第三章　服務合約 Service Contract

在分散式運算環境下，利用網路服務做為元件的軟體開發流程中，程式設計人員除了在單機軟體開發就存在的問題之外，最容易遇到的問題就是所使用的網路服務發生了問題。由於是第三方所提供的服務，發生了問題除了回報給服務提供者之外，也無從檢視問題發生的原因。在使用網路服務時會發生問題，原因除了可能出自於第三方服務本身之外，網路品質也是要考慮的因素，像是網路壅塞或是網路不穩定都有可能造成服務時間逾時或是資料傳輸失敗或傳輸不完全。而這些問題一再困擾著程式設計人員，他們在開發程式之餘還得一邊思考著要如何進行有效的例外處理，但是由於使用的是第三方提供的服務，在介面合約沒有清楚訂定的前提下，程式設計人員對於要處理的例外更是一無所知，更別提有效的例外處理了。

我們所提出的服務合約其實是一個很簡單的概念。大家都知道第三方服務的品質不具保障，但是基於軟體元件的原理，如果能夠良好地進行元件的安裝與移除，大家還是能夠接受第三方所提供的服務元件。而針對品質不具保障的解決方法，最理想的狀況是第三方在提供服務元件的同時，便已經經過了繁瑣但精確的元件測試，確保該元件的品質能達到水準之上的良好，但是所需要的成本可能會倍增。另外，在使用網路服務時會發生的問題，還有可能是因為服務使用者的使用不當，例如傳入不合法或服務無法處理的參數，雖然服務提供者在提供服務的同時應該要考慮的參數合法性的處理，但是在服務提供者進行這些慎密的容錯處

理時，時間與人力的成本也在急劇上升。

目前以網路服務為元件進行軟體開發的模式，是將網路服務當成傳統的函式呼叫。若服務發生錯誤，則會回傳特定的錯誤代碼。然而這些錯誤，卻又有可能來自於網路等等奇怪的因素。因此使用網路服務的程式設計人員除了原來網路服務就應該檢測的部分（例如回傳值是否正常）之外，可能還要額外的檢測這個網路服務是否超過一段時間沒有回應（可能因為網路斷線或是其它因素）等等可能的各種錯誤。我們認為這些歧異的例外處理（包含元件本身的可預期錯誤，非預期錯誤，以及網路問題等等）應該整合到單一的例外處理機制中。

於是我們提出了服務合約的想法，讓服務提供者在提供服務的同時也一併提供服務合約，而服務請求者在使用服務的同時，也必須滿足服務合約才能進行服務請求。服務合約的內容包括傳入參數與傳回值的資料型態，以及前後置條件(pre/post-condition)和服務回應時間，關於服務合約的細節在後面的小節裡會有更詳細的說明。為了不增加軟體開發者的困擾，我們在提出服務合約的同時也將一併提供一項工具來協助軟體開發者自動處理服務合約的部分，期望有一天服務合約能被 W3C 納入 WSDL 的正式標準，並且被整合開發環境所支援，在處理匯入的 WSDL 文件同時也能一併處理服務合約的部分。

第一節　語法與文法

在這一個小節裡，我們將說明如何撰寫服務合約的內容。服務合約是由服務

提供者在提供服務的同時一併提供，因此希望服務合約能和服務的內容並存，但是當然不能影響原服務的內容，所以服務合約將以程式註解的形態存在，如圖 3 所示。

圖 3 是一個完整的含有服務合約區塊的例子：這是一個除法運算的服務，先不管服務合約區塊的話，第 11 行就是服務的主要內容，而第 12 行的 return S_OK 是指服務運作正常結束，所以回傳一個代表正常結束的 HRESULT 值。兩個傳入服務的參數 a 及 b 的資料型態為整數 int，而服務傳回值 ans 的資料型態也是整數 int，我們就利用這個例子來說明服務合約的語法及文法吧。

首先，服務合約區塊由 /*@CONTRACT 開頭，*/ 結尾，為一完整註解區塊，位置須在服務函式的左大括號後立即出現。

```
01  HRESULT divide (/*[in]*/ int a, int b, /*[out, retval]*/ int* ans)
02  {
03      /*@CONTRACT
04      @param: int;
05      @param: int;
06      @retval: int;
07      @require: _para1 != 0;
08      @ensure: _para0 == _return_ * _para1;
09      @RTT: 3000;
10      */
11      *ans = a / b;
12      return S_OK;
13  }
```

圖 3：服務合約區塊的格式

服務合約內目前暫定共有五項子項目，分別為@param、@retval、@require、@ensure、@RTT，均以“@”的符號開頭，以下分項敘述。

@param 及@retval 指的是 parameter 以及 return value。這兩者的格式是相似的，後面的參數定義傳入服務的參數以及服務傳回值的資料型態，目前接受的參數型態有：int、long、char、double、float 等原生型態再加上 BSTR，BSTR 是字串的一種，特點是在字串前會先有一個 4bytes 的整數表示字串長度（不含字串結尾的 byte 數），而字串本身是 Unicode 字元（每一字元 2bytes），可包含 null 字元(\0)，而字串結尾是兩個 null 字元(0x00)[16]。須注意的是，每個子項目後面只能接一個參數，而且不可以沒有，如果沒有參數或為 void，可不需列出該子項目。若有多個傳入服務的參數，則依順序將@param 分成多行撰寫，但代表服務傳回值的@retval 只能有一個。

@require 與@ensure 指的是前置條件的“需求”和後置條件的“保證”。這兩者的格式也是相似的，後面的參數須為一完整邏輯式。@require 為合約先備條件，用來檢查進行服務前的狀況是否符合條件，常用來檢查將傳入服務的參數是否符合服務所能接受的型態或範圍；而@ensure 為合約後備條件，用來檢查服務結束後的傳回值是否符合合約所制定的條件。在@ensure 的參數中可以使用_return_來代表服務回傳值，而兩者的參數均可以使用_para[0-9]來代表傳入服務的參數，次序由 0 開始，依序為_para0、_para1、…等，如此便能在合約先備或後備條件的檢查中，輕鬆的加入參數做為檢查的條件運算。要特別注意的是在@require 後面的

參數中，不可使用_return_，因為在檢查合約先備條件時，服務的傳回值是不存在的。而@RTT 是用來規範服務時間的，RTT 取自 Round-Trip Time，即封包來回的時間，後面的參數為一數字代表服務回應的時間，單位是毫秒。

以圖 3 來說，服務先備條件@require: _para1 != 0; 其中_para1 指的是第二個傳入參數 b 不可為零；而服務後備條件@ensure: _para0 == _return_*_para1; 裡的_para0 與_return_分別指的是第一個傳入參數 a 及服務傳回值 ans。接下來服務回應時間@RTT 的參數 3000，代表 3000 毫秒，也就是三秒鐘後如果服務還沒有回應，便視同服務逾時。

每一個子項目和參數之間須有一冒號(：)區隔，參數後須有一分號(；)作為項目結束，符號皆為半形字元。項目與項目之間的空白與換行符號可忽略，但為了程式以及服務合約的可讀性，建議服務提供者在撰寫服務合約的同時也能適時地加入空白以換行符號。表 3 為服務合約子項目的列表敘述。

表 3：服務合約標籤說明

合約標籤	說明
@param	傳入服務的參數型態，目前支援原生型態加上 BSTR。可有多項。
@retval	服務傳回值的參數型態，目前支援原生型態加上 BSTR。只能唯一。
@require	合約先備條件，在服務請求前進行檢查。須為一完整邏輯式。
@ensure	合約後備條件，在服務請求後進行檢查。須為一完整邏輯式。
@RTT	最大服務回應時間，單位為毫秒。

第二節　優點與缺點

服務合約一開始的構想是希望軟體開發者在利用網路服務作為元件進行軟體開發時，不管遇到的問題是服務元件本身的、外在的、可預期或不可預期的錯誤，甚至是網路等等任何奇形怪狀的原因所造成的，都能夠不被影響，並且能繼續正常地進行程序。而最顯而易見的問題便是服務逾時的問題，不管逾時的原因是因為服務設計不當或是網路品質影響等等，雖然一般來說作業系統對於網路連線的控管會有基本的連線逾時設定，但是以軟體開發而言，作業系統所限制的逾時時間太過冗長，如果每一次都要等候這麼久的時間才能判斷服務逾時的話，會顯得很沒有效率。

因此最先開發的優點是服務回應時間能夠讓服務提供者自定，服務合約透過@RTT 的制定，當服務運作的時間超過了@RTT 制定的時間，也就是服務回應時間逾時，服務請求者便會捕捉(catch)到合約例外，並繼續進行後續程序或是合約例外的處理。

除了逾時的問題之外，因為服務使用者的使用不當而造成服務發生問題的例子也屢見不鮮，而服務使用不當的例子最常見的莫屬傳入參數不被服務所接受，如型態不符或是超出範圍等等。

服務合約透過先備條件@require 的制定，在進行服務請求前便先行檢查，若發現不符合服務合約所訂定的先備條件，便不會進行服務請求。如此一來不但不

需要透過網路進行服務請求後，經由服務內繁瑣的檢查發現傳入參數不符合服務合約所訂定的先備條件之後才將請求駁回，也能有效降低服務提供者主機及網路的負荷。

而服務合約透過後置條件@ensure 的制定，在服務結束後可逕行檢查服務傳回值是否正確，減少因傳回值不如預期值所造成的程式問題，例如預期傳回值不為零而進行運算，但可能因傳輸不完全或傳輸失敗，甚至服務設計錯誤導致傳回值為零，如果因此造成其它關鍵性的錯誤，對程式設計人員而言更是難以發現並徹底解決的問題。

服務合約的格式是參考 WSDL 的格式，文件架構一律採用 XML 的格式。所以具備良好的可讀性及可擴充性。另外，開放的 XML 格式也歡迎第三方軟體的開發，更希望整合開發環境能因此更快速且容易地支援 WSDL 合併服務合約的處理。

雖然本研究致力朝向開放的標準，但仍有許多地方不夠周全，如本研究所提供的工具需事先定義的合約標籤，諸如@param、@retval、@require、@ensure、@RTT 等標籤是固定的，不能夠彈性地接受所有的自定標籤。但因為文件格式是開放的，所以也歡迎第三方自行發展相關軟體。除此之外，合約先備條件@require 與合約後備條件@ensure 的參數須為一完整邏輯式，此處定義亦不夠彈性，期望在未來能夠將文件格式修訂得更為彈性及便利。