PowerBuilder開發系列（4）

PB .NET Web Service實際運用

文／圖　黃智祥

本文除了簡短介紹Web Service技術外，將分3個段落說明PowerBuilder 11(以下簡稱PB11）在Web Service應用上的相關議題，包括開發部署Web Service、傳統PB程式如何呼叫外部的Web Service服務，以及PB11中DataWindow新增了Web Service資料來源，這是一個蠻有特色的新功能。

雖然PowerBuilder在第8版就將Web service的功能納入之中，但相較於PB11，正式將Web Service獨立為一個新的開發標的(Target），之前PowerBuilder有關Web Service開發功能，顯的複雜多了。在PB11，Web Service的開發已擁有簡易、快速、實用的特性，希望藉由本文，能讓讀者對PB11在Web Service的應用有所了解。

Web Service的問世，最主要的目的就是為分散式運算提供一個跨語言、跨平台的統一介面，在過去無論是企業內部或是合作夥伴間，儘管早已存在許多電腦應用系統，但實際上卻經常礙於各種不同的技術規範、不同語言、不同平台等等，使得這些資源無法被整合以及重複使用，造成許多困擾。

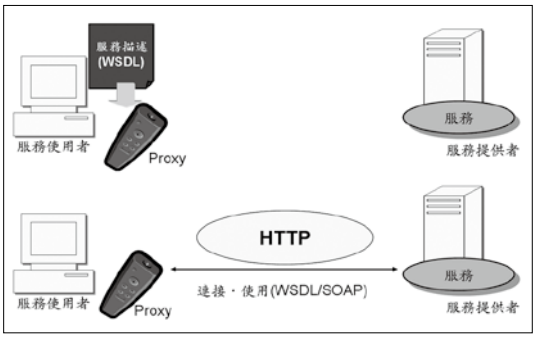

圖1　Web Service運作機制圖。

Web Service以文字資料傳輸為基礎，資料、訊息部分則以XML格式來描述，加上基於HTTP上的SOAP通訊協定，基於這些廣泛通用的元素組合，讓不同語言、不同平台上的系統得以相互溝通。但就開發Web Service而言，首

表1　Web Service成員說明

服務提供者	負責建立Web Service與提供服務使用的相關規範(WSDL檔)。
服務使用者	使用服務。
WSDL檔案	Web Services Description Language的縮寫，是描述服務如何使用的檔案，檔案內容以XML格式來描述，負責向服務使用者說明呼叫服務所需傳遞的參數、服務最後回傳內容等等。服務使用者只要取得WSDL就能了解Web服務的所有資訊，並且進一步使用該服務。
Proxy物件	透過解析WSDL檔，服務使用者便能建立Web Service的代理物件(Proxy)，代理物件具備有遠端Web Service全部的功能，但事實上它只是類似於搖控器的角色。
SOAP	Simple Object Access Protocol的縮寫，這是Web Service的通訊協定，協定中以XML來定義資料交換格式，使得無論是服務提供者或服務使用者有一個統一的標準來溝通，SOAP是整個Web Service最重要的核心，透過HTTP等通用的網路標準協定處理整個Web Service訊息的傳遞。

先要了解Web Service觀念和運作機制。筆者將透過圖1來介紹Web Service運作的架構，讀者需要注意的Web Service成員及其說明如表1所列。

圖1中3個子圖由上至下，依序就是Web Service的使用流程，服務提供者建立Web Service後，須提供WSDL檔給服務使用者，而服務使用者解析WSDL檔後產生Proxy物件，服務使用者透過呼叫Proxy物件中的方法，就類似我們使用無線遙控器操作家電一樣的道理，以間接的方式讓遠端的Web Service工作。

在PB中開發Web Service的3個步驟

在實作上，PB11中無論是建立亦或是部署，都由精靈導引產生，過程與傳統開發PB NVO物件方法非常類似，而且用來描述Web Service的WSDL檔案，也由PowerBuilder自動產生，所以整體步驟非常簡單，一共只需3個步驟：

- 步驟1：新增.NET Web Service Target，如圖2。
- 步驟2：依照PB傳統開發NVO(Non-Visual Object）物件的方式，在元件中建立方法並加入程式邏輯(如圖3）。
- 步驟3：程式邏輯撰寫完畢後，接下來是Web Service的部署。PB11 .NET Web Service Target有專用的Project物件來進行部署工作，

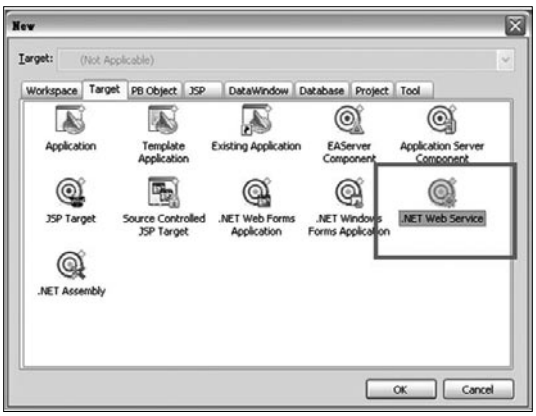

圖2　新增.NET Web Service Target。

圖3　建立Web Service方法並加入程式邏輯。

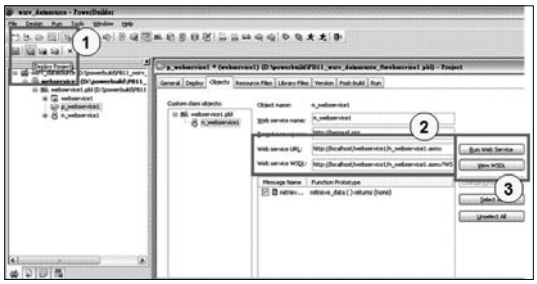

圖4　部署Web Service。

適當的完成Project中各屬性頁的設定後，按下部署鍵，部署工作就能自動進行(如圖4編號1）。

除了部署的操作流程比前幾版的Web Service來的更為簡易外，筆者這裡還要做幾點補充：首先，PB11主要是將NVO物件包裝成.NET語言的Web Service，包裝的過程是由編譯器自動執行，其中Power scripts會被編譯成C#語法的程式碼，所有PB的資料型態也會自動轉成對應的C#資料型態，最後部署到IIS上。

其次，.NET WebService Project也提供了測試網頁來預覽部署結果是否成功。最後關於服務提供者必須提供的WSDL檔，這個檔案也是由Project自動產生，預設是放在IIS伺服器上，URL位址會以「http://....?WSDL」格式呈現在Project的web service WSDL屬性欄位中，PB程式設計師只要將該URL位址轉交給服務使用者即可，如圖4中的編號2紅框。

其中提供測試網頁來預覽部署結果對於Web Service來說非常重要，因為過去Web

Service開發在部署完成後，往往都必須接著開發Web Service client，也就是我們先前所述的服務使用者，如此一來才能測試，得知Web Service開發是否成功，非常麻煩。但在圖4的編號3紅框區域，我們只要點選「Run Web Service」鈕就能透過Project自動產生的網頁來驗證Web Service是否正常。

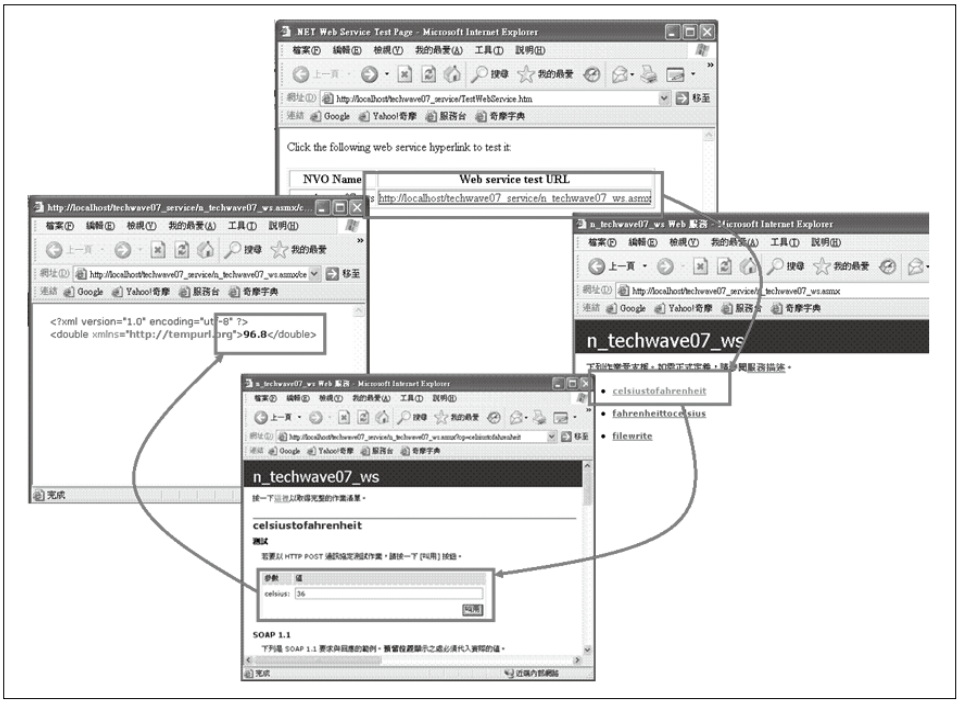

圖5　產生測試網頁，協助驗證開發是否成功。

筆者以攝氏轉換華氏溫度的Web Service為例，測試網頁會提供參數輸入欄位，驗證時只要輸入參數，如：攝氏36度，網頁便能執行並回傳Web Service執行結果，測試Web Service的流程如圖5所示。

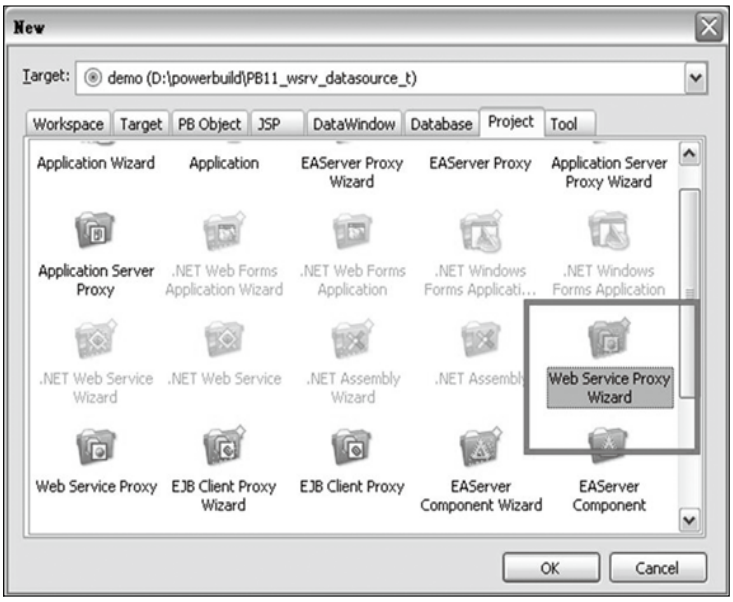

圖6　建立Web Service Proxy物件。

PB11程式呼叫Web Service的3個步驟

如同建立Web Service的簡潔，PB11呼叫Web Service的語法、步驟也設計成非常簡易，由於Web Service本身具備有跨語言的特性，在PB11中我們可以直接在傳統的PB程式中去呼叫、使用Web Service。筆者也將呼叫Web Service的流程區分3個步驟，讓讀者們可以了解呼叫Web Service所涉及的3個物件(WSDL檔案、Proxy物件、SOAP相關物件）要如何使用：

- 步驟1：新增一個「Web Service Proxy Project」。這個Project物件的功用很單純，就是專門負責解析WSDL檔，並建立Web Service在客戶端的proxy物件，新增的方式如圖6。

Web Service Proxy新增的過程是由精靈引導，最關鍵的步驟是必須將WSDL檔案的URL位址或檔案實體目錄位址輸入該Project中，在精靈引導的流程中，程式設計師可以決定使用最新的.NET WSDL Engine或較舊版本的EasySoap WSDL Engine(如圖7右方編號2），這2個版本差別在於所產生的Proxy物件是不是必須運作在.NET Framework環境上，其中.NET WSDL Engine所建立的Proxy物件，雖然支援較多新功能和範圍較寬的資料型態，但應用程式最終必須部署在裝有.NET Framework 2.0的環境上，應用程式中的Web Service才能正常運作，而選擇EasySoap則不必。

建立完成後，執行Proxy Project物件就會

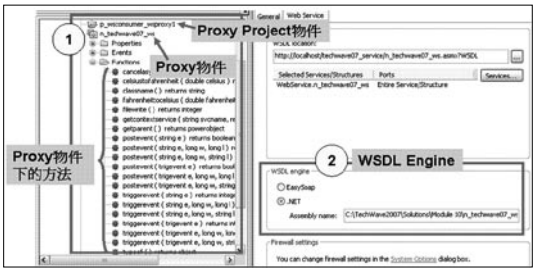

圖7 Web Service Proxy Project Painter與Proxy物件。

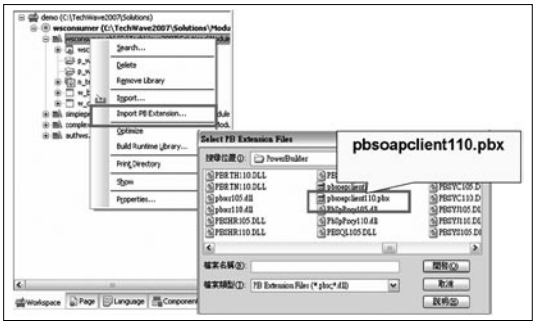

圖8 加入SOAPconnection與SOAPexception物件。

建立Web Service的Proxy(如圖7編號1紅框所標示內容），值得注意的是Proxy物件會提供Web Service所有可用的方法。

- 步驟2：依照不同的WSDL Engine設定加入不同版本的SOAPconnection與SOAPexception物件，這2個物件分別負責建立連線與控制SOAP相關的例外處理，其中如果選擇.NET WSDL Engine來建立Proxy的話，必須加入pbwsclient110.pbx，而透過Easysoap所建立的Proxy則必須選擇pbsoapclient.pbx，如同我們先前所述，這個選擇關係到整個應用程式是否必須運作在.NET Framework環境上有關。
- 步驟3：撰寫程式碼。在程式中呼叫Web Service，程式碼並不複雜，如圖9編號2

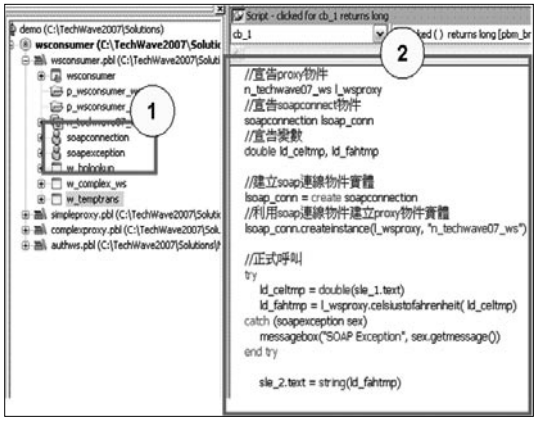

圖9 在程式中呼叫Web Service。

圖10 Web Service執行結果。

紅框所示，其中編號1部份為步驟2加入SOAPconnection與SOAPexception物件的結果。程式執行結果如圖10。

DataWindow Web Service Data Source

除了上述的介紹的.NET Web Service Target外，PB11還在DataWindow中加入對於Web Service的支援，在過去DataWindow的資料來源主要是來自於關聯式資料庫、外部文字資料兩種，資料的處理通常必須涉及關聯式資料庫的連線資訊，且往往必須在客戶端安裝資料庫連線的驅動程式檔，使得前端管理較為複雜。

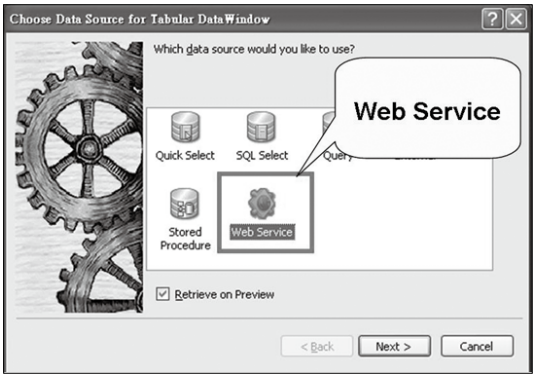

圖11　DataWindow物件可以Web Service當作資料來源。

DataWindow對Web Service的支援可以避免上述的麻煩，如果在DataWindow中使用Web Service當作資料來源，客戶端既不需要安裝任何的資料庫連線軟體，也毋須了解任何與資料庫連線有關的資訊，且除了讀取資料外，資料的插入、刪除與更新等功能，以Web Service為資料來源的DataWindow都有支援，更重要的是資料來源將不再受限制。

PB11中新增DataWindow時，在資料來源選擇視窗中多了一個Web Service選項(如圖11），選擇該選項則需輸入WSDL檔案的Web URL位址或實體檔案目錄。

選擇Web Service為資料來源的DataWindow，除了OLE 2.0與RichText兩種表現樣式不支援以外，其他的樣式的DataWindow皆可支援。需要注意的是，使用Web Service作為資料來源的DataWindow會連帶產生一個.NET Assembly物件，這個Assembly的角色是負責在runtime時與Web Service介接的介面，因此這也代表此類DataWindow應用必須運作在.NET Framework環境上。

除此之外，其它後續操作與一般使用SQL語句讀取資料的DataWindow物件編輯方式完全相同，程式設計師只需要以DataWindow Painter編輯呈現的樣式就行了，Web Service DataWinodw Painter編輯視窗如圖12所示。

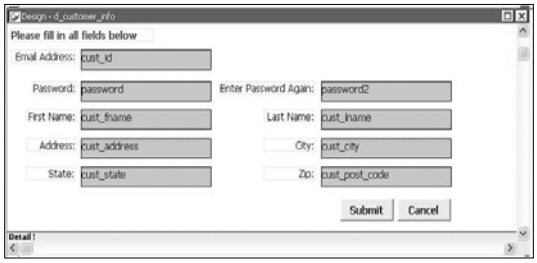

圖12　以Web Service為資料來源的DataWindow物件編輯視窗。

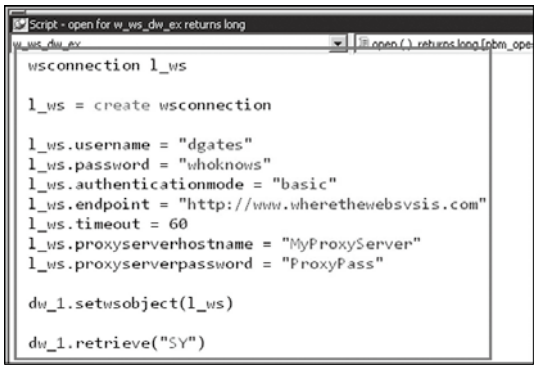

圖13　WSconnection物件可攜帶必要的參數到服務端。

Web Service DataWindow在程式碼撰寫方則面更為簡潔，使用此類DataWindow原則上無需宣告資料庫連線字串，可直接讀取資料。但萬一Web Service來源有特殊的認證或Proxy設定

需求，PB11中也新增了WSconnection物件能夠處理這些工作，無論是Web Service認證帳號、密碼、endpoint、Proxy Name等資訊，都可當作WSConnection的參數傳遞到服務端，程式撰寫方式類似於Transaction Object，如圖13所示。

Web Service DataWindow除了取得資料外，一般的insert、Delete、Update功能它也都支援，只要Web Service本身有提供資料更新相關函數即可。例如在PB11開發環境中，新增一個功能，可以設定Web Service DataWindow的Update相關動作，如圖14。

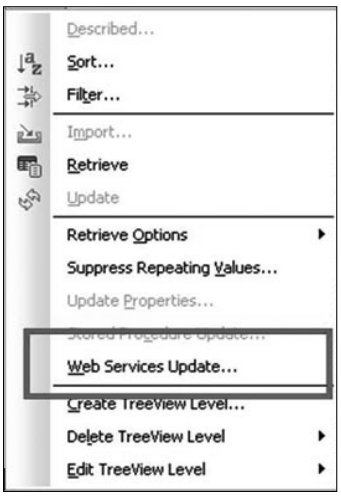

圖14 針對Web Service DataWindow，PB11新增了update屬性設定功能。

點選這個功能，程式設計師可以將Update、Insert、Delete等DataWindow常用的功能對應到Web Service所提供的方法，

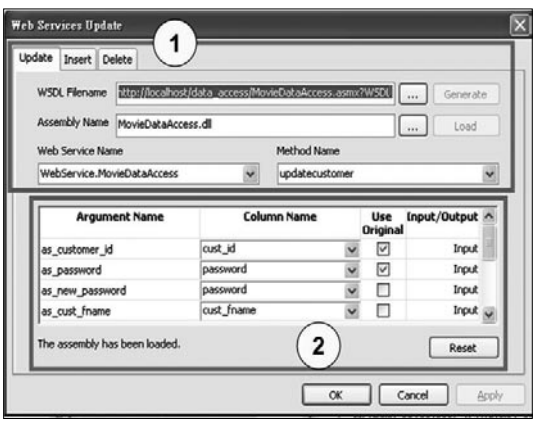

圖15 update屬性設定功能視窗能夠設定DataWindow去對應Web Service所提供的Update、Insert、Delete方法。

如圖15編號1的區塊，讀者們可以看到Web Service Update設定共分3個頁籤，每個頁籤上的欄位長相基本上都相同。因此以Update為例，程式設計師只要選定WSDL檔與相對應的Update Method Name，接下來便可以在編號2的區塊中去設定更新時對應的資料欄位。如此的設定可確保了DataWindow物件保有它一致的使用方式，如Update時，在程式中可依傳統方式下簡單的update( ) 指令便能夠輕易完成工作。

總結

Web Service帶來的不僅僅只是新技術、新名詞，以這些年的發展來看，依靠Web Service技術去整合其他的系統資源會是一個必然的趨勢，例如現在SOA不少產品，就用了不少Web Service的技術。PB在11這個版本中以全新的設計，為Web Service開發量身打造出更簡易的開發流程，相信對PB開發人員有比以往更高的生產力。

下一篇則是這個系列報導的最後一篇，筆者將會說明PowerBuilder .NET Assembly的開發與使用。Assembly正如其名，是零組件，概念上是一塊塊的程式積木，.NET程式設計師可以透過建立或組合不同的Assemblies，便能夠像堆積木一樣，快速的建立一個應用系統。PB11中能夠使用或建立這個類型的檔案，實質的意義上就如同為PB程式設計師開啟一扇通往.NET世界的大門，就擴充性而言，打破了PB運作的界線，是一加一大於二的結果，相信對於PB的開發人員，會有一些新的想法與激盪！

責任編輯／洪羿漣