1.1、Web Service基本概念

Web Service也叫XML Web Service WebService是一种可以接收从Internet或者Intranet上的其它系统中传递过来的请求，轻量级的独立的通讯技术。是:通过SOAP在Web上提供的软件服务，使用WSDL文件进行说明，并通过UDDI进行注册。

XML：(Extensible Markup Language)扩展型可标记语言。面向短期的临时数据处理、面向万维网络，是Soap的基础。

Soap：(Simple Object Access Protocol)简单对象存取协议。是XML Web Service 的通信协议。当用户通过UDDI找到你的WSDL描述文档后，他通过可以SOAP调用你建立的Web服务中的一个或多个操作。SOAP是XML文档形式的调用方法的规范，它可以支持不同的底层接口，像HTTP(S)或者SMTP。

WSDL：(Web Services Description Language) WSDL 文件是一个 XML 文档，用于说明一组 SOAP 消息以及如何交换这些消息。大多数情况下由软件自动生成和使用。

UDDI (Universal Description, Discovery, and Integration) 是一个主要针对Web服务供应商和使用者的新项目。在用户能够调用Web服务之前，必须确定这个服务内包含哪些商务方法，找到被调用的接口定义，还要在服务端来编制软件，UDDI是一种根据描述文档来引导系统查找相应服务的机制。UDDI利用SOAP消息机制（标准的XML/HTTP）来发布，编辑，浏览以及查找注册信息。它采用XML格式来封装各种不同类型的数据，并且发送到注册中心或者由注册中心来返回需要的数据。

1.2、XML Web Service的特点

Web Service的主要目标是跨平台的可互操作性。为了实现这一目标，Web Service 完全基于XML（可扩展标记语言）、XSD（XML Schema）等独立于平台、独立于软件供应商的标准，是创建可互操作的、分布式应用程序的新平台。因此使用Web Service有许多优点:

1、跨防火墙的通信

如果应用程序有成千上万的用户，而且分布在世界各地，那么客户端和服务器之间的通信将是一个棘手的问题。因为客户端和服务器之间通常会有防火墙或者代理服务器。传统的做法是，选择用浏览器作为客户端，写下一大堆ASP页面，把应用程序的中间层暴露给最终用户。这样做的结果是开发难度大，程序很难维护。 要是客户端代码不再如此依赖于HTML表单，客户端的编程就简单多了。如果中间层组件换成Web Service的话，就可以从用户界面直接调用中间层组件，从而省掉建立ASP页面的那一步。要调用Web Service，可以直接使用Microsoft SOAP Toolkit或.net这样的SOAP客户端，也可以使用自己开发的SOAP客户端，然后把它和应用程序连接起来。不仅缩短了开发周期，还减少了代码复杂度，并能够增强应用程序的可维护性。同时，应用程序也不再需要在每次调用中间层组件时，都跳转到相应的"结果页"。

2、应用程序集成

企业级的应用程序开发者都知道，企业里经常都要把用不同语言写成的、在不同平台上运行的各种程序集成起来，而这种集成将花费很大的开发力量。应用程序经常需要从运行的一台主机上的程序中获取数据；或者把数据发送到主机或其它平台应用程序中去。即使在同一个平台上，不同软件厂商生产的各种软件也常常需要集成起来。通过Web Service，应用程序可以用标准的方法把功能和数据"暴露"出来，供其它应用程序使用。

XML Web services 提供了在松耦合环境中使用标准协议（HTTP、XML、SOAP 和 WSDL）交换消息的能力。消息可以是结构化的、带类型的，也可以是松散定义的。

3、B2B的集成

B2B 指的是Business to Business，as in businesses doing business with other businesses,商家(泛指企业)对商家的电子商务，即企业与企业之间通过互联网进行产品、服务及信息的交换。通俗的说法是指进行电子商务交易的供需双方都是商家(或企业、公司)，她们使用了Internet的技术或各种商务网络平台，完成商务交易的过程。

Web Service是B2B集成成功的关键。通过Web Service，公司可以只需把关键的商务应用"暴露"给指定的供应商和客户，就可以了，Web Service运行在Internet上，在世界任何地方都可轻易实现，其运行成本就相对较低。Web Service只是B2B集成的一个关键部分，还需要许多其它的部分才能实现集成。 用Web Service来实现B2B集成的最大好处在于可以轻易实现互操作性。只要把商务逻辑"暴露"出来，成为Web Service，就可以让任何指定的合作伙伴调用这些商务逻辑，而不管他们的系统在什么平台上运行，使用什么开发语言。这样就大大减少了花在B2B集成上的时间和成本。

4、软件和数据重用

Web Service在允许重用代码的同时，可以重用代码背后的数据。使用Web Service，再也不必像以前那样，要先从第三方购买、安装软件组件，再从应用程序中调用这些组件；只需要直接调用远端的Web Service就可以了。另一种软件重用的情况是，把好几个应用程序的功能集成起来，通过Web Service "暴露"出来，就可以非常容易地把所有这些功能都集成到你的门户站点中，为用户提供一个统一的、友好的界面。 可以在应用程序中使用第三方的Web Service 提供的功能，也可以把自己的应用程序功能通过Web Service 提供给别人。两种情况下，都可以重用代码和代码背后的数据。

从以上论述可以看出，Web Service 在通过Web进行互操作或远程调用的时候是最有用的。不过，也有一些情况，Web Service根本不能带来任何好处，Web Service有一下缺点：

1、 单机应用程序

目前，企业和个人还使用着很多桌面应用程序。其中一些只需要与本机上的其它程序通信。在这种情况下，最好就不要用Web Service，只要用本地的API就可以了。COM非常适合于在这种情况下工作，因为它既小又快。运行在同一台服务器上的服务器软件也是这样。当然Web Service 也能用在这些场合，但那样不仅消耗太大，而且不会带来任何好处。

2、 局域网的一些应用程序

在许多应用中，所有的程序都是在Windows平台下使用COM，都运行在同一个局域网上。在这些程序里，使用DCOM会比SOAP/HTTP有效得多。与此相类似，如果一个.net程序要连接到局域网上的另一个.net程序，应该使用.net Remoting。其实在.net Remoting中，也可以指定使用SOAP/HTTP来进行Web Service 调用。不过最好还是直接通过TCP进行RPC调用，那样会有效得多。

1.3、XML Web Service的应用

1.最初的 XML Web Service 通常是可以方便地并入应用程序的信息来源，如股票价格、天气预报、体育成绩等等。

2.以 XML Web Service 方式提供现有应用程序，可以构建新的、更强大的应用程序，并利用 XML Web Service 作为构造块。

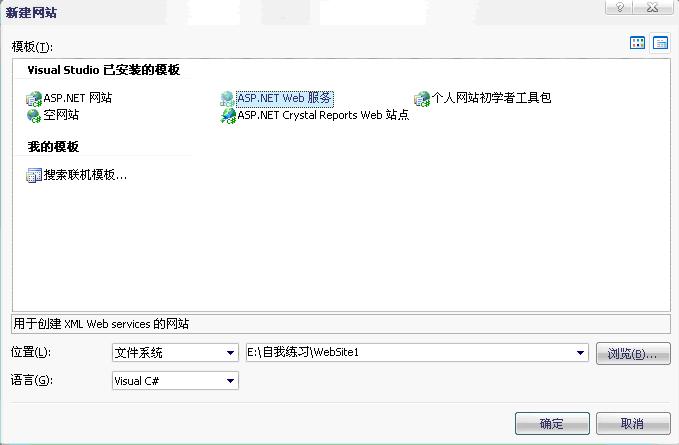
例如，用户可以开发一个采购应用程序，以自动获取来自不同供应商的价格信息，从而使用户可以选择供应商，提交订单，然后跟踪货物的运输，直至收到货物。而供应商的应用程序除了在Web上提供服务外，还可以使用XML Web Service检查客户的信用、收取货款，并与货运公司办理货运手续。

二、Web Service开发

.net平台内建了对Web Service的支持，包括Web Service的构建和使用。与其它开发平台不同，使用.net平台，你不需要其他的工具或者SDK就可以完成Web Service的开发了。.net Framework本身就全面支持Web Service，包括服务器端的请求处理器和对客户端发送和接受SOAP消息的支持。下来我们就一步一步的用Microsoft Visual Studio .net 20058（后面简称VS.Net 2008）创建和使用一个简单的Web Service。

2.1、用创建一个最简单的Web Service

首先，打开VS2005,打开"文件-新建-网站"，选择"ASP.NET Web服务"



查看Service.cs代码，你会发现VS.Net 2005已经为Web Service文件建立了缺省的框架。原始代码为：

view plaincopy to clipboardprint?  
using System;   
using System.Linq;   
using System.Web;   
using System.Web.Services;   
using System.Web.Services.Protocols;   
using System.Xml.Linq;   
[WebService(Namespace = "<http://tempuri.org/>")]   
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]   
// 若要允许使用 ASP.NET AJAX 从脚本中调用此 Web 服务，请取消对下行的注释。   
// [System.Web.Script.Services.ScriptService]   
public class Service : System.Web.Services.WebService   
{   
public Service () {   
//如果使用设计的组件，请取消注释以下行   
//InitializeComponent();   
}   
[WebMethod]   
public string HelloWorld()   
{   
return "Hello World";   
}   
}   
using System;  
using System.Linq;  
using System.Web;  
using System.Web.Services;  
using System.Web.Services.Protocols;  
using System.Xml.Linq;  
[WebService(Namespace = "<http://tempuri.org/>")]  
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]  
// 若要允许使用 ASP.NET AJAX 从脚本中调用此 Web 服务，请取消对下行的注释。  
// [System.Web.Script.Services.ScriptService]  
public class Service : System.Web.Services.WebService  
{  
public Service () {  
//如果使用设计的组件，请取消注释以下行   
//InitializeComponent();   
}  
[WebMethod]  
public string HelloWorld()  
{  
return "Hello World";  
}  
}

默认工程里面已经有一个Hello World的方法了，直接运行看看效果，



点击显示页面上图中的"HelloWorld"超链接，跳转到下一页面：



点击"调用"按钮，就可以看到用XML格式返回的Web Service结果下图。说明我们的Web Service环境没有问题，而且还初步接触了一下最简单的Web Service。

2.2、创建一个简单带有功能的Web Service

上面我们宏观的了解了webservice，其实它就是个对外的接口,里面有函数可供外部客户调用(注意:里面同样有客户不可调用的函数).假若我们是服务端,我们写好了个webservice,然后把它给了客户(同时我们给了他们调用规则),客户就可以在从服务端获取信息时处于一个相对透明的状态.即是客户不了解(也不需要)其过程,他们只获取数据.在代码文件里,如果我们写了一个函数后,希望此函数成为外部可调用的接口函数,我们必须在函数上面添上一行代码[WebMethod(Description="函数的描述信息")],如果你的函数没有这个申明,它将不能被用户引用.下来我们开始编写一个简单的Web Service 的例子。

先把默认的HelloWorld方法注释掉，简单的写了求加减乘除运算的四个方法；

view plaincopy to clipboardprint?  
using System;   
using System.Linq;   
using System.Web;   
using System.Web.Services;   
using System.Web.Services.Protocols;   
using System.Xml.Linq;   
[WebService(Namespace = "<http://tempuri.org/>")]   
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]   
// 若要允许使用 ASP.NET AJAX 从脚本中调用此 Web 服务，请取消对下行的注释。   
// [System.Web.Script.Services.ScriptService]   
public class Service : System.Web.Services.WebService   
{   
public Service () {   
//如果使用设计的组件，请取消注释以下行   
//InitializeComponent();   
}   
//[WebMethod]   
//public string HelloWorld()   
//{   
// return "Hello World";   
//}   
[WebMethod(Description = "求和的方法")]   
public double addition(double i, double j)   
{   
return i + j;   
}   
[WebMethod(Description = "求差的方法")]   
public double subtract(double i, double j)   
{   
return i - j;   
}   
[WebMethod(Description = "求积的方法")]   
public double multiplication(double i, double j)   
{   
return i \* j;   
}   
[WebMethod(Description = "求商的方法")]   
public double division(double i, double j)   
{   
if (j != 0)   
return i / j;   
else   
return 0;   
}   
}   
using System;  
using System.Linq;  
using System.Web;  
using System.Web.Services;  
using System.Web.Services.Protocols;  
using System.Xml.Linq;  
[WebService(Namespace = "<http://tempuri.org/>")]  
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]  
// 若要允许使用 ASP.NET AJAX 从脚本中调用此 Web 服务，请取消对下行的注释。  
// [System.Web.Script.Services.ScriptService]  
public class Service : System.Web.Services.WebService  
{  
public Service () {  
//如果使用设计的组件，请取消注释以下行   
//InitializeComponent();   
}  
//[WebMethod]  
//public string HelloWorld()  
//{  
// return "Hello World";  
//}  
[WebMethod(Description = "求和的方法")]  
public double addition(double i, double j)  
{  
return i + j;  
}  
[WebMethod(Description = "求差的方法")]  
public double subtract(double i, double j)  
{  
return i - j;  
}  
[WebMethod(Description = "求积的方法")]  
public double multiplication(double i, double j)  
{  
return i \* j;  
}  
[WebMethod(Description = "求商的方法")]  
public double division(double i, double j)  
{  
if (j != 0)  
return i / j;  
else  
return 0;  
}  
}

运行可以看到我们自己写的可以被调用的方法，如下图：



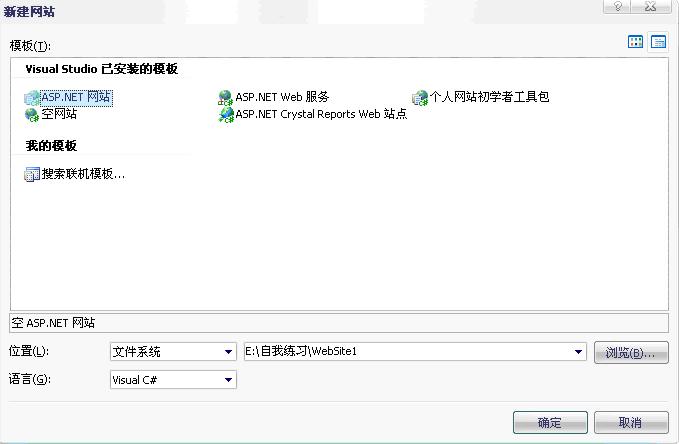
同样点击addition方法,进入addition方法的调用页。



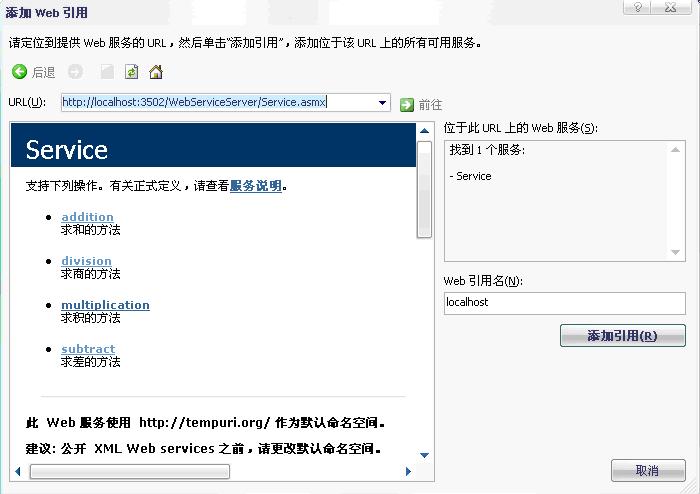
在参数上面输入参数i=3,j=3,如上图,点击调用,就可以看到用XML格式返回的Web Service结果(i与j相加的结果)下图

到这里,我们会发现,其实webservice并不是那么的神秘,它也不过只是个接口,对我们而言,侧重点就是是接口函数的编写.

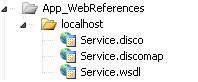
2.3、用ASP.NET调用Web Service  
首先，打开VS2005,打开"文件-新建-网站"，选择"ASP.NET网站"。



选好存储位置，语言后点击确定，进入默认页面。然后先添加Web引用，把WebService引到当前的工程里面。方法是：在资源管理器中点击右键，选择添加Web 引用，调出对话框：



在URL中填入，前面写好的WebService运行后浏览器上面显示的地址（即：WebService发布后的访问地址 ），点击"前往"按钮，如上图，就会显示出所引用的WebService中可以调用的方法，然后点击"添加引用"，就将webservice引用到了当前的工程里面 ，如下图，解决方案中会出现引进来的WebService文件



我们在这就练习调用webservice的四个方法，做一个简单的调用的例子，先在网站的前台添加几个控件，代码如下：

view plaincopy to clipboardprint?  
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="\_Default" %>   
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "<http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>">   
<html xmlns="<http://www.w3.org/1999/xhtml>">   
<head runat="server">   
<title>无标题页</title>   
</head>   
<body>   
<form id="form1" runat="server">   
<div>   
<asp:TextBox ID="Num1" runat="server"></asp:TextBox>   
<select id="selectOper" runat = "server">   
<option>+</option>   
<option>-</option>   
<option>\*</option>   
<option>/</option>   
</select>   
<asp:TextBox ID="Num2" runat="server"></asp:TextBox>   
<asp:Button ID="Button1" runat="server" Text="=" onclick="Button1\_Click" />   
<asp:TextBox ID="Result" runat="server"></asp:TextBox>   
</div>   
</form>   
</body>   
</html>   
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="\_Default" %>  
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "<http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>">  
<html xmlns="<http://www.w3.org/1999/xhtml>">  
<head runat="server">  
<title>无标题页</title>  
</head>  
<body>  
<form id="form1" runat="server">  
<div>  
<asp:TextBox ID="Num1" runat="server"></asp:TextBox>  
<select id="selectOper" runat = "server">  
<option>+</option>  
<option>-</option>  
<option>\*</option>  
<option>/</option>  
</select>  
<asp:TextBox ID="Num2" runat="server"></asp:TextBox>  
<asp:Button ID="Button1" runat="server" Text="=" onclick="Button1\_Click" />  
<asp:TextBox ID="Result" runat="server"></asp:TextBox>  
</div>  
</form>  
</body>  
</html>

然后在后台写调用的代码，调用之前和使用其它的对象一样，要先实例化，实例化的方法是localhost.Service a = new localhost.Service();然后就可以通过a来访问WebService里面提供的方法了。在这个例子里面，动态的创建了一个button控件来触发WebService的调用，后台代码如下：  
运行后可以看到效果，如下图所示，在前面两个Textbox里面输入两个操作数，在中间的下拉列表中选择操作符，然后点击"="号，将计算的结果输出到第三个Textbox里面。



而整个计算并不是在本地进行的，是在Web服务端进行计算的然后将结果通过XML返还给了调用方的，所以，在运行该程序的时候，WebService程序还必须启动，否则会报无法连接远程服务器的异常，如下图：



后台代码如下：

view plaincopy to clipboardprint?  
using System;   
using System.Configuration;   
using System.Data;   
using System.Linq;   
using System.Web;   
using System.Web.Security;   
using System.Web.UI;   
using System.Web.UI.HtmlControls;   
using System.Web.UI.WebControls;   
using System.Web.UI.WebControls.WebParts;   
using System.Xml.Linq;   
public partial class \_Default : System.Web.UI.Page   
{   
protected void Page\_Load(object sender, EventArgs e)   
{   
}   
protected void Button1\_Click(object sender, EventArgs e)   
{   
string selectFlag = selectOper.Value;   
localhost.Service web = new localhost.Service();   
if (selectFlag.Equals("+"))   
{   
Result.Text =(web.addition(double.Parse(Num1.Text),double.Parse(Num2.Text))).ToString();   
}   
else if (selectFlag.Equals("-"))   
{   
Result.Text = (web.subtract(double.Parse(Num1.Text), double.Parse(Num2.Text))).ToString();   
}   
else if (selectFlag.Equals("\*"))   
{   
Result.Text = (web.multiplication(double.Parse(Num1.Text), double.Parse(Num2.Text))).ToString();   
}   
else if (selectFlag.Equals("/"))   
{   
Result.Text = (web.division(double.Parse(Num1.Text), double.Parse(Num2.Text))).ToString();   
}   
}   
}   
using System;  
using System.Configuration;  
using System.Data;  
using System.Linq;  
using System.Web;  
using System.Web.Security;  
using System.Web.UI;  
using System.Web.UI.HtmlControls;  
using System.Web.UI.WebControls;  
using System.Web.UI.WebControls.WebParts;  
using System.Xml.Linq;  
public partial class \_Default : System.Web.UI.Page   
{  
protected void Page\_Load(object sender, EventArgs e)  
{  
}  
protected void Button1\_Click(object sender, EventArgs e)  
{  
string selectFlag = selectOper.Value;  
localhost.Service web = new localhost.Service();  
if (selectFlag.Equals("+"))  
{  
Result.Text =(web.addition(double.Parse(Num1.Text),double.Parse(Num2.Text))).ToString();  
}  
else if (selectFlag.Equals("-"))  
{  
Result.Text = (web.subtract(double.Parse(Num1.Text), double.Parse(Num2.Text))).ToString();  
}  
else if (selectFlag.Equals("\*"))  
{  
Result.Text = (web.multiplication(double.Parse(Num1.Text), double.Parse(Num2.Text))).ToString();  
}  
else if (selectFlag.Equals("/"))  
{  
Result.Text = (web.division(double.Parse(Num1.Text), double.Parse(Num2.Text))).ToString();  
}  
}  
}

到此一个一个简单的WebService的开发和调用就已经完成了，在实际应用中可以根据自己的需要，写一些功能强大的，复杂的WebService，不管多么复杂，整个流程都是这样的。

转载地址：<http://blog.csdn.net/h0322/archive/2009/11/07/4776819.aspx>

本文来自CSDN博客，转载请标明出处：<http://blog.csdn.net/wl_ldy/archive/2010/05/23/5618626.aspx>