[原文： http://wenku.baidu.com/view/f87b55f19e31433239689314.html](http://wenku.baidu.com/view/f87b55f19e31433239689314.html)

[WebServices简介](http://www.cnblogs.com/aijun/archive/2011/03/15/1984497.html)

先给出一个概念 SOA ，即Service Oriented Architecture ，中文一般理解为面向服务的[**架构**](http://lib.csdn.net/base/architecture)，

既然说是一种架构的话，所以一般认为 SOA 是包含了运行环境，编程模型，

架构风格和相关方法论等在内的一整套新的分布式软件系统构造方法和环境，

涵盖服务的整个生命周期。而在 SOA 的架构风格中，服务是最核心的抽象手段。

SOA 中的服务是构建在一些列基于开放标准的基础之上的，

Web 服务定义了如何在异构系统之间实现通信的标准化方法，

从而就使得 Web 服务可以跨越运行平台和实现语言，

同时也使得 Web 服务成为了实现 SOA 中服务的主要技术。

至于SOA 的话，太高深的技术，这里不予讨论（嘿嘿），本篇博文只介绍 WebServices 这项技术。

**引子**

有没有一种办法可以实现跨应用程序进行通信和跨平台进行通信呢？

换句话说，就是有什么办法可以实现我的应用程序 A 可以和应用程序 B 进行通信呢？

或者说是，我用 [**Java**](http://lib.csdn.net/base/java) 写的应用程序和用 . Net 开发的应用程序之间进行通信呢？

很多时候，上面提到的这些，我们是必须要使用的，**比如，一个跨应用程序吧，**

拿腾讯 QQ 来说吧，我估计每一个人都用过腾讯 QQ 上面的天气预报工具吧 **！ ！ ！**

**上面的这个天气预报功能是如何实现的呢？**

有一种办法，那就是腾讯公司放个卫星上天，并且在公司中成立一个气象部门，天天关注于天气，

然后每时每刻更新腾讯 QQ 上的这个天气预报信息，

确实，这种办法确实行得通，不过，要真这样做的话，估计马化腾也该被踢出去了（哪有这么蠢啊？），

那么上面这个是如何实现的呢？别急，且听我慢慢道来**~~~**

**然后，我们再来看看跨平台这个东东又是什么呢？**

这里主要是拿 . Net 平台和Java 平台来说明例子，

假若，有两个公司，每个公司呢都有自己的一个项目，一个公司呢使用 . Net 开发，一个呢，使用 Java 开发，

恩，本来呢，这两个是相互独立的，井水不犯河水，但是有一天，突然，这两个公司给合并了，

合并后的老总发现，如果把两个项目结合起来将会大大的赚一笔，为此，如何做？

因为要把两个项目结合在一起，那么这两个项目之间总应该通通信吧 **！！！**

可这两个项目又是基于不同的平台，怎么通信呢？麻烦了吧 **！！！**

而后再看一种情况，就是比如一个公司使用的服务器是 Windows Server 2008，

那么它如何和 IT 供应商的UNIX 或者是 [**Linux**](http://lib.csdn.net/base/linux) 服务器进行连接呢？也很复杂吧 **！！！**

**WebServices特点介绍**

WebServices 提供一个建立分布式应用的平台，使得运行在不同[**操作系统**](http://lib.csdn.net/base/operatingsystem)和不同设备上的软件，或者是用不同的程序语言和不同厂商的软件开发工具开发的软件，所有可能的已开发和部署的软件，能够利用这一平台实现分布式计算的目的。WebServices的思想是：使得应用程序也具有 Web 分布式编程模型的松散耦合性。

**WebServices的特点：**

**（1），WebServices 是自包含的。**即在客户端不需要附加任何软件，只要客户机支持 HTTP 和XML 就 OK 了。

**（2），WebServices 是自我描述的。**在客户端和服务端都不需要知道除了请求和响应消息的格式和内容外的任何事。

**（3），WebServices 是跨平台和跨语言的。**客户端和服务端均可以在不同的平台和语言环境中实现，同时，不必为了支持 WebServices 而更改现有的代码。

**（4），WebServices 是基于开放和标准的。**XML 和 HTTP 是WebServices 的主要技术基础，而 XML 和HTTP 早就成了业内标准了。

**（5），WebServices 是动态的。**

**（6），WebServices 是可以组合的。**也就是通过一个 WebService 访问另外一个 WebService 来达到组合的目的。通过组合 WebServices 便可以将简单的 WebServices 聚合成为实现更多复杂功能的复杂的服务。

**（7），WebServices 是松散耦合的。**它完全解耦了客户端和服务端。

**（8），WebServices 提供编程访问的能力。**换句话说，就是可以通过编写程序来访问Web 服务。

**（9），WebServices 是基于经过考验的成熟技术上构建的。**比如 XML 和 HTTP。

**（10），WebServices 提供打包现有应用程序的能力。**

**（11），WebServices 通过网络进行发布，查找和使用。**

上面这些特点呢，现在不清楚的话，也不用紧，等下还会有详细的说明的 **！！！**

**WebServices到底是什么？**

如果简单的说的话，WebServices就是一组函数库，不过这和我们平时概念中的函数库却又有所不同，我们平时所使用的函数库要么是自己写的（在自己的应用程序当中写一组函数库），

要么是调用底层的 API（操作系统 API 如Win32 API），上面的这两种情况有一个共同点，

那就是函数库是位于客户端本地的，比如，您调用 Win32 API的话，就是调用本地操作系统上的函数库，而这里提到 Web 服务也是一组函数库这个概念和上面提到的函数库这个概念的区别就在此处，**因为如果将Web服务看做一组函数库的话，那么这组函数库不是位于本地的，而是位于远程机器上（当然也可以是本地机器中）。**

何为 Web 服务？

也就是网络服务，那就是把网络上不知道哪个地方的一些函数看做是一组服务，然后我再通过网络就可以使用这些服务。

关于什么是 Web 服务，上面的说法那是山寨版的，稍微正经一点的说法是：

**Web 服务是一种部署在 Web 上的对象或者是应用程序组件。**

**Why WebServices？**

为什么需要使用 WebServices呢？这必须根据 WebServices 的特点以及其优势进行分析了。

**首先，上面呢，也说了，Web服务的话，就是一组网络上的应用程序组件，**

这样的话，您便可以通过在您的应用程序中使用 Web 服务来将您的应用程序提升到服务层面上来。

既然可以看做是一组服务了的话，那么当然就是可以提供给别个（别的应用程序）使用。

比如，我可以通过 Web 服务来公开一些接口给别人使用，至于这些要不要收费呢？那就看我心情了，前面举了腾讯 QQ 上查询天气的例子，这个例子呢，就可以在这里来做一个解释了，

在中国，应该只有一个卫星来进行天气预报的吧？腾讯也不可能为了天气预报而专门放个卫星上天吧？

可是腾讯 QQ 又确实是可以查询天气的，这里，便可以通过 Web 服务来解决。

首先，中国气象局应该是有一个卫星的，气象局根据卫星所返回的结果实时发布全国各地的天气状况，并且将这些天气信息以 Web 服务的形式公开，然后呢，腾讯 QQ 便可以通过这个 Web 服务来访问到天气状况了，再将这些天气状况反馈到 QQ 上就 OK 了。

**然后，上面提到了 Web服务是应用程序组件，既然是组件，那么就可以对这个组件重复的进行使用了，**

同时可以通过 Web 服务来实现将这个应用程序组件作为一个服务来进行使用，

更为强大的是，可以将多个 WebServices组合成为更为强大的 WebServices ，

并且是通过互联网哦**！！！**

这也是一大优点啊，

**然后呢，最基本的 WebServices是基于 XML 和 HTTP 的**

（当然这是最基本的 WebServices ，比如 WebServices 还可以通过 HTTPS 或者是 SMTP 来实现通信），

这又有什么好处呢？很明显，XML 和HTTP 这些都已经是标准了，

不论你是 Java 平台呢，还是 . Net 平台开发出来的（或者是是使用 Web 服务），既然我是使用 XML 和 HTTP 的话，我才懒得鸟你什么 Java 还是 . Net 呢，我也不管你是 [**linux**](http://lib.csdn.net/base/linux) 还是 Windows ，这一切都和我 Web 服务无关，

我关注的只是通过 HTTP 协议来传输 XML 就 OK 了，

至于这些 XML 是如何被服务提供者开发出来的或者这些 XML 是如何被服务请求者使用的，这些都和我无关，这里便可以看出 Web 服务的另一个优势了，那就是跨语言跨平台（实现协同工作），所以可以通过 Web 服务来实现不同应用程序和不同平台之间的通信。

Web 服务允许独立于实现服务基于的硬件或者是软件平台和编写服务所用的编程语言使用服务，

根据上面这两点呢，

便可以解决掉最开始提出的使用 Java 开发的应用程序如何和使用 . Net 开发的应用程序之间进行通信这一问题，

同时，也可以解决 Linux 或者是UNIX 和 Windows Server 2008 之间进行连接这一问题了。

最后就是通过使用不同的 Web 服务，也不管 Web 服务是那种编程语言实现的，

我们都可以从不同的平台和操作系统进行访问，**从而大大提高了不同应用程序共享数据和应用的能力。**

**并且 Web服务提供了构建 SOA 所必须得技术基础。**

从上面可以看出通过 WebServices解决了我们曾经想都不敢想的问题，这么强大的东西为什么不加以好好利用呢？

**WebServices体系结构**

在Web 服务的体系结构中，涉及到三个角色，

一个是 Web 服务提供者，一个是 Web 服务中介者，还有一个就是 Web 服务请求者，

同时还涉及到三类动作，即发布，查找，绑定，

**Web 服务提供者：**

可以发布 Web 服务，并且对使用自身服务的请求进行响应，

Web 服务的拥有者，它会等待其他的服务或者是应用程序访问自己。

**Web 服务请求者：**

也就是 Web 服务功能的使用者，它通过服务注册中心也就是 Web 服务中介者查找到所需要的服务，

再利用 SOAP 消息向 Web 服务提供者发送请求以获得服务。

**Web 服务中介者：**

也称为服务代理，用来注册已经发布的 Web服务提供者，并对其进行分类，同时提供搜索服务，

简单来说的话，Web 服务中介者的作用就是把一个 Web 服务请求者和合适的 Web 服务提供者联系在一起，

充当一个管理者的角色，一般是通过 UDDI来实现。

**发布：**

通过发布操作，可以使 Web服务提供者向 Web 服务中介者注册自己的功能以及访问的接口。

**发现（查找）：**

使得 Web 服务请求者可以通过 Web 服务中介者来查找到特定种类的Web服务。

**绑定：**

这里就是实现让服务请求者能够使用服务提供者提供的服务了。

**WebServices三种基本元素之 SOAP**

**SOAP 即 Simple Object AccessProtocol 也就是简单对象访问协议。**

SOAP 呢，其指导理念是“唯一一个没有发明任何新技术的技术”，是一种用于访问 Web 服务的协议。

因为 SOAP 基于XML 和 HTTP ，其通过XML 来实现消息描述，然后再通过 HTTP 实现消息传输。

SOAP 是用于在应用程序之间进行通信的一种通信协议。因为是基于 XML 和HTTP 的，所以其独立于语言，独立于平台，并且因为 XML 的扩展性很好，

所以基于 XML 的 SOAP 自然扩展性也不差。

通过 SOAP 可以非常方便的解决互联网中消息互联互通的需求，其和其他的 Web 服务协议构建起 SOA 应用的技术基础。

SOAP 协议的一个重要特点是它独立于底层传输机制，Web 服务应用程序可以根据需要选择自己的数据传输协议，可以在发送消息时来确定相应传输机制。

由于 HTTP 协议本身的一些特点和局限性，使得当 SOAP 使用HTTP 绑定的 Web 服务并不能满足某些企业应用的需求。

比如，HTTP 不是一个可靠传输协议，所以有可能在传输过程中出现问题，然后 HTTP 协议基于Request/Response 模型，也就是说客户端需要在等待响应消息接收完成后才能继续执行，而此时如果响应时间过长呢？

基于上面的这些需求，便需要选择合适的传输协议了。

关于这方面的内容的话，有点深奥了，有兴趣的可以去看看 IBM 的一些关于这方面内容的介绍。

还有一点需要提及一下，那就是 SOAP 是可以绕过防火墙的，将来将会作为 W3C 的标准进行发展。

**WebServices三种基本元素之 WSDL**

WSDL 即Web Services Description Language也就是 Web 服务描述语言。**是基于 XML的用于描述 Web 服务以及如何访问 Web 服务的语言。**

服务提供者通过服务描述将所有用于访问 Web服务的规范传送给服务请求者，

要实现 Web服务体系结构的松散耦合，服务描述是一个关键，不管是请求者还是服务提供者，通过服务描述便可以不必了解对方的底层平台，编程语言等，服务描述与底层的 SOAP 基础结构相结合，足以封装服务请求者的应用程序和服务提供者的 Web服务之间的这个细节。

**WSDL 描述了 Web服务的三个基本属性：**

（1）服务所提供的操作

（2）如何访问服务

（3）服务位于何处（通过 URL 来确定就 OK 了）

**WebServices三种基本元素之 UDDI**

UDDI 即 Universal Description，Discovery and Integration，也就是通用的描述，发现以及整合。

WSDL 呢，用来描述了访问特定的 Web 服务的一些相关的信息，可以在互联网上，

或者是在企业的不同部门之间，如何来发现我们所需要的 Web 服务呢？而 Web 服务提供商又如何将自己开发的 Web 服务公布到因特网上，这就需要使用到 UDDI 了，UDDI的话，是一个跨产业，跨平台的开放性架构，可以帮助 Web 服务提供商在互联网上发布 Web 服务的信息。

UDDI 呢是一种目录服务，企业可以通过UDDI来注册和搜索Web 服务。

简单的话，UDDI 就是一个目录，只不过在这个目录中存放的是一些关于 Web 服务的信息而已。并且 UDDI 通过SOAP 进行通讯，构建于 . Net 之上。

**开发 Web服务的方式**

**（1）开发阶段：**

        实现一个 Web 服务，使这个 Web 服务能响应和接收 SOAP 消息，

      （这个呢，其实可以通过 Visual Studio 来帮助实现），

       定义好逻辑模块（这个 Web 服务总要干点事情吧），

       然后再撰写 WSDL 文件（这个呢，开发工具会自动生成的，不需要人工撰写了）

**（2）部署阶段：**

       指定 Web 服务的传输协议，将 Web 服务注册到相应服务描述部署文件（这些也是可以由工具来自动完成的）

**（3）发布阶段：**

       将 Web 服务的接口和调用的地址公开给客户端调用，

       常用的发布方式为基于 Web 提供的WSDL 的链接，当然 UDDI 也是一个选择。

**总结一下 WebServices的优点**

其实呢，前面介绍的都是关于 WebServices 的优点，在这里就只是浅要的总结一下了。首先，WebServices 是基于 Internet 和异构平台的应用，这样便可以方便的实现通过网络来通信，同时可以实现在不同的平台之间共享数据。然后就是，WebServices 是基于 XML 和HTTP 的，也就是基于标准和开放的，基于 XML 的话，扩展性自然好，自然跨语言。基于 HTTP 的话，自然跨平台了。最后，再回忆一下 WebServices 是一种应用程序组件吧，这样便可以将 WebServices 重复使用了。

**谈谈 WebServices 的缺点**

首先就是由于 XML 文件的难以解析，所以在使用 Web 服务的时候，会消耗较多的 CPU 和内存资源，而后，SOAP 是基于XML 的，所以在网络传输中传输的是 XML 文件，但是由 XML 文件相比于二进制文件来说，要大很多，自然就会消耗更多的网络资源了。而后，由于通过 WSDL 解耦了Web 服务提供者和请求者，且 SOAP 消息时从发送者向接收者单向传送的，这在一定程度上造成了 WebServices 是一种无状态服务，尽管现在 . Net 中可以通过 Session 来实现在客户端和服务端共享一些数据，但是单单依靠 Session 来实现客户端和服务端的状态交互也太牵强了吧。WebServices 在数据绑定上也存在一些缺陷，因为所有的数据在传输中都是使用的 XML 来实现的，因此，需要在二进制数据和 XML 之间进行一个转换（通过序列化和反序列化来实现），而在转换过程中有可能出现语义丢失的情况。最后就是 WebServices 的技术要求相对比较高，因为涉及到底层的 HTTP 协议以及SOAP ，WSDL 和UDDL 这三大平台元素，

所以学习的曲线也是比较长的，

当然，在 . Net 中可以通过 Visual Studio 非常快速和简单的开发和访问一个 Web 服务，但这只限于在简单的使用上，而对于本质的东西，是比较难的。

**后续**

正如题目所言，是 WebServices 简介，既然是简介的话，那么自然就是以简为目标了，说明一下的是，上面的这篇博文呢，源自前几天做的一个关于 WebServices 的演讲，演讲的 PPT 还存有，有兴趣要的可以留个邮箱的。

**. Net中 WebServices 的实战**

**下面呢，就来具体看看在 . Net中如何开发一个WebServices 以及如何使用这个 WebServices**

开发环境：

Windows 7 下IIS 7.5

Visual Studio TeamSystem 2008

Sql Server 2008

**首先来看看如何开发一个 WebServices**

先建立一个 ASP[**.NET**](http://lib.csdn.net/base/dotnet) 应用程序项目，然后再在项目中添加一个 WebServices 服务，

然后就是在这个 WebServiceTest中编写业务逻辑了，

本次实例的业务逻辑呢就是从[**数据库**](http://lib.csdn.net/base/mysql)“图书馆管理系统”中取出所有的读者的信息。

**WebServiceTest.asmx中的代码如下**

**usingSystem.Web.Services;  
usingSystem.Data;   
usingSystem.Data.SqlClient;   
usingSystem.Web.Configuration;**

namespace **WebServiceDemo**  
{   
    [WebService(Namespace = "[http://tempuri.org/")]](http://tempuri.org/%22)%5d)  
   [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]  
    [System.ComponentModel.ToolboxItem(false)]

    public class**WebServiceTest :System.Web.Services.WebService**    {   
        **[WebMethod]**       public DataSet **GetAllReader**()  
        {   
            **DataSet ds = newDataSet();**           string **connStr** =  
               WebConfigurationManager.ConnectionStrings["DBConnString"].ConnectionString;  
            string **sqlStr** = "SELECT[ReaderID],[ReaderIDType],[ReaderName]," +  
                                  "[ReaderSex],[ReaderBirth]" +   
                           "FROM [图书馆管理系统].[dbo].[Reader]";  
            using (SqlConnection sqlCon = new SqlConnection(connStr))  
            {   
               using (SqlCommand sqlCom =sqlCon.CreateCommand())  
               {   
                   sqlCom.CommandType = CommandType.Text;   
                   sqlCom.CommandText = sqlStr;   
                  using (SqlDataAdapter sqlDa = newSqlDataAdapter(sqlCom))  
                   {   
                       sqlDa.Fill(ds);   
                   }   
               }   
            }   
            **return ds;**  
        }   
    }   
}

**然后我再在这一个项目 WebServiceDemo中添加一个页面 Test . aspx**

**来实现访问自身应用程序中的 WebServices**

**（Test. aspx和 WebServiceTest . asmx 位于同一应用程序中）**。

这个 Test . aspx 呢非常简单，仅仅在上面放了一个 GridView ，然后稍微写点 Code-Behind 就 OK 了，

**其代码如下：**

using System;

namespace **WebServiceDemo**  
{   
    public partial class Test : System.Web.UI.Page   
    {   
        protected void Page\_Load(objectsender, EventArgs e)   
        {   
            if(!IsPostBack)   
            {   
               **WebServiceTesttest= new WebServiceTest();**  
               GridView1.DataSource = **test.GetAllReader();**  
               GridView1.DataBind();   
            }   
        }   
    }   
}

**再来浏览一下 Test . aspx页面**

可以看出已经达到了预定的效果，也就是从数据库中通过 WebServices取出了数据。

**而从上面的代码可以看出，仅仅是将 WebServices看做是一个类了，**

**将其作为一个类来进行使用（实质上也就是一个类而已）。**

下面我们还需要看一种情况，

**那就是，实现在另外的一个应用程序中访问上面建立的 WebServices。**

其实这种情况呢，就是和访问网络上的 WebServices 一样了，

比如腾讯 QQ 就是使用这种方式来实现的，

为了模拟这种实现，我首先将上面建立的这个 ASP[**.net**](http://lib.csdn.net/base/dotnet) 应用程序 部署到 IIS 上面，

且指定了一个端口为 81

**然后我再建立一个项目 TestWebServices**

并且在这个项目里面也添加一个页面 Test . aspx

在Test . aspx 上也只放一个 GridView 控件。

**然后就要给这个项目添加 Web服务的引用了（右键TestWebService 点击“添加 Web 引用”）**

如果您要访问的是互联网上的 Web服务，比如查询天气，

那么就需要在上面的 URL中写入 Web 服务所在的地址，然后“前往”就 OK 了，

由于本次的演示，我只是把我的 Web服务放在了本地的 IIS 上面，

所以在此处呢选择“本地计算机上的 Web服务”，

**从上面的截图中就可以看出，在 81号端口上面我有一个 Web 服务，**

**就是前面的 Demo中建立的 Web 服务 WebServiceTest**

然后我选择这个 Web 服务，单击它即可，

上面的这幅截图中便可以看出我在 Web 服务WebServiceTest 中公开的接口了，

由于我只在其中写了一个接口 GetAllReader ，所以在这里便只显示了一个了。

在这一步中，您便可以添加这个 Web 引用了，不过要注意的是，

如果在这一步添加 Web 引用的话，那么这个 Web 服务中所有被公开的方法都会被添加到您的项目中，

比如，如果我在上面的 Web 服务中还有一个 GetAllName 的方法的话，

那么在这一步添加 Web 引用的话，就会将 GetAllReader 和 GetAllName 全部添加到您的项目当中，

但是有时候，这样会太浪费了，因为我可能根本就不需要使用 GetAllName 而只需要 GetAllReader，

此时，可以单击上面的 GetAllReader 进入下一步，

在这一步中添加 Web 引用的话，那么就只会在项目中添加 GetAllReader 这个方法的引用了，

我们在这里使用这种方法来添加 GetAllReader 的引用。

单击“添加引用”，此时可以看到在项目中生成一些文件，

（这里呢，其实就是代理模式来实现了）

既然在项目中引用了这个 Web服务了，

那么下一步就是在 Test . aspx中使用这个 Web 服务了。

**看看 Test . aspx的 Code-Behind 吧**

using System;

namespace TestWebService  
{   
    public partial class Test : System.Web.UI.Page   
    {   
        protected void Page\_Load(objectsender, EventArgs e)   
        {   
            if(!IsPostBack)   
            {   
                //WebServiceTest.GetAllReader 这一段是我引用后的服务名   
               WebServiceTest.GetAllReader.WebServiceTesttest =  
                   new **WebServiceTest.GetAllReader.WebServiceTest**();  
               GridView1.DataSource = **test.GetAllReader**();  
               GridView1.DataBind();   
            }   
        }   
    }   
}

**下面就来看一看效果了**

从上面的效果便可以看出，我们已经成功在另外的应用程序中访问了 Web 服务，

也可以得出 Web 服务实现了在不同应用程序之间数据的共享。

如果，读者对通过网络 URL 来访问WebServices 有疑问的话，

可以参考一下笔者的另外一篇稍微带有 WebServices 性质的博文，

在其中实现了一个访问互联网上提供的天气查询 Web 服务。

<http://www.cnblogs.com/QinBaoBei/archive/2010/03/30/1700898.html>

上面呢，通过几个 Demo 对WebServices 在 .Net 中的实战进行了一个简单的应用了。

在这里一切似乎都和前面所提到的 SOAP ，WSDL，UDDI 均扯不上关系，

其实不然，只不过这些全部都由别个（工具）给你完成了，而你只是简单的开发一下逻辑就 OK 了，

不过呢，简单归简单，理解前面的一些原理还是很有必要的。

这里还点一下，就是上面呢，我通过 WebServices 来查询数据库并且返回一个 DataSet，

不过，大家有没有想过，如果这个 DataSet 中的数据量有很大呢，比如 100万条，甚至更多，

这样的话，在网络传输中那不慢的要死啊，其实有一种比较好的解决办法，那就是压缩，

关于压缩呢，又有好几种方法，比如 GZIP 可以将其压缩到原来的一半以上，

这几种方法各有所长，一个晚上又过咯，很晚啦，以后有时间的话，可能会写一篇这方面的博文吧 **！！！**

关于WebServices 的简介就到这里了 **！！！**