是區域外部或內部所宣告的變數都

還是可以互相傳遞使用，這個技術有一個專有名詞稱之為「.NET程式碼的互通性（.NET Interoperability）」。

在上述的例子中，位於Conditional Compiler區塊中的語法非常類似於傳統的PB Scripts，只是宣告類別的方式必須以.NET Namespace的完整名稱來宣告之，對於PB的既有用戶或程式設計師來說，只要大致上了解.NET程式的基本概念就能很快上手，執行結果如圖17。

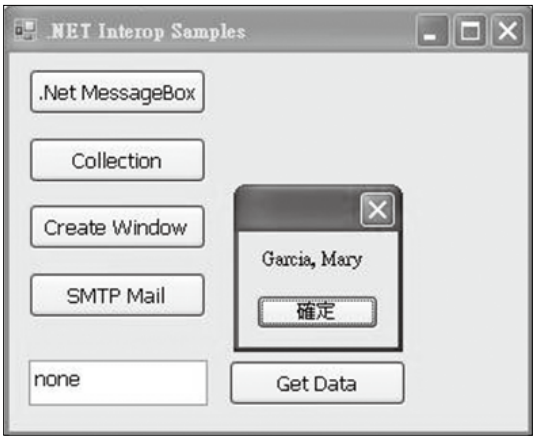

圖17　引用.NET Collection的執行結果。

Code Access Security機制

在本期報導的最後這一部份，將介紹PB11.5內含的一個新程式碼安全管控機制。過去，程式執行權的控制都是透過帳號、密碼模式來進行，只要使用者通過驗證程序就能使用應用程式全部的功能，但這樣的方式在要求日益嚴格的客戶端程式執行安全管制上已經顯得過於粗糙。其中一個問題是，如果程式設計師開發應用時引用了來自第三方所提供的程式，該如何保證這些程式不會威脅企業內部的安全呢？畢竟我們無法確認第三方的使用者不會寫一些後門程式或者暗地進行一些企業所不允許的動作，但最糟糕的是.NET應用很難不去引用外部的程式，因為那正是.NET開發的基本精神。

CAS（Code Access Security），正是用來解決這個問題的技術，這個安全設定的機制是架構在.NET Framework之上，這是一種向下細部到程式碼的安全管裡機制，透過限制某些程式碼的執行，便可確保該應用不會暗地進行超出程式設計師預期的動作。

雖然CAS是針對程式碼控制的技術，但PB程式設計師無需著墨於如何撰寫程式碼，在PB11.5中我們只要透過.NET Windows Form的Project找到安全性（Security）屬性頁籤（如圖18），透過設定便能達到目的。

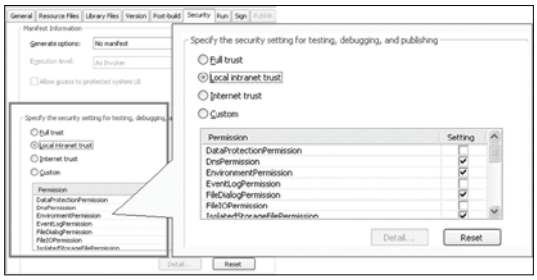

圖18　PB11.5 .NET Windows Form Security屬性設定頁。

這個安全性頁籤中的選項其實是對應到.NET Framework 組態工具 （Mscorcfg.msc）中的「執行階段安全性原則」，讀者可以在你可以在[%windir%\Microsoft.NET\Framework\v2.0.xxxx]或[\\Program Files\Microsoft.NET\SDK\v2.0\Bin]中找到 Mscorcfg.msc。

上述的這個工具是.NET Framework中提供管理人員設定安全性原則和使用遠端服務的應用程式的圖形化設定工具，同時它也支援管理和設定全域組件快取（GAC）中的組件。這個機制的存在，讓.NET CAS的控管並非一定要依靠寫程式達成，如圖19。

在PB環境中，程式設計師可對客戶端的程式執行做幾個層級限制：Full trust、Local

圖19　.NET Framework Mscorcfg.msc安全性原則設定工具。

intranet trust、Internet trust、Custom。

除了Full trust是完全不對程式做任何限制外，其他3個選項都能讓程式設計師自行選定程式中的哪些行為將被限制，細部微調的相關屬性如表1。

如：SQLClientPermission的選項如果沒有被勾選，當使用者執行應用中有關資料庫連線的程式碼時，不但連線動作被禁止，同時也會彈

圖20　當資料庫連線功能被限制時，客戶端執行連線動作將被禁止。

跳出錯誤的警告訊息告知使用者該應用不支援資料庫連線功能（如圖20）。

只要在程式部署前決定好安全性選項，則這些有關程式執行安全性控制的屬性就會被記錄到我們先前所述的伺服器端版本識別檔案中，連同主程式一起部署到客戶端對程式行為進行管控，且在PB11.5中除了.NET Windows Form Target外，.NET Web Form Application應用與.NET Web Service應用也都支援這樣的安全控制屬性。

表1 安全性原則微調

屬性名稱	安全性項目
DataProtectionPermission	能否存取已經加密的資料和記憶體。
DNSPermission	DNS使用權。
EnvironmentPermission	環境變數使用權限。
EventLogPermission	能否將log資訊輸出到PBTrace.log中。
FileDialogPermission	是否能夠呼叫作業系統的檔案對話方塊。
FileIOPermission	能否作檔案IO，包括：讀、寫、附加以及路徑描述等。
IsolatedStorageFilePermission	能否存取隔離儲存區檔案。
KeyContainerPermission	自訂使用者權限：System.Security.Permission.KeyContainerPermission。
OleDBPermission	自訂使用者權限：System.Data.OleDb.OleDbPermission。
PerformanceCounterPermission	是否有作業系統上的效能計數器存取權。
PrintingPermission	是否有印表機資源的存取權。
ReflectionPermission	使否有探索其他組件中未開放屬性的權利。
RegistryPermission	能否存取作業系統中註冊機碼中的變數。
SecurityPermission	是否有其他managed code的執行權。
SocketPermission	能否使用ConnectToServer函式與其他伺服器連線的權限。
SQLClientPermission	能否存取Microsoft SQL Server的權限。
StorePermission	自訂使用者權限：System.Security.Permissions.DataProtectionPermission。
UIPermission	是否能夠存取使用者介面如：視窗、剪貼簿的權限。
WebPermission	是否能夠存取Web網站的權限。

總結

本文介紹簡短的介紹了如何利用PB11開發Smart Client應用，相信讀者能夠快速認識PB11開發.NET Windows Form的好處，除了不必撰寫程式，透過簡單的屬性設定就能輕易達到版本控管的目的外，.NET Windows Form應用還能於Conditional Compilation區域中去引用.NET Assembly的類別及屬性，這使得.NET Windows Form應用在開發上更廣泛且兼具彈性，如果再配合客戶端的CAS程式碼安全性機制，就能讓C/S架構的程式適用於更廣泛的領域。

責任編輯／洪羿漣