1. 什么是 webservice

从外表上看，Web service 就是一个应用程序，它向外界暴露出一个能够通过Web 进展调用的 API。这就是说，你能够用编程的方法通过Web 来调用这个应用程序。

对 Web service 更准确的解释： Web services 是建立可互操作的分布式应用程序的平台。作为一个 Windows 程序员，你可能已经用 COM 或 DCOM 建立过基于组件的分布式应用程序。COM 是一个格外好的组件技术，但是我们也很简洁举出COM 并不能满足要求的状况。Web service 平台是一套标准，它定义了应用程序如何在Web 上实现互操作性。你可以用任何你宠爱的语言，在任何你宠爱的平台上写 Web service ，只要我们可以通过 Web service 标准对这些效劳进展查询和访问。

不管你的 Web service 是用什么工具，什么语言写出来的，只要你用SOAP 协议通过 HTTP 来调用它，总体构造都应如以下图所示。通常，你用你自己宠爱的语言(如 VB 6 或者 VB.NET)来构建你的 Web service，然后用SOAP Toolkit 或者.NET 的内建支持来把它暴露

给 Web 客户。于是，任何语言，任何平台上的客户都可以阅读其WSDL 文档，以调用这个 Web service。客户依据 WSDL 描述文档，会生成一个 SOAP 恳求消息。Web service 都是放在 Web 效劳器 (如 IIS) 后面的，客户生成的 SOAP 恳求会被嵌入在一个 POST 恳求中，发送到 Web 效劳器来。Web 效劳器再把这些恳求转发给Web service 恳求处理器。对 VB 6 程序来说，Web service 恳求处理器是一个与 SOAP Toolkit 组件协同工作的 ASP 页面或 ISAPI extension。而对 VB.NET 程序来说，Web service 恳求处理器则是一个.NET Framework 自带的 ISAPI extension。恳求处理器的作用在于，解析收到的 SOAP 恳求，调用 Web service，然后再生成相应的 SOAP 应答。Web 效劳器得到 SOAP 应答后，会再通过 应答的方式把它送回到客户端。

2. 根本概念

SOAP

Web service 建好以后，其他人就会去调用它。简洁对象访问协议(SOAP)供给了标准的远程过程调用(RPC)方法来调用 Web service。SOAP 标准定义了 SOAP 消息的格式，以及怎样通过 协议来使用 SOAP。SOAP 也是基于 XML 和 XSD 的，XML 是 SOAP的数据编码方式。客户端和效劳端之间的方法调用恳求和结果返回值都放在这些消息里。

XML 和 XSD

可扩展的标记语言(XML)是 Web service 平台中表示数据的根本格式。除了易于建立和易于分析外，XML 主要的优点在于它既是平台无关的，又是厂商无关的。无关性是比技术优越性更重要的。XML 解决了数据表示的问题，但它没有定义一套标准的数据类型，更没有说怎么去扩展这套数据类型。例如，整形数到底代表什么？16 位，32 位，还是64 位？这些细节对实现互操作性都是很重要的。W3C 制定的 XML Schema(XSD)就是特地解决这个问题的一套标准。它定义了一套标准的数据类型，并给出了一种语言来扩展这套数据类型。

Web service 平台就是用 XSD 来作为其数据类型系统的。当用某种语言(如 VB.NET 或 C#)来构造一个 Web service 时，为了符合Web service 标准，全部使用的数据类型都必需被转换为 XSD 类型。

WSDL(Web Services Description Language)

用于描述效劳端所供给效劳的 XML 格式。WSDL 文件里，描述了效劳端供给的效劳，供给的调用方法，以及调用时所要遵循的格式，比方调用参数和返回值的格式等等。WSDL 很

像 COM 编程里的 IDL(Interface Description Language)，是效劳器与客户端之间的契约，双方必需按契约严格行事才能实现功能。

WSML(Web Services Meta Language)

用于描述 WSDL 里供给的方法与实现该方法的COM 对象之间的映射关系。该文件是Microsoft 的实现中特有的，不是 SOAP 标准的一局部。一般状况下，该文件只在效劳端存在。

1. Webservice 的技术特点

长项一： 跨防火墙的通信

假设应用程序有成千上万的用户，而且分布在世界各地，那么客户端和效劳器之间的通信将是一个麻烦的问
题。由于客户端和效劳器之间通常会有防火墙或者代理效劳器。在这种状况下，使用DCOM 就不是那么简洁，通常也不便于把客户端程序公布到数量如此浩大的每一个用户手中。

传统的做法是，选择用扫瞄器作为客户端，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。

举个例子，在应用程序里参加一个页面，必需先建立好用户界面(Web 页面)，并在这个页面后面，包含相应商业规律的中间层组件，还要再建立至少一个ASP 页面，用来承受用户输入的信息，调用中间层组件，把结果格式化为HTML 形式，最终还要把“结果页”送回扫瞄器。要是客户端代码不再如此依靠于HTML 表单，客户端的编程就简洁多了。

假设中间层组件换成 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，还削减了代码简单度，并能够增加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”。

从阅历来看，在一个用户界面和中间层有较多交互的应用程序中，使用 Web Service 这种构造，可以节约花在用户界面编程上20%的开发时间。另外，这样一个由Web Service 组成的中间层，完全可以在应用程序集成或其它场合下重用。最终，通过 Web Service 把应用程序的规律和数据“暴露”出来，还可以让其它平台上的客户重用这些应用程序。

长项二： 应用程序集成

企业级的应用程序开发者都知道，企业里常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气。应用程序常常需要从运行在IB M 主机上的程序中猎取数据；或者把数据发送到主机或UNIX 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。

例如，有一个订单登录程序，用于登录从客户来的订单，包括客户信息、发货地址、数量、价格和付款方式等内容；还有一个订单执行程序，用于实际货物发送的治理。这两个程序来 自不同软件厂商。一份订单进来之后，订单登录程序需要通知订单执行程序发送货物。通 过在订单执行程序上面增加一层 Web Service，订单执行程序可以把“Add Order”函数
“暴露”出来。这样，每当有订单到来时，订单登录程序就可以调用这个函数来发送货物了。

长项三： B2B 的集成

用 Web Service 集成应用程序，可以使公司内部的商务处理更加自动化。但当交易跨越供给商和客户、突破公司的界限时会怎么样呢？跨公司的商务交易集成通常叫做B2B 集成。

Web Service 是 B2B 集成成功的关键。通过 Web Service，公司可以把关键的商务应用“暴露”给指定的供给商和客户。例如，把电子下单系统和电子发票系统“暴露”出来，客户就可以以电子的方式发送订单，供给商则可以以电子的方式发送原料选购发票。固然，这并不是一个的概念， EDI(电子文档交换)早就是这样了。但是，Web Service 的实现要比EDI 简洁得多，而且 Web Service 运行在 Internet 上，在世界任何地方都可轻易实现，其运行本钱就相对较低。不过，Web Service 并不像 EDI 那样，是文档交换或 B2B 集成的完整解决方案。Web Service 只是 B2B 集成的一个关键局部，还需要很多其它的局部才能实现集成。

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。只要把商务规律“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规律，而不管他们的系统在什么平台上运行，使用什么开发语言。这样就大大削减了花在B2B 集成上的时间和本钱，让很多原本无法承受EDI 的中小企业也能实现 B2B 集成。

长项四： 软件和数据重用

软件重用是一个很大的主题，重用的形式很多，重用的程度有大有小。最根本的形式是源代码模块或者类一级的重用，另一种形式是二进制形式的组件重用。

当前，像表格控件或用户界面控件这样的可重用软件组件，在市场上都占有很大的份额。但这类软件的重用有一个很大的限制，就是重用仅限于代码，数据不能重用。缘由在于， 公布组件甚至源代码都比较简洁，但要公布数据就没那么简洁，除非是不会常常变化的静态 数据。

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，要先从第三方购置、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。举个例子，要在应用程序中确认用户输入的地址，只需把这个地址直接发送给相应的Web Service，这个 Web Service 就会帮你查阅街道地址、城市、省区和邮政编码等信息，确认这个地址是否在相应的邮政编码区域。Web Service 的供给商可以按时间或使用次数来对这项效劳进展收费。这样的效劳要通过组件重用来实现是不行能的，那样的话你必需下载并安装好包含街道地址、城市、省区和邮政编码等信息的数据库，而且这个数据库还是不能实时更的。

另一种软件重用的状况是，把好几个应用程序的功能集成起来。例如，要建立一个局域网上的门户站点应用，让用户既可以查询联邦快递包裹，查看股市行情，又可以治理自己的日程安排，还可以在线购置电影票。现在Web 上有很多应用程序供给商，都在其应用中实现了这些功能。一旦他们把这些功能都通过Web Service “暴露”出来，就可以格外简洁地把全部这些功能都集成到你的门户站点中，为用户供给一个统一的、友好的界面。

将来，很多应用程序都会利用Web Service，把当前基于组件的应用程序构造扩展为组件/Web Service 的混合构造，可以在应用程序中使用第三方的Web Service 供给的功能，也可以把自己的应用程序功能通过Web Service 供给应别人。两种状况下，都可以重用代码和代码背后的数据。

3.如何调用 webservice

4.0 webservice 的调用过程

客户端：取得效劳端的效劳描述文件WSDL，解析该文件的内容，了解效劳端的效劳信息，以及调用方式。依据需要，生成恰当的SOAP 恳求消息〔指定调用的方法，已经调用

的参数〕，发往效劳端。等待效劳端返回的SOAP　回应消息，解析得到返回值。
效劳端：生成效劳描述文件，以供客户端猎取。接收客户端发来的SOAP 恳求消息，解析其中的方法调用和参数格式。依据 WSDL 和 WSML 的描述，调用相应的 COM 对象来完成指定功能，并把返回值放入 SOAP 回应消息返回给用户。
高层接口

使用高层接口，不需要知道 SOAP 和 XML 的任何信息，就可以生成和使用一个WebService。Soap Toolkit 2.0 通过供给两个 COM 对象——SoapClient 和 SoapServer，来完成这些功能。

在客户端，只需要生成一个 SoapClient 实例，并用 WSDL 作为参数来调用其中的 mssoapinit 方法。SoapClient 对象会自动解析 WSDL 文件，并在内部生成全部 Web Service 的方法和参数信息。之后，你就可以像调用 IDispatch 接口里的方法一样，调用里面全部的方法。在 VB 或是脚本语言里，你甚至可以直接在SoapClient 对象名后面直接加上.方法(参数…)进展调用。
低层接口

要使用低层接口，你必需对SOAP 和 XML 有所了解。你可以对SOAP 的处理过程进展掌握，特别是要做特别处理的时候。
在客户端，首先要创立一个　Connector 对象，负责　连接。设定 Connector 的一些头部信息，比方 EndPoinURL 和 SoapAction 等。假设网络连接需要使用代理效劳器，那也要在这里设定相关的信息。接着创立SoapSerializer 对象，用于生成Soap 消息。依据WSDL 里定义，把全部参数按挨次序列化，得到一个完整的SOAP 恳求消息。该 Soap 消息，作为 Payload 通过　Connector 被发送到效劳端。最终，生成一个SoapRead er 对象，负责读取效劳端返回的SOAP 消息，取得其中的返回值。

4.0 使用 PowerBuilder 调用

适用版本 8.0 需下载 Bulletin Web Services Toolkit 4.1

4.1　使用 java 调用

需要下载 apache　soap。下载地址：
导入：

```
import org.apache.soap.*;
import org.apache.soap.rpc.*;
```

例程：

```
import  java.io.*;
import java.util.*;
import java.net.*;
import  org.w3c.dom.*;
import org.apache.soap.util.xml.*;
import  org.apache.soap.*;
import  org.apache.soap.encoding.*;
import org.apache.soap.encoding.soapenc.*;
import  org.apache.soap.rpc.*;
import  org.apache.soap.transport.    .SOAP    Connection;
```

```
public class testClient {

    public static void main(String[] args) throws Exception {

        URL url = new URL
//改成你的地址
    SOAPMappingRegistry smr = new SOAPMappingRegistry ;
    StringDeserializer sd = new StringDeserializer ;
    smr.mapTypes (Constants.NS_URI_SOAP_ENC, new QName (“”, “Resu
lt“), null, null, sd);

        // 创立传输路径和参数
    SOAP  Connection st = new SOAP  Connection;
        // 创立调用
    Call call = new Call ;
    call.setSOAPTransport(st);
    call.setSOAPMappingRegistry (smr);

        call.setTargetObjectURI call.setMethodName(“addNumbers“);
    call.setEncodingStyleURI

“);

        Vector params = new Vector;
    params.addElement(new Parameter(“NumberOne“, Double.class, “10“,
 null));
    params.addElement(new Parameter(“NumberTwo“, Double.class, “25“,
 null));
    call.setParams(params);

        Response resp = null;
        try {
     resp = call.invoke (url,
s“);
    }
    catch (SOAPException e) {
    System.err.println(“Caught SOAPException (“ + e.getFaultCode  +“):
“ + e.getMessage );
    return;
    }
```

```
    // 检查返回值
if (resp != null && !resp.generatedFault) {
Parameter ret =  resp.getReturnValue; Object
value = ret.getValue;

    System.out.println ("Answer--> " + value);
}
else {
   Fault fault = resp.getFault ;
   System.err.println ("Generated fault: ");
   System.out.println (" Fault Code = " + fault.getFaultCode);
   System.out.println (" Fault String = " + fault.getFaultString);
}
   }
}
```

4. 3 在 asp 中使用 Jscript 调用

需下载 msSoapToolkit20.exe

引用：MSSOAP.SoapClient

例程：

```
   <%@ LANGUAGE = JScript %>
<HTML>
<HEAD>
<TITLE>webservice 演示</TITLE>
</HEAD>
<BODY>
   <%
       var WSDL_URL = var a, b, res
        var soapclient
         a = 12
         b = 13
       soapclient = Server.CreateObject("MSSOAP.SoapClient")
       soapclient.ClientProperty("Server    Request") = true
在 ASP 中运行 需要设置 Server    Request 选项
",Service1Soap,"")
       res = soapclient.test(2,3)
%>
<h3>webservice 演示</h3>
<B>Result:</B>  <%=res%><P><P>
</BODY>
```

```
</HTML>
```

4. 4 在 asp 中使用 vbscript 调用

需下载 msSoapToolkit20.exe

引用：MSSOAP.SoapClient

例程：

```
<%@ LANGUAGE = VBScript %>
<HTML>
<HEAD>
<TITLE>webservie 演示</TITLE>
</HEAD>
<BODY>
<%
Dim soapclient
Const WSDL_URL = set soapclient =
    Server.CreateObject("MSSOAP.SoapClient")
    soapclient.ClientProperty("Server    Request") = True
    soapclient.mssoapinit
   ://192.168.0.4:8080/yundan/Service1.WSDL","Service1","Service1Soap", ""
Dim res
       res = soapclient.test(23, 34)
%>
<h1>webservie 演示</h1>
<B>Result:</B> <%=res%><P><P>
</BODY>
</HTML>
```

4. 5 使用 C#调用

无需下载任何组件

Visual◊工程◊建 windows 应用程序。◊C#工程

在解决方案资源治理器中添加 web 引用,输入 wsdl 文件所在地址。

将 web 引用改名.

yundan.〔service_name〕即可引用

需引用 System.web.services

例程：

```
using System;
using System.Drawing;
using System.Collections;
using System    ponentModel;using
System.Windows.Forms; using
System.Data;
namespace csharp
```

```
{
        public class Form1 : System.Windows.Forms.Form
        {
            private System.Windows.Forms.Label label1;
            private System.Windows.Forms.TextBox textBox1;
            private System    ponentModel.Container components = null;public
            Form1
            {
                InitializeComponent;
            }
            protected override void Dispose( bool disposing )
            {
                if( disposing )
                {
                    if (components != null)
                    {
                        components.Dispose;
                    }
                }
                base.Dispose( disposing );
            }
            #region Windows Form Designer generated code
            private void InitializeComponent
            {
                this.label1 = new System.Windows.Forms.Label;
                this.textBox1 = new System.Windows.Forms.TextBox;
                this.SuspendLayout;
                this.label1.AutoSize = true;
                this.label1.Location = new System.Drawing.Point(88, 48);
                this.label1.Name = "label1";
                this.label1.Size = new System.Drawing.Size(91, 14);
                this.label1.TabIndex = 0;
                this.label1.Text = "Webservice 演示";
                this.textBox1.Location = new System.Drawing.Point(88, 128);
                this.textBox1.Name = "textBox1";
                this.textBox1.TabIndex = 1;
                this.textBox1.Text = "textBox1";
                this.AutoScaleBaseSize = new System.Drawing.Size(6, 14);
                this.ClientSize = new System.Drawing.Size(292, 273);
                this.Controls.AddRange(new System.Windows.Forms.Control[]
{
                this.textBox1,
```

```
            this.label1
});
            this.Name = "Form1";
            this.Text = "C#Webservie 演示";
            this.Load += new System.EventHandler(this.Form1_Load);
            this.ResumeLayout(false);
        }
        #endregion
        [STAThread]
        static void Main
        {
            Application.Run(new Form1);
        }
            private void Form1_Load(object sender, System.EventArgs e)
        {
            int str;
            你的web引用的名字.Service1 cc=new 你的web引用的名字.Service1;
            str=cc.test(123,324);
            textBox1.Text=str.ToString;
        }
    }
}
```

4.6(javascript)

需下载 msSoapToolkit20.exe

引用：MSSOAP.SoapClient

例程：

```
var WSDL_URL = ""WScript.echo("Connecting: " + WSDL_URL)
var Calc = WScript.CreateObject("MSSOAP.SoapClient")
Calc.mssoapinit(WSDL_URL, "", "", "")
var Answer
Answer = Calc.test(14,28)
WScript.Echo("14+28=" + Answer)
```

将其存成 domo.js 文件，直接双击运行。

4.7 使用 vb.net 调用

无需下载任何组件

Visual◊工程◊建 windows 应用程序。◊Basic 工程

在解决方案资源治理器中添加 web 引用,输入 wsdl 文件所在地址。
将 web 引用改名为 yundan. yundan.〔service_name
〕即可引用
需引用 System.web.services
例程:

```
    Public Class Form1
        Inherits System.Windows.Forms.Form
#Region " Windows 窗体设计器生成的代码 "
    Public Sub New
        MyBase.New
        InitializeComponent
    End Sub
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub
    Private components As System  ponentModel.IContainer Friend
    WithEvents Label1 As System.Windows.Forms.Label Friend
    WithEvents TextBox1 As System.Windows.Forms.TextBox
    <System.Diagnostics.DebuggerStepThrough> Private Sub InitializeCom
ponent
        Me.Label1 = New System.Windows.Forms.Label
        Me.TextBox1 = New System.Windows.Forms.TextBox
        Me.SuspendLayout
        Me.Label1.AutoSize = True
        Me.Label1.Location = New System.Drawing.Point(96, 40)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(91, 14)
        Me.Label1.TabIndex = 0
        Me.Label1.Text = "Webservice 演示"
        Me.TextBox1.Location = New System.Drawing.Point(88, 144)
        Me.TextBox1.Name = "TextBox1"
        Me.TextBox1.TabIndex = 1
        Me.TextBox1.Text = "TextBox1"
        Me.AutoScaleBaseSize = New System.Drawing.Size(6, 14)
        Me.ClientSize = New System.Drawing.Size(292, 273)
        Me.Controls.AddRange(New System.Windows.Forms.Control {Me.Tex
tBox1, Me.Label1})
```

```
        Me.Name = "Form1"
        Me.Text = "VB.net webserive 演示"
        Me.ResumeLayout(False)
    End Sub
#End Region
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As Sy
stem.EventArgs) Handles MyBase.Load
        Dim cc As yundan.Service1 = New yundan.Service1
        TextBox1.Text = cc.test(12, 123)
    End Sub
End Class
```

4.7 使用 vb6.0 调用

需下载 msSoapToolkit20.exe

添加引用：Microsoft Soap Type Library

位置:"C:\Program Files\Common Files\MSSoap\Binaries\ MSSOAP1.dll"

调用方法：

```
Dim cc As New MSSOAPLib.SoapClient
```

例程：

添加一个 button 控件 Command1◊添加 3 个 textbox 控件，text1,text2,text3◊标准 EXE◊建工程

代码如下：

```
Option Explicit
Dim cc As New MSSOAPLib.SoapClient
Private Sub Command1_Click
cc.mssoapinit ""Me.Text3.Text = cc.test(CInt(Text1.Text), CInt(Text2.Text))
End Sub
```

4.8 使用 vbscript 调用

需下载 msSoapToolkit20.exe

引用：MSSOAP.SoapClient

例程：

```
Option Explicit
Const WSDL_URL = ""WScript.echo "Connecting: " & WSDL_URL
Dim Calc
Set Calc = CreateObject("MSSOAP.SoapClient")
Calc.mssoapinit WSDL_URL
Dim Answer
Answer = Calc.test(14,28)
WScript.Echo "14+28=" & Answer
```

将其存成 domo.vbs 文件，直接双击运行。

4.7 使用 vc 调用

需下载 msSoapToolkit20.exe

引用

```
#import  "msxml3.dll"
using  namespace  MSXML2;
#import "C:\Program Files\Common Files\MSSoap\Binaries\mssoap1.dll" exc
lude("IStream", "ISequentialStream", "_LARGE_INTEGER", "_ULARGE_INTE
GER", "tagSTATSTG", "_FILETIME") raw_interfaces_only
using  namespace  MSSOAPLib;
```

例程：

建工程àMFC AppWizard(exe)[Mclient]àstep1à 根本对话à 其他默认值即可修

改源文件：

```
<  StdAfx.h>
// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently, but
//      are changed infrequently
//

     #if  !defined(AFX_STDAFX_H__045CD307_9518_4AF1_8CE3_8FFE38D1A
CB2__INCLUDED_)
#define AFX_STDAFX_H__045CD307_9518_4AF1_8CE3_8FFE38D1ACB2___IN
CLUDED_

     #if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

     #define VC_EXTRALEAN       // Exclude rarely-used stuff from Window
s headers

     #include <afxwin.h>        // MFC core and standard components
 #include  <afxext.h>          // MFC extensions
 #include  <afxdisp.h>         // MFC Automation classes
 #include  <afxdtctl.h>        // MFC support for Internet Explorer 4 Common
Controls
 #ifndef  _AFX_NO_AFXCMN_SUPPORT
 #include  <afxcmn.h>          // MFC support for Windows Common Controls
 #endif // _AFX_NO_AFXCMN_SUPPORT


      #import "msxml3.dll"
using  namespace  MSXML2;
```