第五章 Web Service

第一章 Web service 简介

现在 Internet 正在不断地发展着，在互联网应用刚开始的时候，我们浏览的网页只是静态的，不可交互的。而现在随着技术的日益发展，将提供给网页浏览者一个可编程的 Web 站点。这些站点将在组织、应用、服务、驱动上更加紧密的结合在一起，这些站点将通过一些应用软件直接连接到另一个 Web 站点，这些可编程的 Web 站点相比传统的 web 站点来说，将变得更加能重复使用，也更加智能化！

.net 平台给我们提供了一种运行环境，即公用语言运行环境（CLR，Common Language Runtime）。对 CLR 来说，它提供了一种内置机制来创建一个可编程的站点，、对于 Web 程序开发者和 VB 程序员来说，这将是一致、熟悉的。这种模型是可以重复使用，也可以再扩展。 它包含了开放的 Internet 标准（HTTP, XML, SOAP, SDL）。以便它能被网页浏览者访问。

ASP.NET 使用.asmx 文件来对 Web Services 的支持。.asmx 文件和.aspx 文件一样都属于文本文件。它包含在.aspx 文件之中，成为 ASP.NET 应用程序的一部分。

下面我们将举一个简单的例子来介绍.asmx 文件，我们还是从“ Hello，World ”这个经典的例子说起，代码如下：

```
<!-- 文件名：webservice\sisam.asmx -->
<%@ WebService Language="VB" Class="HelloWorld" %>
    Imports System.Web.Services
    Public Class HelloWorld :Inherits WebService
    Public Function <WebMethod( )> SayHelloWorld( ) As String
            Return("Hello World")
   End Function
    End Class
```

说明：

1．编码最开始必须进行 WebService 声明，从而定义这个文件为一个 Web Service。而且，在同一行中设置好编程语言的类型。

2．然后，引入名字空间 System.Web.Services。注意，这个名字空间属于最基本的元素，必须要包含它。

3．接着，声明 service 中的功能模块，也就是类模块，这里的类名叫 HelloWorld。这个类来源于基类 WebService，而且应该是 public 类型。

4．最后，定义 service 的可访问方法。在表示方法的符号前面，要设置好自定义属性。对应于 C#语言，属性值就是[WebMethod]；对应于 VB，就是。如果没有设置这个属性，那么这个方法就不能从 service 中访问。一个局部应用可以使用任何的 public 类型的类，但是只有具备[WebMethod]的类才可以通过 SOAP 被远程地访问。

当对 service 的请求发生时，.asmx 文件将自动地被 ASP.NET 运行环境所编译。随后的请求就可以由缓冲的预编译类型对象执行。

为了测试编写好的代码，我们需用一个支持 ASP.NET 的 Web 服务器。假设这个 Web 服务器的名称叫做 server1，其上有一个虚拟目录 test。请跟随下面步骤开始测试：

1．将代码保存为 HelloWorld.asmx

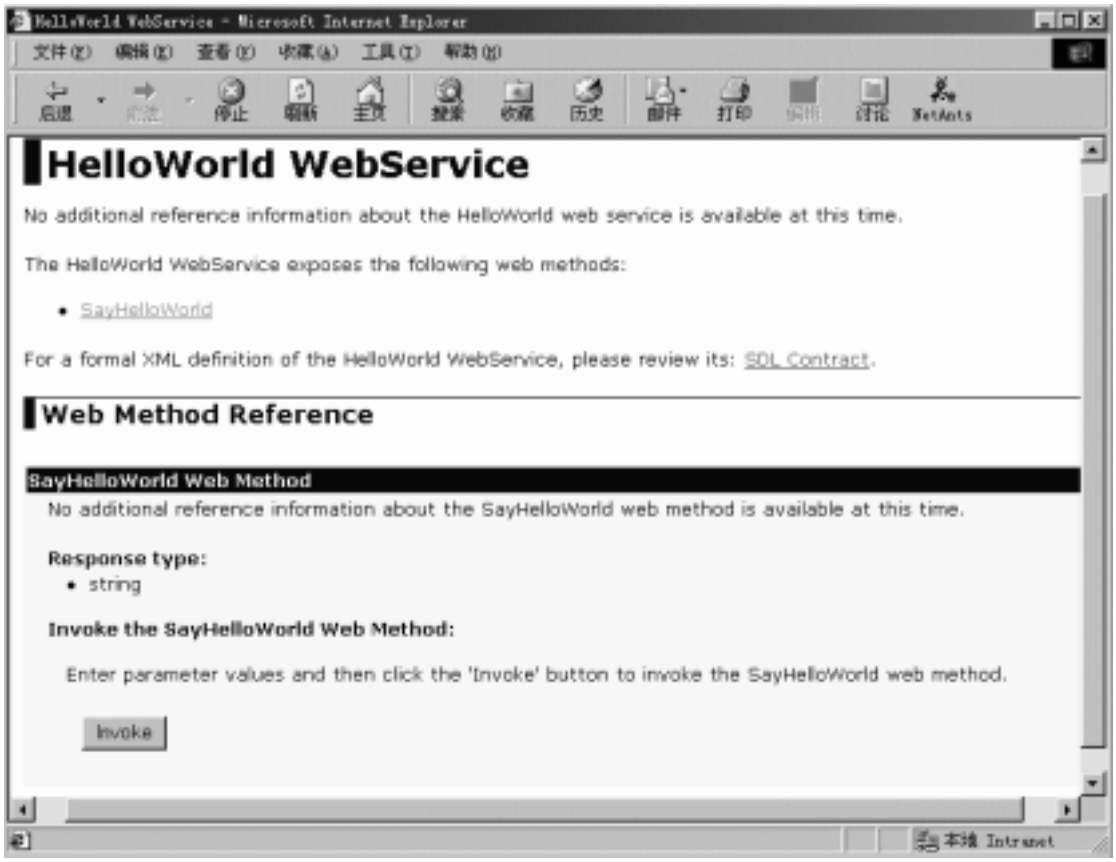
2．放到 Web 服务器 Foo 的虚拟目录 Bar 下

3．打开 Internet Explorer5，在地址栏输入 http://server1/test/HelloWorld.asmx

这时，我们将看到关于这个 Web Service 的公用方法－也就是那些标记为 WebMethod 属性的字符，并得知调用这些方法可以使用的协议，比如 SOAP 或者 HTTP GET。

在 Internet Explorer 的地址栏中输入 http://Foo/Bar/HelloWorld.asmx?SDL 后，将产生基于服务描述语言（Service Description Language ：SDL）语法的具备相同信息的 XML 文件。这个 SDL 文件非常重要，客户端就是使用它来访问 service。

我们来看一下程序运行的效果：



从客户端进行访问：

除了允许开发者使用的创建 Web Services 的技术以外，Microsoft 的.NET 框架给客户端

提供了一套访问并使用 Web Services 的精致且高深的工具和代码。由于 Web Services 是基于如简单对象访问协议 SOAP（Simple Object Access Protocol）和 HTTP 这样的开放协议标准的，从而，我们就可以使用这种客户端技术使用非 ASP.NET 的 Web Services。当然，这也需用高水平地合成 ASP.NET Web Services 和这种客户端技术。

SDK 中有一个工具叫做 WebServiceUtil.exe，我们可以使用它来下载一个 Web Services 的 SDL 描述语言，并创建表达这个 Service 的代理类。比如，当我们输入以下命令，就可以创建一个叫做 HelloWorld.cs 的代理类：

WebServiceUtil /c:proxy /pa:http://someDomain.com/someFolder/HelloWorld.asmx?SDL

这个类看起来与前面创建的类非常相似。它包含一个方法 SayHelloWorld，该方法返回一个字符串。将这个代理类编译到一个应用程序中，然后调用这个代理类的方法，结果就是：通过 HTTP，这个代理类包装 SOAP 请求，然后接收 SOAP 编码响应，最后汇集成为一个字符串。

从客户端来看，代码是很简单的，返回的结果也很简单，就是一个字符串"Hello World"。同样为了对照方便，我们列出了使用 VB、C#以及 JScript 三种语言编写的代码：

C#

```
HelloWorld myHelloWorld = new HelloWorld();
String sReturn = myHelloWorld.SayHelloWorld();
```

VB

```
Dim myHelloWorld As New HelloWorld()
Dim sReturn As String = myHelloWorld.SayHelloWorld()
```

JScript

```
var myHelloWorld:HelloWorld = new HelloWorld();
var sReturn:String = myHelloWorld.SayHelloWorld();
```

通过上面的例程，你可能对 Web Services 有了初步的印象。下面，我们将介绍 Web Services 中涉及到的各种数据类型，也就是 Web Services 方法的输入/输出参数类型。因为 Web Services 的执行是建立在 XML 架构之上的，所以它能够支持丰富的数据类型。下表列出了使用 SOAP 协议时 Web Services 支持的数据类型：

类型	描述
基础类型	也即标准基础类型，包括： String、Int32、Byte、Boolean、Int16、Int64、Single、Double、Decimal 、 DateTime(类 似 XML 中 的 timeInstant)、DateTime(类似 XML 中的 date)、DateTime(类 似 XML 中 的 time) 以 及 XmlQualifiedName(类似 XML 中的 QName)。
枚举类型	枚举类型。例如：. "public enum color { red=1, blue=2 }"

基础，枚举数组	上面提到的类型数组。例如：string[] 和 int[]
类和结构	带有公用域或属性的类和结构，公用域和属性是串行结构的
类和结构体数组	上述类型的数组
DataSet	ADO.NET DataSet 类型。DataSets 能在类和结构体作为字段来使用。
DataSet 数组	上述类型的数组
XmlNode	XmlNode 是 XML 文档片断的内存表示，就好像一个轻量级的 XML 文档对象模型。比如说，"" 就可以存储在一个 XmlNode 类型变量中。我们可以将 XmlNodes 作为参数传递，以 SOAP 兼容方式附加到传递给 Web Services 的 XML 文档上。返回值也是同样原理。XmlNode 也可看成是类或结构中的字段。
XmlNode 数组	上述类型的数组

当通过 SOAP 或者 HTTP GET/POST 调用 Web Services 时，返回值可以是上述提到的任何一种数据类型。

参数的数据类型

使用 SOAP 协议时，"通过值"以及"通过引用"这两种输入/输出参数形式都可被支持。如果是"通过引用"的参数类型，就会产生两种方式的数据发送效果：到服务器的以及返回到客户端的。但是，当通过 HTTP GET/POST 传递输入参数给 Web Services 时，就只支持有限的数据类型了，而且还必须是"通过值"形式的参数。这些类型如下：

类型	描述
基础类型 (有限的)	支持大多数标准基础类型，包括： Int32、String、Int16、Int64、Boolean、Single、Double、Decimal、DateTime、TimeSpan、UInt16、UInt32、UInt64 和 Currency。从客户端来看，所有这些类型都转变为 string。
枚举类型	比如： "public enum color { red=1, blue=2 }"。
基础类型数组，枚举类型数组	上述类型的数组，比如 string[]和 int[]

现在我们将举一个例子，来说明上面我们介绍的数据类型：

这个例子利用 WebServiceUtil.exe 建立的 SOAP 代理来使用上面列出的数据类型。注意：因为在.asmx 文件中定义了多于一个的公用类，所以，我们必须要指定哪一个作为 WebService 类，这可以通过设置 WebService 标识的 Class 属性来实现，代码如下：

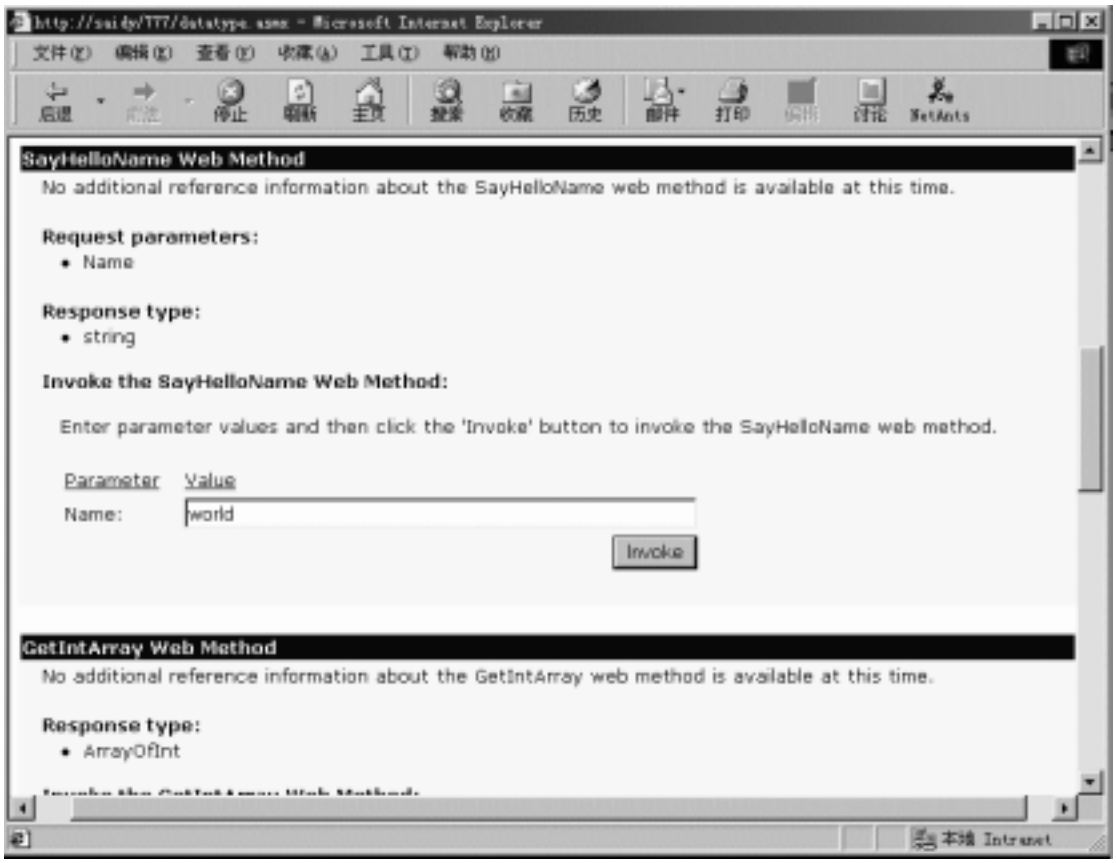
```
<%@ WebService Language="C#" Class="DataTypes" %>
```

源文件 webservice\datatype.asmx 的内容如下：

```
<%@ WebService Language="VB" Class="DataTypes" %>

Imports System
Imports System.Web.Services

Public Enum Mode

    EOn = 1
    EOff = 2

End Enum

Public Class Order

    Public OrderID As Integer
    Public Price As Double
End Class


Public Class DataTypes

‘ SayHello 方法显示从 service 中返回的一个字符串信息。
    Public Function <WebMethod()> SayHello() As String

        Return "Hello World!"
    End Function


‘ SayHelloName 方法返回一个字符串，并接受一个字符串参数。
    Public Function <WebMethod()> SayHelloName(Name As String) As String

        Return "Hello" & Name
    End Function


‘ GetIntArray 方法显示了如何返回一个整数数组。
    Public Function <WebMethod()> GetIntArray() As Integer()

        Dim I As Integer
        Dim A(5) As Integer
```

```
        For I = 0 to 4
            A(I) = I*10
        Next
        Return A
    End Function


‘ GetMode 方法返回一个枚举数值。
    Public Function <WebMethod()> GetMode() As Mode

        Return Mode.EOff
    End Function


‘ GetOrder 方法返回一个类。
    Public Function <WebMethod()> GetOrder() As Order

        Dim MyOrder As New Order

        MyOrder.Price=34.5
        MyOrder.OrderID = 323232

        Return MyOrder
    End Function


‘ GetOrders 方法返回定单对象数组。
    Public Function <WebMethod()> GetOrders() As Order()

        Dim MyOrder(2) As Order

        MyOrder(0) = New Order()
        MyOrder(0).Price=34.5
        MyOrder(0).OrderID = 323232
        MyOrder(1) = New Order()
        MyOrder(1).Price=99.4
        MyOrder(1).OrderID = 645645

        Return MyOrder
    End Function

End Class
```

程序运行的效果如下：



当我们单击 invoke 的时候，将显示：

```
<?xml version="1.0" ?>
<string xmlns="http://tempuri.org/">Helloworld</string>
```

对于使用客户端应用程序而言，使用 WebServiceUtil 代理生成工具配置这些数据类型是透明的。请看关于 Web Service 的一个客户端例程：

客户端访问的文件：clint.aspx，内容如下：

```
<%@ Import Namespace="DataTypesService" %>

<html>
<style>
  div
  {
    font: 8pt verdana;
    background-color:cccccc;
    border-color:black;
    border-width:1;
```

```
        border-style:solid;
        padding:10,10,10,10;
    }

</style>

<script language="VB" runat="server">

        Public Sub Page_Load(Sender As Object, E As EventArgs)

            Dim D As DataTypes = New DataTypes()
            Message1.InnerHtml = D.SayHello()
            Message1.InnerHtml = Message1.InnerHtml & D.SayHelloName("Bob")
            Message3.InnerHtml = Message3.InnerHtml & D.GetMode()

            Dim MyIntArray As Integer() = D.GetIntArray()
            Dim MyString As String = "Contents of the Array:<BR>"

            For I = 0 To MyIntArray.Length - 1
                MyString = MyString & MyIntArray(I) & "<BR>"
            Next

            Message2.InnerHtml = Message2.InnerHtml & MyString

            Dim MyOrder As Order = D.GetOrder()
            Message4.InnerHtml = Message4.InnerHtml & "<BR>OrderID: " & MyOrder.OrderID
            Message4.InnerHtml = Message4.InnerHtml & "<BR>Price: " & MyOrder.Price

            Dim MyOrders As Order() = D.GetOrders()
            Message5.InnerHtml = Message5.InnerHtml & "<BR>OrderID: " &
MyOrders(0).OrderID
            Message5.InnerHtml = Message5.InnerHtml & "<BR>Price: " & MyOrders(0).Price

        End Sub

</script>

<body style="font: 10pt verdana">
<H4>Using DataTypes with Web Services</H4>

    <h5>Methods that return a Primitive (String): </h5>
    <div id="Message1" runat="server"/>

    <h5>Methods that return an Array of Primitives (Integers): </h5>
```

```
    <div id="Message2" runat="server"/>

    <h5>Method that returns an Enum: </h5>
    <div id="Message3" runat="server"/>

    <h5>Method that returns a Class/Struct: </h5>
    <div id="Message4" runat="server"/>

    <h5>Method that returns an array of Classes/Structs: </h5>
    <div id="Message5" runat="server"/>

</body>
</html>
```

在客户端程序中，我们使用<%@ Import Namespace="DataTypesService" %>来引入 DataTypesService 这个我们自定义的名字空间。然后在程序中只是调用了 DataTypesService 中的方法。

现在我们来看如何生成名字空间：

1．Datatype.vb 中的内容：

```
Imports System.Xml.Serialization
Imports System.Web.Services.Protocols
Imports System.Web.Services

Namespace DataTypesService
  Public Class    DataType
     Inherits System.Web.Services.Protocols.SoapClientProtocol
Public Sub New()
MyBase.New
Me.Url="http://localhost/QuickStart/aspplus/samples/services/DataTypes/VB/DataTypes.asmx"
End Sub

Public Function
<System.Web.Services.Protocols.SoapMethodAttribute("http://tempuri.org/SayHello")>
SayHello() As String
                Dim results() As Object = Me.Invoke("SayHello", New Object(0) {})
                Return CType(results(0),String)
            End Function
            Public Function BeginSayHello(ByVal callback As System.AsyncCallback, ByVal
asyncState As Object) As System.IAsyncResult
                Return Me.BeginInvoke("SayHello", New Object(0) {}, callback, asyncState)
            End Function
            Public Function EndSayHello(ByVal asyncResult As System.IAsyncResult) As String
```

```
        Dim results() As Object = Me.EndInvoke(asyncResult)
        Return CType(results(0),String)
    End Function
    Public Function <System.Web.Services.Protocols.SoapMethodAttribute("http://tempuri.org/SayHelloName")>
SayHelloName(ByVal <System.Xml.Serialization.XmlElementAttribute("Name", IsNullable:=true)> name As String) As String
        Dim results() As Object = Me.Invoke("SayHelloName", New Object() {name})
        Return CType(results(0),String)
    End Function
    Public Function BeginSayHelloName(ByVal name As String, ByVal callback As System.AsyncCallback, ByVal asyncState As Object) As System.IAsyncResult
        Return Me.BeginInvoke("SayHelloName", New Object() {name}, callback, asyncState)
    End Function
    Public Function EndSayHelloName(ByVal asyncResult As System.IAsyncResult) As String
        Dim results() As Object = Me.EndInvoke(asyncResult)
        Return CType(results(0),String)
    End Function
    Public Function <System.Web.Services.Protocols.SoapMethodAttribute("http://tempuri.org/GetIntArray"), _
    System.Xml.Serialization.XmlArrayAttribute(IsNullable:=true, ArrayType:=System.Xml.Serialization.XmlArrayType.Soap), _
    System.Xml.Serialization.XmlArrayItemAttribute("int", IsNullable:=false)>
GetIntArray() As Integer()
        Dim results() As Object = Me.Invoke("GetIntArray", New Object(0) {})
        Return CType(results(0),Integer())
    End Function
    Public Function BeginGetIntArray(ByVal callback As System.AsyncCallback, ByVal asyncState As Object) As System.IAsyncResult
        Return Me.BeginInvoke("GetIntArray", New Object(0) {}, callback, asyncState)
    End Function
    Public Function EndGetIntArray(ByVal asyncResult As System.IAsyncResult) As Integer()
        Dim results() As Object = Me.EndInvoke(asyncResult)
        Return CType(results(0),Integer())
    End Function
    Public Function <System.Web.Services.Protocols.SoapMethodAttribute("http://tempuri.org/GetMode")>
GetMode() As Mode
        Dim results() As Object = Me.Invoke("GetMode", New Object(0) {})
        Return CType(results(0),Mode)
    End Function
```

```
        Public Function BeginGetMode(ByVal callback As System.AsyncCallback, ByVal
asyncState As Object) As System.IAsyncResult
            Return Me.BeginInvoke("GetMode", New Object(0) {}, callback, asyncState)
        End Function
        Public Function EndGetMode(ByVal asyncResult As System.IAsyncResult) As Mode
            Dim results() As Object = Me.EndInvoke(asyncResult)
            Return CType(results(0),Mode)
        End Function
        Public                                                                  Function
<System.Web.Services.Protocols.SoapMethodAttribute("http://tempuri.org/GetOrder")>
GetOrder() As Order
            Dim results() As Object = Me.Invoke("GetOrder", New Object(0) {})
            Return CType(results(0),Order)
        End Function
        Public Function BeginGetOrder(ByVal callback As System.AsyncCallback, ByVal
asyncState As Object) As System.IAsyncResult
            Return Me.BeginInvoke("GetOrder", New Object(0) {}, callback, asyncState)
        End Function
        Public Function EndGetOrder(ByVal asyncResult As System.IAsyncResult) As Order
            Dim results() As Object = Me.EndInvoke(asyncResult)
            Return CType(results(0),Order)
        End Function
        Public                                                                  Function
<System.Web.Services.Protocols.SoapMethodAttribute("http://tempuri.org/GetOrders"),  _
        System.Xml.Serialization.XmlArrayAttribute(IsNullable:=true,
ArrayType:=System.Xml.Serialization.XmlArrayType.Soap),  _
        System.Xml.Serialization.XmlArrayItemAttribute("Order",          IsNullable:=true)>
GetOrders() As Order()
            Dim results() As Object = Me.Invoke("GetOrders", New Object(0) {})
            Return CType(results(0),Order())
        End Function
        Public Function BeginGetOrders(ByVal callback As System.AsyncCallback, ByVal
asyncState As Object) As System.IAsyncResult
            Return Me.BeginInvoke("GetOrders", New Object(0) {}, callback, asyncState)
        End Function
        Public Function EndGetOrders(ByVal asyncResult As System.IAsyncResult) As Order()
            Dim results() As Object = Me.EndInvoke(asyncResult)
            Return CType(results(0),Order())
        End Function

    End Class
    Public          Enum          <System.Xml.Serialization.XmlRootAttribute("result",
[Namespace]:="http://tempuri.org/", IsNullable:=false)> Mode
```

```
        EOn
        EOff

    End Enum
    Public Class <System.Xml.Serialization.XmlRootAttribute("result",
[Namespace]:="http://tempuri.org/", IsNullable:=true)> Order

        Public OrderID As Integer
        Public Price As Double

    End Class
End Namespace
```

在这个 vb 文件中，我们定义了一个名字空间 DataTypesService。
请看 vb 文件的其中一段代码段：

```
Public Function <System.Web.Services.Protocols.SoapMeth_&
 odAttribute("http://tempuri.org/SayHello")> SayHello() As String
            Dim results() As Object = Me.Invoke("SayHello", New Object(0) {})
            Return CType(results(0),String)
        End Function

Public Function BeginSayHello(ByVal callback As System.AsyncCallback, ByVal asyncState As
Object) As System.IAsyncResult
            Return Me.BeginInvoke("SayHello", New Object(0) {}, callback, asyncState)
End Function

Public Function EndSayHello(ByVal asyncResult As System.IAsyncResult) As String
            Dim results() As Object = Me.EndInvoke(asyncResult)
            Return CType(results(0),String)
End Function
```

上面的代码，是触发 invoke 的单击事件。然后，调用我们在 datatype.asmx 中定义的方法。

要生成上面的名字空间，我们使用 webserviceutil.exe 来编译。

```
webserviceutil -c:proxy /pa:DataTypes.sdl /l:VB /n:DataTypesService
```

5.1.1 小结

web service 提供了在不同体系机构下构建的网站之间相互提供应用接口服务、数据的一种方案。它采用通用的 SOAP、HTTP 以及 XML，就可以把原本互不相干的站点服务形成一整套分布的、自动化和智能化的网络应用，大大减轻了程序员的开发工作量，充分地利用了已经拥有的网络资源和开发资源。在 asp.net 中，web service 文件后缀名采用.asmx，开始应使用<%@ WebService …%>申明，接着引用 System.Web.Services 命名空间，然后定义一个

公用类继承自 WebService 基类，最后实现自己的类，其中向网络开放的功能，应在其方法前面加上 WebMethod 属性。

第二章 一个简单的 Web Service 案例

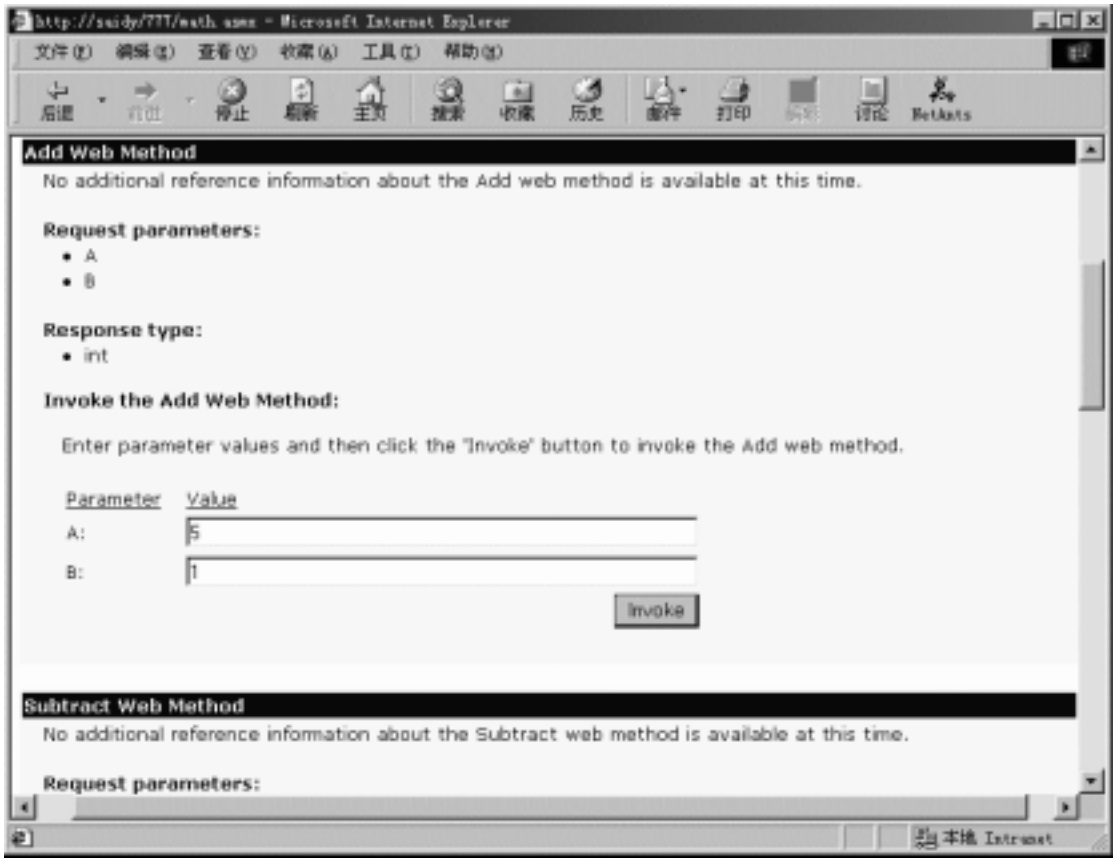
在这个例子中，我们将定义一个 mathservice 类，来对两个数字分别进行加，减，乘，除。当然这个类需要从基类 web service 中继承。请先看该程序的源代码：

源文件：webservice\math.asmx

```
<%@ WebService Language="VB" Class="MathService" %>
Imports System
Imports System.Web.Services
Public Class MathService : Inherits WebService
    Public Function <WebMethod()> Add(A As Integer, B As Integer) As Integer
        Return A + B
    End Function
    Public Function <WebMethod()> Subtract(A As Integer, B As Integer) As Integer

        Return A - B
    End Function
    Public Function <WebMethod()> Multiply(A As Integer, B As Integer) As Integer
        Return A * B
    End Function
    Public Function <WebMethod()> Divide(A As Integer, B As Integer) As Integer
        If B = 0
            Return -1
        End If
        Return CInt(A / B)
    End Function
End Class
```

对于该程序，我们首先要用<%@ WebService Language="vb" Class="MathService" %>来标识。然后定义对应的方法。程序很简单，我们来看一下运行效果，假使我们使用“ 加 ”。如图：

我们要计算 5+1 的值：



通过上面的计算，结果就显示出来了。

一个代理类（proxy class）被创建出来的，我们就可以很容易的创建对象。当然，类中的方法就能够被对象调用。创建代理类，我们可以使用 WebServiceUtil.exe 来创建一个代理类。

我们还是举“加”,“减”,“乘”,“除”的例子。

我们先创建一个文件用于客户端的用户浏览：

```
<%@ Import Namespace="MathServiceSpace" %>
<html>
<script language="VB" runat="server">
   Dim Op1 As Integer = 0
   Dim Op2 As Integer = 0
   Public Sub Submit_Click(Sender As Object, E As EventArgs)
         Try
            Op1 = Int32.Parse(Operand1.Text)
            Op2 = Int32.Parse(Operand2.Text)
         Catch Exp As Exception
               ' Ignored
         End Try

         Dim Service As MathService = New MathService()
         Select (CType(sender,Control).ID)
            Case "Add" :
                  Result.Text = "<b>Result</b> = " & Service.Add(Op1, Op2).ToString()
            Case "Subtract" :
                  Result.Text = "<b>Result</b> = " & Service.Subtract(Op1, Op2).ToString()
            Case "Multiply" :
                  Result.Text = "<b>Result</b> = " & Service.Multiply(Op1, Op2).ToString()
            Case "Divide" :
                  Result.Text = "<b>Result</b> = " & Service.Divide(Op1, Op2).ToString()
         End Select
   End Sub
</script>
<body style="font: 10pt verdana">
   <h4>Using a Simple Math Service </h4>
   <form runat="server">
   <div
style="padding:15,15,15,15;background-color:beige;width:300;border-color:black;border-width:1;
border-style:solid">
      Operand 1: <br><asp:TextBox id="Operand1" Text="15" runat="server"/><br>
      Operand 2: <br><asp:TextBox id="Operand2" Text="5" runat="server"/><p>
      <input type="submit" id="Add" value="Add" OnServerClick="Submit_Click"
runat="server">
      <input type="submit" id="Subtract" value="Subtract" OnServerClick="Submit_Click"
runat="server">
      <input type="submit" id="Multiply" value="Multiply" OnServerClick="Submit_Click"
runat="server">
      <input type="submit" id="Divide" value="Divide" OnServerClick="Submit_Click"
```

```
runat="server">
      <p>
      <asp:Label id="Result" runat="server"/>
   </div>
   </form>
</body>
</html>
```

我们还需要一个 sdl 文件，当然这个文件不用手工输入，我们在浏览一个.asmx 的时候，在后缀名后直接加上？sdl 可以自动生成 sdl 文件。

然后我们在一个.vb 文件里，将定义一个名字空间，.vb 文件的内容：

```
Imports System.Xml.Serialization
Imports System.Web.Services.Protocols
Imports System.Web.Services

Namespace MathServiceSpace
      Public Class    MathService
            Inherits System.Web.Services.Protocols.SoapClientProtocol

            Public Sub New()
                  MyBase.New
                  Me.Url                                                    =
"http://localhost/QuickStart/aspplus/samples/services/MathService/VB/MathService.a"& _
"smx"
            End Sub
           Public                                                    Function
<System.Web.Services.Protocols.SoapMethodAttribute("http://tempuri.org/Add")>      Add(ByVal
<System.Xml.Serialization.XmlElementAttribute("A", IsNullable:=false)> a As Integer, ByVal
<System.Xml.Serialization.XmlElementAttribute("B", IsNullable:=false)> b As Integer) As
Integer
                  Dim results() As Object = Me.Invoke("Add", New Object() {a, b})
                  Return CType(results(0),Integer)
            End Function
            Public Function BeginAdd(ByVal a As Integer, ByVal b As Integer, ByVal callback As
System.AsyncCallback, ByVal asyncState As Object) As System.IAsyncResult
                  Return Me.BeginInvoke("Add", New Object() {a, b}, callback, asyncState)
            End Function
            Public Function EndAdd(ByVal asyncResult As System.IAsyncResult) As Integer
                  Dim results() As Object = Me.EndInvoke(asyncResult)
                  Return CType(results(0),Integer)
            End Function
            Public                                                    Function
<System.Web.Services.Protocols.SoapMethodAttribute("http://tempuri.org/Subtract")>
Subtract(ByVal  <System.Xml.Serialization.XmlElementAttribute("A",  IsNullable:=false)>  a  As
Integer,  ByVal  <System.Xml.Serialization.XmlElementAttribute("B",  IsNullable:=false)>  b  As
```

```
Integer) As Integer
        Dim results() As Object = Me.Invoke("Subtract", New Object() {a, b})
        Return CType(results(0),Integer)
    End Function
    Public Function BeginSubtract(ByVal a As Integer, ByVal b As Integer, ByVal callback
As System.AsyncCallback, ByVal asyncState As Object) As System.IAsyncResult
        Return Me.BeginInvoke("Subtract", New Object() {a, b}, callback, asyncState)
    End Function
    Public Function EndSubtract(ByVal asyncResult As System.IAsyncResult) As Integer
        Dim results() As Object = Me.EndInvoke(asyncResult)
        Return CType(results(0),Integer)
    End Function
    Public                                                          Function
<System.Web.Services.Protocols.SoapMethodAttribute("http://tempuri.org/Multiply")>
Multiply(ByVal <System.Xml.Serialization.XmlElementAttribute("A", IsNullable:=false)> a As
Integer, ByVal <System.Xml.Serialization.XmlElementAttribute("B", IsNullable:=false)> b As
Integer) As Integer
        Dim results() As Object = Me.Invoke("Multiply", New Object() {a, b})
        Return CType(results(0),Integer)
    End Function
    Public Function BeginMultiply(ByVal a As Integer, ByVal b As Integer, ByVal callback
As System.AsyncCallback, ByVal asyncState As Object) As System.IAsyncResult
        Return Me.BeginInvoke("Multiply", New Object() {a, b}, callback, asyncState)
    End Function
    Public Function EndMultiply(ByVal asyncResult As System.IAsyncResult) As Integer
        Dim results() As Object = Me.EndInvoke(asyncResult)
        Return CType(results(0),Integer)
    End Function
    Public                                                          Function
<System.Web.Services.Protocols.SoapMethodAttribute("http://tempuri.org/Divide")>
Divide(ByVal <System.Xml.Serialization.XmlElementAttribute("A", IsNullable:=false)> a As
Integer, ByVal <System.Xml.Serialization.XmlElementAttribute("B", IsNullable:=false)> b As
Integer) As Integer
        Dim results() As Object = Me.Invoke("Divide", New Object() {a, b})
        Return CType(results(0),Integer)
    End Function
    Public Function BeginDivide(ByVal a As Integer, ByVal b As Integer, ByVal callback
As System.AsyncCallback, ByVal asyncState As Object) As System.IAsyncResult
        Return Me.BeginInvoke("Divide", New Object() {a, b}, callback, asyncState)
    End Function
    Public Function EndDivide(ByVal asyncResult As System.IAsyncResult) As Integer
        Dim results() As Object = Me.EndInvoke(asyncResult)
        Return CType(results(0),Integer)
    End Function
```

End Class
End Namespace
有了这四个文件，我们可以编辑一个批处理文件，执行如下的语句：

webserviceutil -c:proxy /pa:MathService.sdl /l:VB /n:MathServiceSpace

这样，我们就可以在客户端执行了。

5.2.1 小结

在这一章中，我们以提供计算加、减、乘、除的网络应用为例，详细的介绍了如何建立起一个完整的 web service 服务的步骤和注意事项，虽然这个例子和实际使用的应用环境有较大的差异，但基本方法应该是一致的。

第三章 数据交换

我们的这个例子说明了 DataSet-----一个基于 XML 技术的强大的数据分离技术，能够用 Web Service 方法返回。DataSet 能够在一个智能化的结构中存储复杂的信息和关系，这是 Web Service 的一个非常有用的方法。

通过 DataSets 的显示，你能够限制通过连接你的数据库服务器的测试。

GetTitleAuthors 方法连接一个数据库并且运行两个 SQL 语句，第一个返回颜色的列表，另外一个返回字体大小的列表。方法把两个结果用一个 DataSet 来存储，并返回一个 DataSet。

PutTitleAuthors 说明一个 Web Service 方法把 DataSet 当作一个参数并返回一个整数，这个整数就是在 DataSet 中的“ Table “ 表的行数。虽然这个方法执行起来有点简单，但是这个方法也能够与数据库服务器把过剩的数据聪明的合并在一起。

我们来看看这个例子，首先：

```
<%@ WebService Language="VB" Class="DataService" %>
```

这句话应该包括。我们还要引入这个名字空间：

```
Imports System.Web.Services
```

第一我们两个方法，Getcolor()：

```
Public Function <WebMethod()> Getcolor() As DataSet
  Dim MyConnection As SQLConnection = New
      SQLConnection("server=localhost;uid=sa;pwd=;database=howff")
```

```
        Dim MyCommand1 As SQLDataSetCommand = New
            SQLDataSetCommand("select * from color", MyConnection)
        Dim MyCommand2 As SQLDataSetCommand = New
            SQLDataSetCommand("select * from size", MyConnection)
        '数据的填充
        Dim DS As New DataSet
        MyCommand1.FillDataSet(DS, "color")
        MyCommand2.FillDataSet(DS, "size")
        Return DS
    End Function
```

Putcolor()方法：

```
    Public Function <WebMethod()> Putcolor(DS As DataSet) As Integer
        '返回行数
        Return DS.Tables(0).Rows.Count
    End Function
```

文件保存为 webservice.asmx，放在虚拟目录下，具体代码如下：

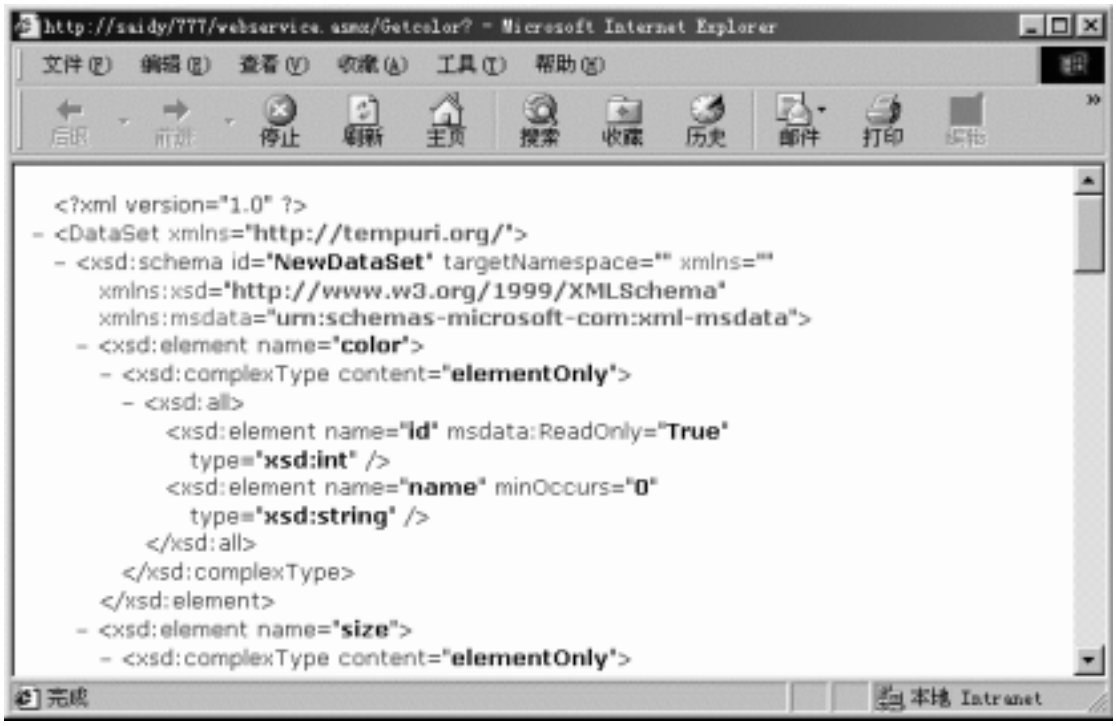
源文件：webservice\webservice.asmx

```
    <%@ WebService Language="VB" Class="DataService" %>

    Imports System
    Imports System.Data
    Imports System.Data.SQL

    '引入 System.Web.Services 名字空间
    Imports System.Web.Services

    Public Class DataService

        Public Function <WebMethod()> Getcolor() As DataSet
        '创建数据库连接和命令集
            Dim MyConnection As SQLConnection = New
                SQLConnection("server=localhost;uid=sa;pwd=;database=howff")
            Dim MyCommand1 As SQLDataSetCommand = New
                SQLDataSetCommand("select * from color", MyConnection)
            Dim MyCommand2 As SQLDataSetCommand = New
                SQLDataSetCommand("select * from size", MyConnection)
        '数据的填充
            Dim DS As New DataSet
            MyCommand1.FillDataSet(DS, "color")
            MyCommand2.FillDataSet(DS, "size")
            Return DS
```

```
        End Function
        '
        Public Function <WebMethod()> Putcolor(DS As DataSet) As Integer
            '返回行数
            Return DS.Tables(0).Rows.Count
        End Function

    End Class
```

运行：

点击“ Invoke “ 按钮，我们可以浏览到一个 xml 形式的文件。



如果我们点击了“ putcolor “ 连接后再点击“ Invoke “ 按钮，则得到不同的结果，有兴趣的读者不妨试一试，就象我们在上面的例子上所看到的一样。

5.3.1 小结

这一章我们学习了 web service 采用 xml 的方式在不同平台间传递数据的例子，xml 标准将是今后数据相互传输和转化的通用标准。

第四章 存取站点对象

我们的例子说明怎么样通过 Web Service 来访问 Web 站点的固有的东西如：Session 和 Application 属性，以及如何关闭 Session。

asmx 文件例子中的第一个方法，UHC 访问 Session，并在“ HitCount “ 变量上加 1。我们定义 Session 计数器方法：

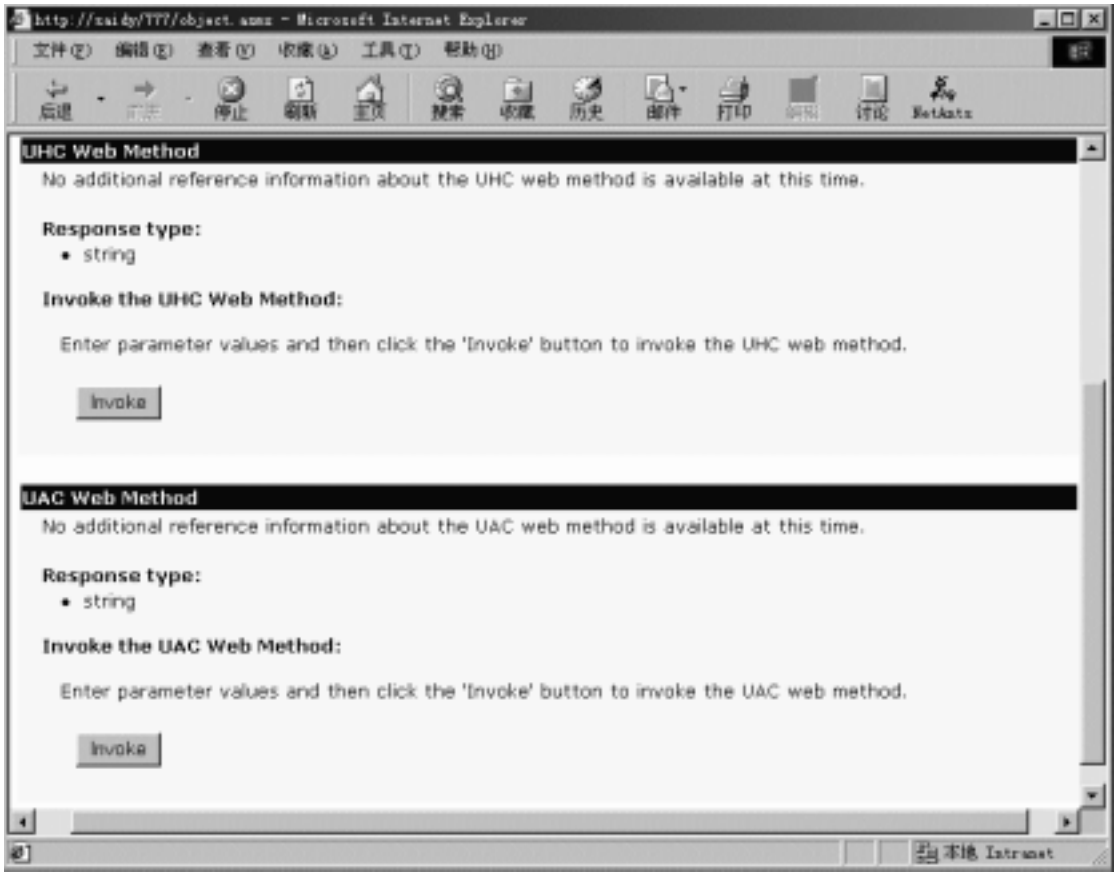
```
Public Function <WebMethod(EnableSession:=true)> UHC() As String
    If Session("HitCounter") Is Nothing
        Session("HitCounter") = 1
    Else
        Session("HitCounter") = Cint(Session("HitCounter")) + 1
    End If
 Return "你已经访问这个服务器 " & Session("HitCounter").ToString() & " 次了."
```

```
    End Function
```
注意在方法名称的前面，我们加上了”<WebMethod(EnableSession:=true)>” 这个修饰语句。在更新计数器的方法中：

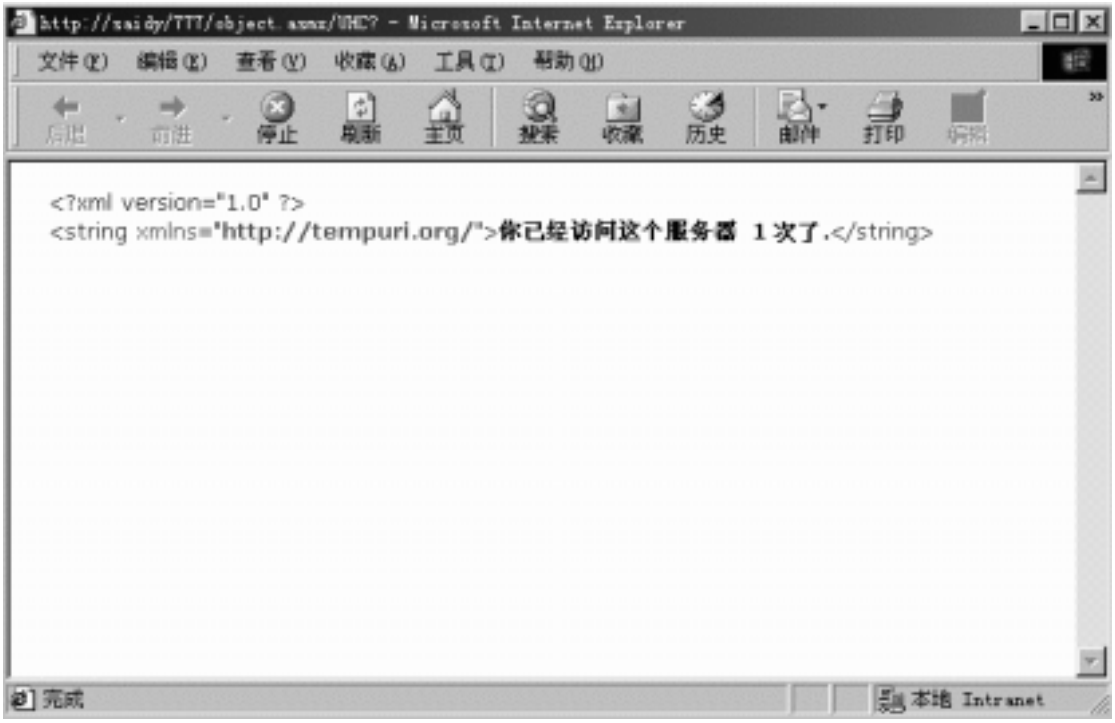
```
        Public Function <WebMethod(EnableSession:=false)> UAC() As String
            If Application("HitCounter") = Nothing
                Application("HitCounter") = 1
            Else
                Application("HitCounter") = CInt(Application("HitCounter")) + 1
            End If
            Return "服务被访问了 " & Application("HitCounter").ToString() & " 次."
        End Function
```
注意与上面方法的区别，我们在修饰方法名称的语句
“ <WebMethod(EnableSession:=false)> “ 不一样。下面是我们完整的例子代码：

```
<%@ WebService Language="VB" Class="SessionService1" %>
Imports System
Imports System.Web.Services

Public Class SessionService1 : Inherits WebService
    '增加计数器的值方法
    Public Function <WebMethod(EnableSession:=true)> UHC() As String

        If Session("HitCounter") Is Nothing
            Session("HitCounter") = 1
        Else
            Session("HitCounter") = Cint(Session("HitCounter")) + 1
        End If

        Return "你已经访问这个服务器 " & Session("HitCounter").ToString() & " 次了."
    End Function

    '更新计数器值的方法
    Public Function <WebMethod(EnableSession:=false)> UAC() As String

        If Application("HitCounter") = Nothing
            Application("HitCounter") = 1
        Else
            Application("HitCounter") = CInt(Application("HitCounter")) + 1
        End If
        Return "服务被访问了 " & Application("HitCounter").ToString() & " 次."
    End Function
End Class
```

下面是我们的运行效果：



点击上面的“ Invoke “ 按钮，有如下结果：



点击下面的“ Invoke“ 按钮，有如下结果（在此之前我们已经点击了很多回）：

5.4.1 小结

本章通过对传统记数器在 web service 中的实现为例，介绍了 web service Application 如何使用 session 对象、application 对象的方法。