Visual Studio 2005網站專案模型

ASP.NET 2.0 專案的部署問題

文／李明儒

Visual Studio 2005推出了嶄新的網站專案模型：Web Site Project，方便的即時編譯特性讓許多開發者耳目一新，但隨著系統成熟，面對部署及持續更新的需求，這個新專案型別的缺點開始一一浮現…。

使用Visual Studio.NET 2003的開發者接觸Visual Studio 2005後，會發現ASP.NET 1.1專案被自動轉換成全新的專案格式（Web Site Project），有別於以往，操作開發起來更為簡便，例如：

1. 專案檔（.csproj、.vbproj）不見了，直接開啟Web所在的資料夾就可以編輯專案，不需要先掛在IIS上，也不需要安裝FPSE（FrontPage Server Extension）。
2. 「.cs／.vb」檔修改儲存後就直接生效，不需建置（Build）就可執行看結果。
3. 引用Partial Class觀念，Server-Side程式不再有OnInit、InitializeComponent等IDE自動產生的WebControl物件宣告指令碼。
4. 同一Web專案中可以C#與VB.NET並存（雖然這不像是個好主意）。

方便之餘，我們也開始注意到一些略為不便的差異，例如：與頁面無關的獨立Class .cs／.vb檔必須集中放在App_Code目錄下、無法將特定的目錄及檔案排除在專案之外…等等。但最嚴重的問題，卻要到專案部署階段才會現身。

如果不想讓原始碼在正式機上露臉，VS 2005提供了Publish Web Site功能可預先建置DLL檔後再部署。但每次編譯（Compile）的DLL檔名會夾帶隨機產生的雜湊碼，ASPX<@Page>宣告則會配合隨機檔名修改而每次不同；系統上線後持續性的修Bug、加功能的工作是少不了的，每次建置出來的檔案不同讓這類小幅更新工作頓時變得棘手。當部署與維護的不便逐漸蓋過開發測試時的便利性，開發社群的不滿漸漸壓過原先的喜悅，微軟面對這波聲浪，採取的策略是亡羊補牢，迅速地提供了配套解決方案以及新的專案選擇。

在接連兩期文章中，我們將簡介Web Site Project的特性，探討部署時會遭遇的問題，並提供解決方案建議。

Web Site Project的特性

Web Site Project是Visual Studio 2005

所引進的新一代專案類型，當初的著眼點是要改良VS.NET 2003網站專案的一些缺陷。這些改良更動了底層的編譯程序及專案模型，帶來便利性的同時，也產生了一些副作用。以下對幾個主要的特性改變加以說明，談一談它們的優點及可能造成的影響。

1.Project Based -> File Based

在ASP.NET 1.1專案中，每個網站應用專案會有一個專屬的專案檔（.csproj或.vbproj），其中表列了該專案所包含的ASPX、ASCX、CS/VB、HTM、CSS、JPG、GIF…等等大小檔案。在Web Site Project中，則不再需要.csproj或.vbproj檔。原來Web Site Project取消了透過檔案維護檔案清單的做法，將整個專案目錄下的所有檔案，自動視為專案的一部分。同時，它也擺脫了對IIS及FPSE的需求，只要能取得網站的完整目錄，就可以直接開啟編修整個專案。即使開發機器上沒有安裝IIS，VS 2005內建的Web Development Server，也能啟動一個隨機Port Number的迷你Web Server，用來測試及偵錯網站程式。

這些特性簡化了新建及接手專案開發工作的複雜度，但也產生一些缺點，例如：有些檔案基於管理方便才放在網站目錄下，不需納入專案中被編譯，在ASP.NET 1.1 Project中可以將它們排除（Exclude）在專案外，Web Site Project則無從支援。

2.Code-Behind -> Code-Beside

以往ASP.NET 1.1 Project中，ASPX中需要透過<asp:TextBox Id=”Text1” runat=”server”>的方式宣告Web Control，而對應的Code-Behind檔中則還有OnInit、InitializeComponent兩個IDE自動產生的Method，用來宣告「TextBox Text1=new TextBox ();」，二者性質重複，也就很容易發生ASPX與Code-Behind程式不一致的情況。因此在VS.NET 2003中，InitializeComponent的Method前後會有註解宣告它們由Web Form Designer自動產生，請勿隨意更動。使用者在拖拉Web Control後，則常需要在Design/Source兩個Tab間切換，好讓ASPX的變動反應到Code-Behind端。

Web Site Project引進了新的Code-Beside模式，利用Partial Class的觀念，InitializeComponent部分的程式碼改成在使用者存取ASPX時才自動產生，現場編譯。如此，沒有ASPX與Code-Beside宣告同步的問題，Code-Beside程式中不再出現這些例行化的瑣碎程式碼，讓開發人員可以100%聚焦在商業邏輯上。

只是這種自動產生與即時編譯的概念，顛覆了過去所有aspx.cs會被編輯在同一個DLL中的概念。亦即WebForm1.aspx.cs與WebForm2.aspx.cs會分別獨立編譯，過去從WebForm1去參照WebForm2中Public Member的技巧變得不可行；而InitializeComponen的完全自動化，以往由Code-Behind程式動態建立、載入WebControl、UserControl的進階技巧變得困難重重。

3.編譯（Compilation）時機的改變

在ASP.NET 1.1專案中，Code-Behind部分的.cs／.vb程式碼必須先編譯成一個DLL檔，放在BIN目錄下。而在Web Site Project的預設模式中，.cs／.vb不需事先編譯，而會在aspx被存取時才會動態編譯（稱

之為隨選編譯On-Demand Compilation，或即時編譯Just-In-Time Compilation）。這個新設計帶了一些好處，例如：當我們在做Server-Side程式碼除錯時，只需修改後存檔，接著用瀏覽器重新載入，馬上就可以看到結果，不像過去必須重新編譯整個網站專案，再重新啟動偵錯程序，方便許多。

因此有個觀念的改變值得注意，當Web Site Project以偵錯模式執行時，雖會對整個網站專案進行完整的建置（Build）作業，純粹只是為了檢測程式碼是否能順利完成編譯罷了，並不在於產生DLL供後續使用；待瀏覽器開啟要Debug的網頁時，還是得重新編譯Code-Beside檔案。由於Web Site Project編譯的範圍涵蓋了ASPX Parsing／ASPX Inline Code／Code-Beside Code，建置速度比ASP.NET 1.1 Web Project慢上許多，每次Debug前耗費大量時間進行非必要性的編譯動作，對大型的專案來說十分不划算。因此我們可以透過專案屬性設定，設定偵錯執行時完全不做編譯或只編譯單一網頁，另外還可指定建置Solution時排除Web Site Project（如圖1），如此可大幅減少開發測試期間的無謂等待。

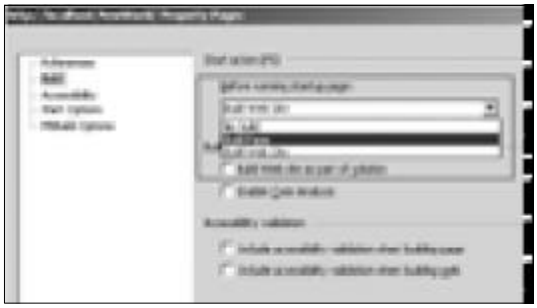

圖1　透過修改Web Site Project屬性改善Debug效率。

4.App_Code目錄

VS 2005自動升級ASP.NET 1.1專案後，你會發現原本散落在專案各角落的獨立Class檔案，會被自動集中到一個名為App_Code的目錄下。原因如同先前所提的隨選編譯概念，網頁相關的Code-Beside程式會在網頁被存取時才進行編譯，因此跟網頁未直接關聯的程式碼就必須另外處理。放在App_Code下的程式碼，會被另外單獨編譯，無論我們如何調整圖1的設定，App_Code裡的程式還是會在建置時，甚至VS 2005 IDE操作期間重新編譯（因為其中可能包含自訂的控制項，設計期間就會用到）。在App_Code下的程式可以透過在web.config指定codeSubDirectories的方式，同時混用VB.NET或C#所寫的類別，只是混用多種語言一般會衍生後續維護上的困擾，能免則免!

由於App_Code編譯時機較為頻繁，如果放了太多的類別，會影響建置及VS 2005操作的速度。此時可考慮將其獨立成一個Class Library專案，對整體操作的順暢度將有所幫助。

5.部署觀念的改變

誠如先前所討論的，Web Site Project的建置模式已趨於隨選編譯。但所謂隨選編譯的前題是，Code-Beside的.cs／.vb檔案必須存在於IIS網站目錄下，才能在使用者存取該網頁時即時進行編譯。將原始程式碼一併部署到正式營運主機，會大幅提高程式邏輯外洩的風險，顯然違背一般的資安原則，正式運作的系統還是應回歸將程式編譯為DLL後再部署的做法。針對這種預先編譯的需求，VS 2005內建了Publish Web Site這項功能，可以對Web Site Project進行預先編譯（Precompile）的動作，再一併將ASPX、DLL檔案複製到特定目錄下。只是在複製時，我們無法選擇只更新其中某幾個

檔案，它會一律將原有目錄清空再全部重新寫入（包含原本的web.config、額外複製過去的檔案都會遺失）。因此千萬不要用它直接更新正式的網站目錄，以免發生悲劇。

比較可行的做法是先Publish到本機目錄，再手動將要更新的檔案及DLL複製到目的主機上。但另外一個大問題是，Publish Web Site會為每個資料夾產生一個檔名包含隨機雜湊碼的DLL檔，且每次編譯的雜湊值都不同。當我們只需少量更新時，得一一比對找尋ASPX對應的DLL，是件頭痛的事。Publish Web Site雖提供了選項，支援「每個網頁擁有固定檔案名的獨立DLL」（如圖2），但也未能徹底解決問題。關於Web Site Project部署的一籮筐的問題，稍後再一一細數。

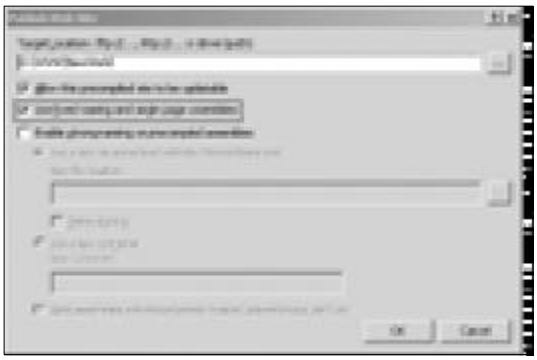

圖2　Publish Web Site可指定產生固定的DLL名稱。

Web Site Project的部署難題

由以上的說明，Web Site Project最重要的變革就在於編譯模式的調整，它帶來了一些便利性。例如：Server-Side程式的更新不需要編輯、免除了ASPX與Server-Side程式間控制項宣告的同步問題，但這些設計較偏重開發測試階段的改良，並未周詳思考部署時可能遭遇的嚴重問題，VS 2005內建的部署功能（Publish Web Site）也未直實地切合實務需求。

隨選編譯模式下，直接更改Server-Side檔案，不需重新編譯及更換DLL檔的想法很酷，但前題是必須將原始程式碼儲存於正式主機上，大大違反了一般公認的資安原則。若要避免隨選編譯模式會曝露原始碼的風險，預先編譯成DLL再部署的做法是較合宜的策略。

VS 2005內建的Publish Web Site的功能，可以預先將Code-Beside原始碼預先編譯成DLL，甚至將ASPX檔案內容也一併藏入DLL中。但是，Publish Web Site卻有幾項頗為嚴重的缺點：

1. 編譯完成的檔案會完全覆寫原有的目錄結構，無法指定只更新那些檔案。實務上，開發、測試、正式台的web.config多半不同，網站目錄下的子目錄、檔案也可能略有差異。由於Publish Web Site會刪除整個目的目錄後全新重建（包含web.config），除了第一次全新部署時可以使用，日後的更新就只能先寫到本機目錄，再手工更新檔案。
2. Publish Web Site功能預設每次編譯出來的DLL檔名中都會夾帶隨機產生的雜湊碼，雖然VS 2005會自動修改ASPX，讓<@Page>的inherits屬性自動指向新產生的DLL檔名（如圖3）。但如果要手工更新某個ASPX的Code-Beside程式，就得先查看ASPX以找出它對應的DLL檔，將該檔案更新至正式機，再將同一目錄下的ASPX也一併更新（因為預設同一資料夾的Code-Beside程式會編譯成同一個DLL檔），如果正式機上的舊版DLL要一併移除乾淨，還得先由正式版上的原有ASPX

追查檔名，過程十分繁瑣。

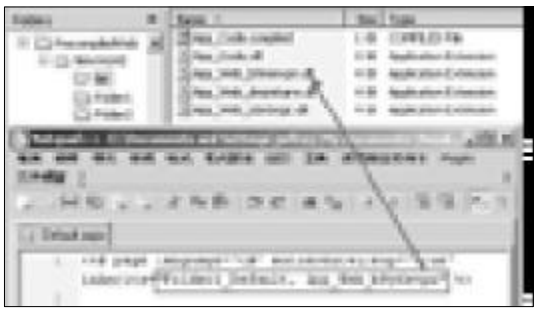

圖3　VS 2005會修改ASPX內容指向隨機產生的DLL檔名。

3.為了解決前述隨機檔名的問題，Publish Web Site提供了一個選項，可以設定DLL採用固定檔名不再隨機產生，但是該選項會一併將DLL拆分成每個ASPX網頁專屬一個DLL檔。雖然這能減少了更新單一ASPX與DLL的複雜度，更換DLL時，也不會影響到其他的ASPX；但在大型Web Application中，可能包含了數百上千個ASPX檔；網站運作過程要涉及這麼多個DLL的載入與管理，對效能肯定有害。（如圖4）

圖4　固定檔名時為每個ASPX一個DLL檔。

以上的所提的問題絕非Web Site Project的原罪，也並非不能解決。微軟翻修了VS 2005的編譯功能，並推出了MSBuild，這個新一代的建置引擎，功能強大，彈性十足，要克服以上的問題輕而易舉（關於MSBuild的介紹與應用說明，讀者可以參考RUN!PC 146期第175頁沈炳宏先生的文章）。Web Site Project所遭遇的部署問題，我認為肇因於制定功能規格時，過於著重開發體驗、測試過程的改良，卻忽視了部署實務的需求。而Publish Web Site這個功能，介面上所能提供的選擇太少，辜負了強大的MSBuild引擎，迫使開發／部署人員得在少得可憐的幾個選項間左右為難。

微軟的回應

隨著以ASP.NET 2.0開發的專案進入測試、部署階段，Web Site Project在部署上的缺失也逐漸浮現，並在開發人員社群中引發熱烈的討論（當然不乏怒火中燒的猛烈批評），微軟傾聽了開發社群的心聲，面對問題，並陸續提出了解決方案。

首先，一個更具彈性的建置選項設定介面以Add-In方式推出，以Web Deployment Project的型式讓我們可以對Web Site Project編譯與部署的細節進行微調，充分發揮MSBuild的彈性與擴充性，以符合各式部署需求。

當然，Web Site Project架構的改變，影響的並不只部署而已。許多開發者還是習慣ASP.NET 1.1時代的專案模式：以.csproj/.vbproj控制專案細節的模式、全部的Code-Bind程式建置成單一的DLL；而一些涉及網頁物件繼承、動態控制項建立等進階技巧的網頁，移殖到Web Site Project時或多或少需要修改。因此另一種新的專案類別—Web Application Project也問市了，讓開發者可以依循原先ASP.NET 1.1網站專案的概念來開發ASP.NET 2.0專案（Web Application Project已包含在VS 2005 SP1中）。下一期，我們將介紹如何應用Web Deployment Project來滿足部署上一些常見的需求。

責任編輯／洪羿漣