Foreword by Andrew Layman

Microsoft Corporation

高级.NET开发系列

Microsoft .net Development Series

PEARSON Addison Wesley

.NET Web Services

.NET Web Services
Architecture and Implementation

# .NET Web Services
## 架构与实现

[美] Keith Ballinger 著
张晓坤 谭立平 袁纯良 译

微软公司
Web Services 程序经理
亲自撰写。
内容权威经典，示例丰富！

Keith Ballinger

中国电力出版社
www.infopower.com.cn

Pearson Education

高级.NET开发系列

# .NET Web Services
## Architecture and Implementation

# .NET Web Services
## 架构与实现

[美] Keith Ballinger 著
张晓坤 谭立平 袁纯良 译

中国电力出版社
www.infopower.com.cn

.NET Web Services: Architecture and Implementation (ISBN 0-321-11359-4)
Keith Ballinger
Copyright ©2003 by Pearson Education, Inc.
Original English Language Edition Published by Pearson Education, Inc.
All rights reserved.
Translation edition published by PEARSON EDUCATION ASIA LTD and CHINA ELECTRIC POWER PRESS, Copyright © 2004.

本书翻译版由 Pearson Education 授权中国电力出版社独家出版、发行。
未经出版者书面许可，不得以任何方式复制或抄袭本书的任何部分。

本书封面贴有 Pearson Education 防伪标签，无标签者不得销售。
北京市版权局著作权合同登记号 图字：01-2004-4933

图书在版编目（CIP）数据
.NET Web Services：架构与实现 /（美）贝列哲著；张晓坤等译. —北京：中国电力出版社，2004
（高级.NET 开发系列）
ISBN 7-5083-2733-0
I.N... II.①贝...②张... III.互联网络－网络服务器 IV.TP368.5
中国版本图书馆 CIP 数据核字（2004）第 094362 号

丛 书 名：高级.NET 开发系列
书　　名：.NET Web Services：架构与实现
编　　著：（美）Keith Ballinger
翻　　译：张晓坤 谭立平 袁纯良
责任编辑：牛贵华
出版发行：中国电力出版社
地址：北京市三里河路 6 号　邮政编码：100044
电话：（010）88515918　传　真：（010）88518169
印　　刷：北京丰源印刷厂
开本尺寸：185×233　1/16　印　张：15.5　字　数：332 千字
书　　号：ISBN 7-5083-2733-0
版　　次：2004 年 12 月北京第 1 版　2004 年 12 月第 1 次印刷
定　　价：29.80 元
版权所有　翻印必究

# 译者序

一时间 Web Services（有些地方译为 Web Service）成了下一代互联网的代名词，被炒得沸沸扬扬。有人说，它为形形色色的软件小模块提供了“标准胶水”，将企业应用统一到 Web 层面上。在 IBM 的开发者园地 developerWorks 中有一篇文章“为什么需要 Web Service?”（http://www-900.ibm.com/developerWorks/cn/webservices/ws-wsar/part1/index.shtml）给出了相对全面的答案。然而，时至今日，许多人对 Web Services 还停留在想像或是概念理解的阶段。拨开那些令人眼花缭乱的对 Web Services 的远景描绘，好像现在能够实现的也只有与 Web Services 相关的协议和技术标准，例如，SOAP、WSDL、UDDI、XML 等。标准的意义不仅仅意味着能让 Web Services 行得更远，走得更快，在其背后，更是代表着制定标准的组织的各自利益。WS-I（参见 http://www.ws-i.org/）、W3C（http://www.w3c.org）及 OASIS（Organization for the Advancement of Structured Information Standards，结构化信息标准推动组织；其官方网址为 http://www.oasis-open.org/home/index.php）等标准组织及机构你争我夺，各大厂商在这些组织中来来往往，竞争者也能成为合作者，而技术就在这种竞争与合作的交错中不断进步。

那么，什么是 Web Services 呢? Web Services 是一系列标准和正在发展中的标准，它们是由 W3C 设计和指定的。Web Services 是基于互联网标准和 XML 技术建立的，用以描述、发布到网络供其他应用程序调用。也就是说，这是一种分布式计算体系结构，或者更进一步地说，这是一种松耦合的系统。

从前，分布式的应用程序逻辑需要使用分布式的对象模型，例如，DCOM、CORBA、RMI 等等。这些系统有一个共同的缺陷，那就是它们无法扩展到互联网上：它们要求服务客户端与系统提供的服务本身之间必须进行紧耦合，即要求一个同类基本结构。这样的系统往往十分脆弱，如果一端的执行机制发生变化，那么另一端便会崩溃。也许有人说，要求提供紧耦合的基本结构是无可厚非的，许多应用程序均是基于这种系统构建而成的。但是，当各个公司需要相互合作或者信息技术提供商要扩大业务范围时，就很难实现单一而统一的基本结构。你根本无法保证你希望与之进行远程通信的管道的另一端，具备所有你需要的基本结构：对于它使用的操作系统、对象模型或编程语言，你可能一无所知。

相反，Web Services 彼此是松耦合的。连接中的任何一方均可更改执行机制，却不影响应用程序的正常运行。从技术角度讲，人们已转向使用一种基于消息的异步技术来实现高可靠性的系统性能，通过使用诸如 HTTP、简单邮件传输协议（SMTP）以及至为重要的 XML 来实

现统一的连接。

目前，Web Services 的开发和部署将会大量依赖 J2EE 和.NET 这两个平台及其上面的开发工具。而.NET 的集成度更高一些，工具也会更方便一些。从.NET 框架的角度看，所有组件都可以是 Web Services，而 Web Services 也只是一种组件而已。事实上，.NET 框架提取出微软组件对象模型（COM）的精华，将它们与松耦合计算的精华有机地结合在一起，生成了强大、高效的 Web 组件系统：简化了程序员的“管道”操作，深入地集成了安全性，引进了基于 Internet 的操作系统，极大地改善了应用程序的可靠性和可扩展性。

本书作者是微软公司最早研究 Web Services 技术的主要领导者 Keith Ballinger 先生。如果你一直在关注这项技术，那么我想这本书一定可以为你提供你正在寻找的信息。本书对 Web Services 所使用的传输协议、接口定义和服务发现机制（UDDI）、安全和消息基础结构以及底层技术（XML、SOAP、TCP/IP 和 HTTP），从概念的描述到具体的实现，都有其独到的见解。XML 序列化如何影响 Web Services？什么是 SOAP 协议？WSDL 究竟有什么用处？我们为什么要使用 UDDI 等等。可能你已经开始创作 Web Services，也可能没有，但我相信 Keith Ballinger 一定会教给你很多你想知道的东西！

本书由张晓坤、谭立平和袁纯良共同翻译。一如既往，这里首先要感谢中国电力出版社给了我们这次机会。感谢本书的编校人员，正是他们一丝不苟、兢兢业业的工作，保证了这本书的质量。感谢我的老朋友高军，他的信任和鼓励一直是我们的动力。还要感谢卢青峰，他对本书的支持网站 http://www.coreader.com 做了许多工作。虽然翻译过程中我们小心谨慎，但仍然难免有所疏漏，欢迎批评指正！我们将在 http://www.coreader.com 上提供本书的勘误以及相关代码。敬请关注！

张晓坤

2004 年 8 月

此为试读，需要完整PDF请访问：www.ertongbook.com

## 对本书的赞美

“自 Web Servies 诞生之日起，Keith Ballinger 就是微软公司 Web Services 方面研究的主要领导人了。任何在微软公司 Web Services 平台上工作的人都将从本书获益，因为 Keith 的视角是独特的。”

——Bob Beauchemin，DevelopMentor 公司

“我认为这是一本非常优秀的书，它有高级的例子和示例代码，并且展示了底层之下.NET 的工作方式：它胜过我所读到的任何一本关于 Web Services 的书籍……这本书对.NET 的底层工作以及它如何与 Web Services 协同工作进行了深入地描述……”

——Len Fenster，微软公司首席顾问

“这本书对于 Web Services 来说，是一个很好的导论，它为我们提供了十分具体的信息，用以帮助我们理解 Web Services 的原理和实现问题……Ballinger 清楚地概述了任何想要实现 XML Web Services 的组织都应该考虑的基本架构主题。”

——Colin Bowern，微软公司顾问

“这本书涵盖了 Web Services 的所有主要组件的信息：传输协议、接口定义和服务发现机制、安全和消息基础结构以及底层技术（XML、TCP/IP 和 HTTP）。对于每个主题的描述都既易于理解又非常完整。所有例子都有助于更清晰地阐述内容。”

——Max Loukianov，Solomio 公司

# 序

在所有现代工业当中，计算机产业最生动地反映了下列特点，即自由资本、较高的就业率和相对较少的规范，这些特点使这一行业为人所乐道。这为企业家和其他产品、市场、金融、分配和团体组织方面的改革者提供了一个舞台。计算机产业从来都不是一成不变的，它粉粹了所有可能的保守估计。

在 1943 年，IBM 当时的董事长 Thomas Watson 先生说了一句名言："我想全世界大概需要 5 台计算机。"如果我们以那个时代的眼光来看，情况确实如此：购买并且维护那样庞大又昂贵的计算机，只有迫切需要它的大型组织才会那么做。

大无畏的创造者们创建了一个新技术流派，而企业家和投资者围绕它们建立了新型的产业。到 20 世纪 60 年代，在 Remington Rand 等公司及 IBM 的领导下，计算机变得更小更便宜，商业组织已经可以买得起一台"大型机"计算机，用于集中进行管理活动。到 20 世纪 70 年代，半导体技术的发明导致了微型计算机的诞生。某些公司（如 DEC）果断地打破了大型机的价格和集中式的模型，使得某些部门和小型组织也可以拥有计算机。然后是集成电路、风险资本、Intel 微处理器芯片、微软 Windows 和 20 世纪 80 年代个人计算机的层层演变。大多数计算机将会更小、更便宜并且更专注于独特的用途，永远都是这样。在 20 世纪 90 年代，通信业的投资与改革打破常规，通信也被集成到计算机架构中，究其原因是带宽成本的降低使得中央计算机充当起服务众多个人计算机的角色。

以 World Wide Web 为代表的基于浏览器的计算，取得了巨大的成功，但它仅仅处理了一种计算风格，即与人即时交互的远程计算机。它没有充分利用广泛分散的个人计算。World Wide Web 的发明——廉价的计算机、廉价的通信、可扩展的协议格式和基于协议的异步的松耦合——仍然没有被广泛地应用到代表人的意愿，而与其他计算机交互的计算机上。

这种情况即将发生改变。在 1996 年，XML（eXtensible Markup Language，可扩展标记语言）诞生了，并在随后的几年内被应用于 SOAP（Simple Object Access Protocol，简单对象访问协议）和 XML Web Services 的开发。定义：XML Web Service 是一种计算资源，只根据它执行的功能进行定义；Web Services 之间通过协议使用 XML、SOAP 和 WSDL 进行交互，并且交互双方可以不需要人为干预。

为了追逐趣味和利润，商业组织和技术改革者们创造了一个又一个令人激动的发明。你将能读到这些历史和最新的描述，它是由在广泛推进 XML Web Services 技术的过程中起关键作用的人所撰写的。

——Andrew Layman

# 前　言

这本书是数年工作的结晶。不只是我个人的工作（虽然写这本书经常至深夜），也是微软公司.NET 框架和 XML 消息小组许许多多的人多年共同努力工作的结果。还有一些其他的公司和天才的个人，例如 IBM 的 Sam Ruby，为这门技术的普及起到了重要的作用。

但也出现了一些问题：为什么在这门技术上会投入如此之多的人力和资金呢？为什么微软和其他公司会意识到 Web Services 是一项有巨大潜力的工业革命呢？这本书可能不能给你一个完整的答案，但我试图传递出我对于 Web Services 的理解中最重要的一些信息，特别是那些在.NET 之下的信息。在这么做的同时，我也希望你能看到，这门技术是多么精彩！

许多关于 Web Services 的书主要关注具体的技术，以及如何使用类库构建 Web Services 和客户端；还有一些书试图给出 SOAP、WSDL（Web Services Description Language）和其他技术的一些概述。那么，什么是 Web Services 呢？它们为什么会存在？当然我还是不能完整地回答这些问题，但我能帮助感兴趣的人更好地理解这门技术，并且由此设计和架构更好的 Web Services。我试图以一种方式，那就是解释它们为什么存在的原因来表达这些内容。

当然，作为在微软的.NET 框架之下构建 Web Services 的程序经理，炫耀这点东西让我觉得有点迫切。我切实地感到我为构建最佳 Web Services 技术所作的事情使得我适于带领你学习这门技术的主要特征，大多数代码示例都使用了 C#和 ASP .NET Web Services。

我将本书的结构设计成既可以从前向后读，也可以随意选读。尽管每一章是构建在前一章之上，但大多数章节单独阅读也是很有益处的。本书由 15 章组成：

- 第 1、2 章解释了什么是 Web Services 以及构成 Web Services 的标准。
- 第 3～6 章是关于.NET 框架如何让开发人员构建 Web Services 应用程序的深入描述。
- 第 7～14 章回过头来讲述关于构建 Web Services 架构的规范（从 HTTP 到 SOAP，再到 WS-Security）。
- 第 15 章陈述了关于架构和设计 Web Services 应用程序的一些建议。

——Keith Ballinger

# 致　谢

没有人是孤立存在的，任何创作者都不只是一个小小的次大陆，而是与大群个体息息相关的。

所以，这本书的出版离不开许多人的帮助。

感谢我的妻子 Lara，没有她就没有这本书的面世。

感谢我的母亲，没有她就没有今天的我。

感谢微软公司中所有为.NET Framework 和 Web Services 作出贡献并使之成为现实的出色的人们，特别感谢 Stefan Pharies、Ramesh Seshadri、Alex Dejarnett、Elena Kharitidi、Kim Shearer 和 John Koropchak。

感谢扩展 WSDK 组中的全体成员：Hervey Wilson、Vick Mukherjee、HongMei Ge、Jim Huang、B.C.Kim、Sidd Shenoy、David Bradley、Sidney Higa、Oliver Sharp、John Shewchuk、Chris Kaler、Eric Zinda、Robert Wahbe、Don Box、Brad Severtson，还有 Andrew Layman。

最后，对 Addition-Wesley 的有关人员及书稿的审阅者，尤其是 Mary O'Brien、Alicia Carey、Jody Thum 和 Dan Edgar 表示诚挚的感谢。

# 目　录

此为试读，需要完整PDF请访问：www.ertongbook.com

# 1

# Web Services 简介

为 Internet 提供连接组织的底层软件和硬件，代表了在过去几十年中某些最复杂的技术。就在几年以前，Internet 还只是少数人用于发送 e-mail 的网络。似乎在一夜之间，基于 HTTP 和 HTML 的 Web 页面出现了。HTTP（Hypertext Transport Protocol，超文本传输协议）是一个应用程序级的协议，它很容易实现和调试。基于 HTML（Hypertext Markup Language，超文本标记语言）的 Web 页面使用人类可读的格式创作，并且 Web 浏览器在解释时也更为灵活。

在 HTTP/HTML 发展的随后几年，XML 开始出现了。XML 是一种简单、易懂的数据标记格式。依据这种单一而简单的数据编码方式，要创建一个将 HTTP 和 XML 组合进 Web Services 的新型应用程序框架就不再困难了——这使得开发人员可以使用在以前是不可能的方式创建分布式应用程序。我之所以说 Web Services 是一个令人激动的技术，是因为 Web 架构从根本上不同于许多其他网络和分布式程序方法学。Web Services 继承了 Web 的许多良好特征，但同样也继承了一些缺陷。

本章概述了 Web Services 的这些特征以及缺点。从介绍 Web Services 开始，包括工作定义，并且将它们放在了分布式应用程序开发的大背景中。读完本章后，对于什么是 Web Services，以及这门技术将会有如何广阔的软件开发前景，你将会有了一个更清楚的理解。

## 问题：共享数据

简单地说，计算机需要共享数据。许多情况都证实了这点：商业交易需要在合作伙伴之间共享信息。一个部门需要向另一个部门发送数据。消费应用程序需要与其他消费应用程序协同工作。

最近，微软确定了能够使用 Web Services 的不同应用程序类型，它们是：

- 数据提供者，例如，诸如股票报价的数据提供者。
- 企业对企业的集成处理，例如，从一个公司向另一个公司发出购买订单。
- 与众多合作伙伴，甚至是竞争者之间的集成。
- 企业应用程序集成，例如，一个公司的 e-mail 数据库与其人力资源（HR）数据库之间的集成。

Web Services 的一个典型应用就是辅助桌面应用程序。辅助桌面应用程序通常是小型

Windows 应用程序，它可以让桌面操作员查询内部顾客数据，然后使用他们在辅助桌面调用过程中收集的信息更新这些数据。这种类型应用程序的开发人员需要“拉出（pull）”来自中心雇员数据库中的每个雇员信息。他们还需要从他们自己的辅助桌面数据库中“拉出”数据，并且修改它们。

## 解决方案：分布式应用程序开发

分布式应用程序开发（distributed application development）是一种从一台机器向另一台机器获取数据的技术与工程。这些数据可以是不确定的变量：购买订单、顾客数据、π的100～200位等等。曾经有许多技术用于构建来回传送数据的应用程序。CORBA（Common Object Request Broker Architecture，通用对象请求代理架构）、RMI（Remote Method Invocation，远程方法调用）和DCOM（Distributed Component Object Model，分布式组件对象模型）就是其中的几种。

然而，所有这些技术都有缺陷，并且没有一个可以在普遍存在的不同环境广泛流行。大多数技术甚至没有试图把这当成一个设计目标。例如，DCOM是基于COM（Component Object Model，组件对象模型）的，而后者是基于一种二进制标准，实际上不能在微软Windows平台之外部署。尽管跨平台使用DCOM在技术上是可能的，但认为DCOM能在这种环境中提供服务，仍然只是一个美好的想法。没有人曾对此做出真正的尝试，并能以一种稳定而成本划算的方式成功地运作它。

通用对象请求代理架构或CORBA，是一种分布式技术（实际上，是一种技术规范），它可以在不同的环境中工作。然而，由于诸多原因，包括许多开发人员发现实现基于CORBA的应用程序和服务很难，CORBA也没有真正流行起来。

简单对象访问协议（Simple Object Access Protocol，SOAP），是分布式应用程序领域的新举措。Web Services技术，例如SOAP，对于分布式应用程序开发比早先的技术更快捷。在SOAP得以成功的众多因素当中，最重要的有如下几点：

- 适当地平衡了Web架构（例如，XML和其他Web标准，诸如HTTP）
- 使用了基于模块的设计
- 创建了一个消息传递架构，它并不强调特定的实现、编程模型或语言
- 假定在处理过程中将会出现失败，并且允许处理过程很容易地侦测这一点

随后的几节将谈到促使SOAP成功的这些原因。

## Web架构

Web架构假定并且优美地处理了失败、延迟以及所有其他的现实世界的问题。这也是Web

Services 对于大多数继承的部分所具有的主要特征。当你启动一个 Web 浏览器时，你将预期有一些网页、图形和其他你试图下载的文件，可能向你询问凭据，或者有固定的位置，或者简单地不再存在。Web Services 将面临这个现实环境的挑战。

假定失败与先前相比是一个更高级的概念。例如，假设你构建了一个下载电影的应用程序。随着电影大小的增加，丢失包的可能性也同样会增加。作为这个应用程序的开发人员，你有几个选择：如果丢失了任何一帧画面，可以重新下载每一帧画面，可以只请求丢失的帧，也可以略过它们。很显然，第一个选择不是最好的，但在后面两个选择中作决定也很困难。

在最基本的级别上，Web 作出了最后一个选择：它忽略了它所丢失的东西。这是一个有趣的选择，并且在面对它时，看起来像是一个错误的选择。但其他选择将需要在基础结构上进行巨大的投资。显然，有时候你需要一个持久稳固的基础结构，但并非总是如此。

想像一下，如果你的浏览器需要确认它所下载的每帧图像，这将会发生多少次通信，而服务器将一直忙碌，直到你接收到图像为止。另一个例子，考虑普遍存在的 404 错误代码：在 HTTP 规范中，这表示服务器可能丢失了一个文件。

Web 还具有高度的并发性。这是到目前为止计算机中最具并发性的系统。每个 Web 服务器和每个客户端将并行地操作。但对于并发性的编程是很困难的。这可能是对现代 Web 开发的一个讽刺。准确地说系统自身对于开发也是难于管理的。最本质的挑战就是，以一种高效的考虑失败原则的方式来处理状态。

Web Services 是一些有助于开发分布式应用程序的技术，它同时考虑到了失败和并发性。在详细了解 Web Services 如何做到这些之前，让我们花点时间定义一下什么是 Web Service 吧！

## 定义 Web Services

Web Services 可能是最难定义的术语之一了。大多数定义在某种程度上都是不完全的，它们可能只是我们在一般意义上使用这个术语的一种指示而已。如果我们以什么不是 Web Service 开始，事情可能要容易一些。Web Services 不是人类可读的 Web 站点——至少在本书中使用这个术语时不是这样。Web Services 并不是任何终端用户（即使是非常熟练的用户）可以直接与之交互的技术。Web Services 是一种跨机器进程所使用的技术。

## Internet 标准的意义

Web Services 很难定义，但有一种严肃的说法：Web Service 是通过 Internet 标准可访问的应用逻辑。从一台机器向另一台机器获取数据的观念——也就是建立分布式应用程序——已经有几十年的历史了。曾经有许多技术用于公开应用逻辑，但这些技术通常都基于一些很难实现并且特有的协议。某些标准，特别是像 XML 这样的 Internet 标准，已经简化了分布式应用程序的构建。

**什么造就了标准？**

有些人可能会指出 SOAP 不是标准，因为它（在编写本书时）还不是 W3C（World Wide Web Consortium，万维网联盟）建议的标准。但我认为，标准可能在标准组织之外产生，这也是合乎情理的。像 SOAP 之类的标准，由于它已经跨许多平台和语言而被实现，所以已经成为一种事实标准。SOAP 很容易满足这种鉴定性测试，因为在我能想到的每种语言、我所听到的每种操作系统以及其他我未曾听说的系统中，它有超过 50 种实现！并且它是基于 W3C 的推荐标准的，例如 XML 和 HTTP。

### WSDL 的意义

关于 Web Services 的定义，另一种半正规的说法就是：它就是 WSDL（Web Services Description Language，Web Services 描述语言）文档所能描述的任何东西。尽管事实如此，但这个定义还是不如前一个定义好，但当我说“Web Services”时，这是在我的脑海里所呈现的定义中与前一个定义最接近的一个。而当我谈到 Web Services 客户端时，我考虑的是通过 WSDL 文档以学会使用 Web Services 通信的技术。当然你可以通过 WSDL 构建一个什么也不做的服务器或客户端，但至少 WSDL 的概念是存在的。

### 互操作的意义

那么为什么 Web Services 技术会存在？曾经有许多技术用于访问和公开应用程序逻辑，但这些技术通常都有严重的缺点。Web Services 试图通过提供不同的、具有互操作性的、松耦合以及独立实现的程序，来解决这些问题。我想要说的是，大多数 Web Services 技术的两个最大的设计目标就是互操作性和松耦合。

互操作性通常是人们对 Web Services 产生兴趣的原因。互操作性的意义是很容易理解的。例如，Web Services 可以让一个 Perl 开发人员与微软的 Windows 2000 系统上的 C++开发人员进行互操作，而后者又能与 Macintosh OSX 系统上的 AppleScript 程序员进行互操作。我确实见到过这种情形，不只是演示而是在实际部署的程序当中，可以让商业应用比过去更容易集成。这是因为它们可以与其合作者协同工作，而不必采用与其相同的平台。表 1.1 列出了当前可行的一些 Web Service 技术示例。

表 1.1 流行的 Web Service 技术

| 产品 | 支持的操作系统 | 语言 |
| --- | --- | --- |
| Apache SOAP | UNIX, Windows | Java |
| WASP | UNIX, Windows | C++ |
| GLUE | UNIX, Windows | Java |
| SOAP BEA WebLogic | UNIX, Windows | Java |

此为试读，需要完整PDF请访问：www.ertongbook.com