
内附CD-RO
C++Builder 6
SOAP/Web Service
开发
李维
华中科技大学出版社

C++ Builder 6
SOAP/Web Service 开发

李 维

华中科技大学出版社
中国·武汉

图书在版编目(CIP)数据

C++Builder 6 SOAP/Web Service 开发/李　维
武汉:华中科技大学出版社，2002 年 8 月
ISBN 7-5609-2749-1

Ⅰ.C…
Ⅱ.李…
Ⅲ.程序设计-C++
Ⅳ.TP31

C++Builder 6 SOAP/Web Service 开发　　李　维

策　　划:周　筠(http://yeka.xilubbs.com　junzhou@public.wh.hb.cn)　　责任校对:陈元玉
技术编辑:任颂华(TR@SOE)　　责任监印:张正林

出版发行:华中科技大学出版社　　武昌喻家山　邮编:430074　电话:(027)87545012

录　　排:华中科技大学惠友科技文印中心
印　　刷:湖北新华印务有限公司

开本:787×1092　1/16　　印张:25.75　　插页:2　　字数:400 000
版次:2002 年 8 月第 1 版　　印次:2002 年 8 月第 1 次印刷　　印数:1—6 000
ISBN 7-5609-2749-1/TP・472　　定价:58.00 元(含 1CD)

分布与集成的艺术

——《C++Builder 6 SOAP/Web Service 开发》引介

再没有一个地方比 Internet/Intranet 更具分布性：跨地区、跨平台、跨应用……

也再没有一个地方比 Internet/Intranet 更需要集成：地区间、平台间、应用间……

这乃是当今每个系统/应用程序开发者所面临的现实问题。也许正应了中国的一句古话，所谓“合久必分，分久必合”。从最初的 VAX/DEC（合），到 PC 革命（分），再到 Intranet/Internet（合）；从单机应用（分），到网络分布应用（分），再到现在追求的应用间的集成（合），不正验证了这一潮流？

技术的发展永远是和现实需要相配套的。我们需要用一种新的概念和技术来解决当今极为分布的环境下，各应用之间互相集成的问题。幸运的是，这样的概念和技术已经出现了，它就是 SOAP（Simple Object Access Protocol，暂译为“简化对象存取的协议[1]”）和 Web Service。

李维是台湾著名的 IT 资深技术专家，在 Borland 技术圈享有国际性声望。李维先生对最前沿的技术向来有着敏锐和深刻的理解，由他来为我们描绘这一当今最先进、最热门技术的发展前景最合适不过了。《C++Builder 6 SOAP/Web Service 开发》是李维先生的最新著作。在这本书里，他详细地介绍了 SOAP/Web Service 技术。

本书一共有十五章。大致分为以下两个部分：

第一部分，概念性的介绍。

---

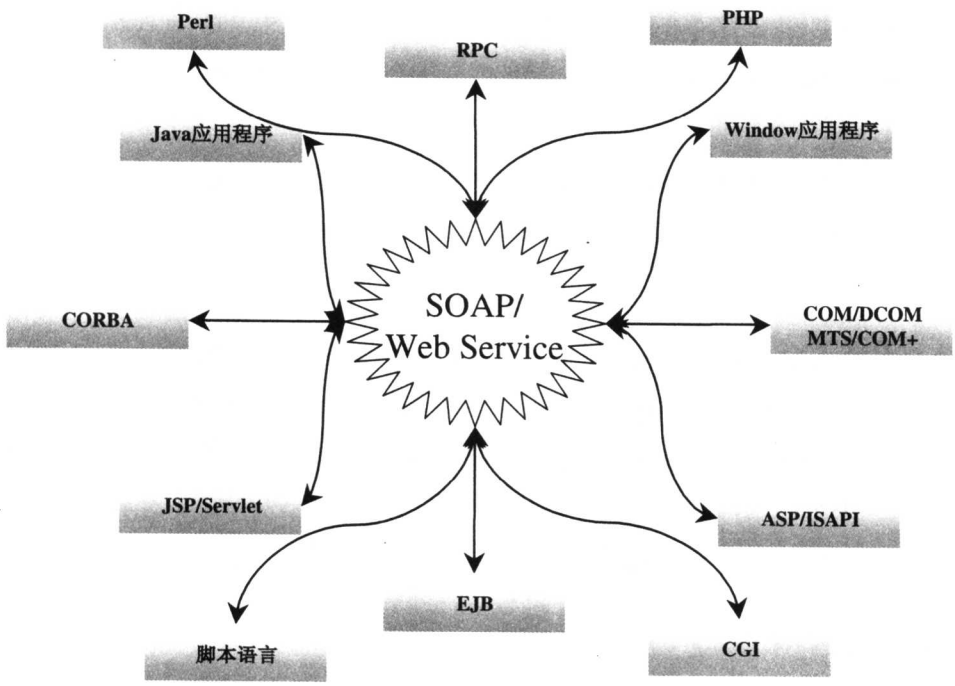
[1] SOAP 中的 Simple 很难翻译。也许使用 Delphi/BCB 的程序员会觉得开发 SOAP/Web Service 应用确实很“简单”，但这是 Borland 将 SOAP 优雅地封装而带来的结果。SOAP 本身并不“简单”。所以，我偏向于将 Simple 翻译为“简化”。我个人认为，将其理解为“（能够）简化对象存取（的过程和开发）的协议”比较合适。

第二部分，实例开发。

第 1 章“SOAP 和 Web Service 的概念”，描述了 SOAP/Web Service 的源起，属于第一部分。重点阐明了由于 Internet/Intranet 是一个多元化的计算环境，是由各种不同的实现技术、平台和操作系统共同结合而成的。因此，必须使用大家都接受的标准，来集成彼此的服务。SOAP 和 Web Service 就是在这种需求下产生的标准，而且迅速成为大家都接受的开放标准。

第 2 章“组件模型，Internet/Intranet 和 SOAP”，从描述当今各个流行组件模型（CORBA，COM/DCOM，JAVA RMI 以及 EJB）的基本运作模式和 BRIDGE 技术的限制出发，自然而又明确地表明：第一，SOAP 并不能替代现有的组件模型；第二，SOAP 更像是一位“万国通译员”，通过它，各组件模型下开发的应用才能够无缝而自然地交互和集成。不过，这位通译员的特色是它本身并不精通多国语言，而是通过要求各个系统讲述同一种语言——SOAP 来达到通译的效果。:=)

第 2 章里有一张图，向我们展示了程序员的“理想国”：



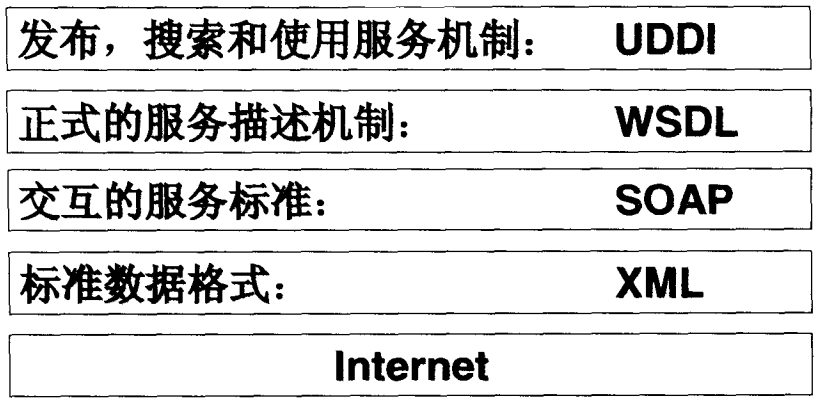
这里已经不再有各种语言、各种组件、各种应用之间的藩篱，也不再需要复杂而专用的 BRIDGE 技术，不再有不同数据包和协议之间“鸡同鸭讲”的困惑。一切都通过 SOAP 和 Web Service 这个标准来互相沟通。

第 3 章“开发 Web Service 应用系统”，暂时离开了概念和理论部分而进入第二部分。本章讲述了 BCB 中有关 Web Service 的控件组及其应用。它从若干个最简单的例子出发，介绍了开发 Web Service 程序服务器端和客户端的步骤（以及各个 Wizard 的选项和使用），也顺便介绍了 WAD（Web App Debugger）调试、WSDL 的管理、Web Service 应用的不同类型和简单的数据库开发。这一章内容非常丰富，而且有许多重要的、实践性的东西在后面的章节没有再被提到。所以读者不应该轻易放过这一章，应该予以重点阅读。

第 4 章“什么是 SOAP？”、第 5 章“SOAP 和数据封装”、第 6 章“SOAP 和远程调用”，又回到了概念部分。它从 SOAP 的产生谈起，谈到了 SOAP 的定义、目标、功能规范等，最后分析了 SOAP 的优缺点（第 4 章）；谈到了 SOAP 中的数据类型（简单类型和复合类型），数据封装的规则，以及 BCB 中对某些特定类型的转换支持类（第 5 章）；远程调用的 Serialization 和 De-Serialization，对象/接口引用等（第 6 章）。我们可以从中清晰地看到一个标准如何从最初的构想开始，经过众多个人、组织的协同努力，逐渐地完善、丰富，最后形成一个国际化的标准。标准的形成是以坚实的应用为基础的，这从一个侧面暴露了我们目前的应用层次还不高，因此，不能为这些标准的形成做多大的贡献。

第 7 章“Web Service 和 UDDI”，则讲述了 Web Service 和 WSDL/UDDI 等概念。如果说 SOAP 是一个交互的标准，那么 Web Service 就是“建立在 SOAP 之上而提供应用程序和应用系统之间相互沟通的能力”的“服务”。UDDI（Universal Description Discovery and Integration）则提供了一个公开的标准，它是以结构化的方式来注册、管理企业信息以及相关的服务信息。通过 UDDI，软件人员可以发布或是搜寻企业以及 Web Service 信息，从而达到资源共享和集成的目的。

上述四章是对 SOAP/Web Service 概念的全面回顾和总结，读者应该认真体会这些概念，并自己开发一到两个 Web Service，以期更好地领悟 SOAP/Web Service 的内涵。以下这张图是我们在阅读完上述四章后，应该牢牢记在脑海中的：



从第 8 章开始，本书全面进入第二部分。李维开始展示他深厚的技术功力，通过丰富的例子向我们说明前面介绍的这些概念的应用。

第 8 章“处理复杂数据类型的 Web Service 应用系统”，先用三个例子分别说明了如何使用动态数组、Base 64 Encoding/Decoding 和 MIME Encoding/Decoding 处理 BLOB 类型的数据，实现一个 Web Service AVI 播放器、一个 Web Service 图形处理系统和一个 Web Service WAV 文件播放器；接着又讲述了如何处理记录类型的复杂数据，其中要牵涉到 TRemotable 和 TRemotableXS 这两个重要的类。虽然 BCB 已经帮助我们完成了大部分常见 BCB 类型的数据类型和类在这两个类中的定义，但是我们应该充分掌握如何创建自己的 TRemotable(XS)派生类来处理自己的数据类型。

第 9 章“使用 MS SOAP Toolkit 开发 Web Service”，在 BCB 本身开发 Web Service 的机制上介绍了另一种开发 Web Service 应用程序的方式：利用 Microsoft 提供的 MS SOAP Toolkit（简称为 MSST）。MSST 采用的 WSDL 绑定方式与 BCB 6 中采用的方式截然不同。前者采用的是动态绑定而 BCB 6（包括 Delphi 6）采用的是静态绑定。读者应该了解这两者的区别，并且要掌握 MSST 的开发方式，因为毕竟 MSST 对由微软.NET 技术实现的 Web Service 的支持是最好的。

第 10 章“Web Service 和数据库应用系统”，以数据库应用开发作为介绍对象，向我们介绍了如何构造服务器端（SOAP Server Data Module 及其构成），如何构造客户端（TSoapConnection 控件和其它），并顺便介绍了 BLOB 数据的“透明”处理(即不再需要我们显式地调用 Base64 Encoding/Decoding 等函数了）。这一章介绍的数据库应用模式还是 C/S 模式的。读者可以将本章的应用作为一个很好的起点来构思自己的数据库应用系统。

第 11 章“开发分布式 Web Service 应用系统”，是上一章的延伸，讲述了最具魅力的分布式应用系统。然而更重要的是本章给我们指出了一种以最小代价将现有的分布式系统升级到 Web Service 应用系统的方向，并演示了实现的方法。读者所在的企业内部或其上下级供应链中一定已经存在异构平台上的很多应用，也一定会面临如何将其集成的问题。于是关键的关键就是如何在不更改原始应用的情况下，将这些应用升级到 Web Service-Ready？本章从构造一个小型的数据库应用出发，以 COM+作为后台操作的实现技术，用 Web Service 作为中间层来封装和公布 COM+的服务接口，最后通过前台 WAD 类型/ISAPI 类型的客户端程序来调用 Web Service，进而进行分布式数据库的操作。

以此作为出发点，我们可以举一反三，对于现有的、以不同分布技术（CORBA/EJB/COM+）实现的应用，采用类似的封装形式包装成为 Web Service，从而一举消除了异构平台间应用集成的问题。

这一章的例子虽然简单，但其中的思路却十分新颖实用，给我们带来的启示是巨大的。读者应该认真阅读并加以思考：如何能将这一思路应用到目前自己企业的实际中去。

第 12 章“Web Service 和执行效率”，集中讨论了如何提高 Web Service 应用的执行效率并提出了若干很实用的建议。这里提到的很多技术对于其它的应用也同样适宜，只是我们在一开始开发 Web Service 应用时可能会忘记在 Web Service 应用系统中使用这些技术，或是不知道如何在 Web Service 的开发中搭配这些方法。

第 13 章“C++Builder 6 中 SOAP 和 Web Service 的幕后制作”，是全书中非常特殊的一个章节。李维尝试从 Borland 公司 Rick Nadler（Delphi 6 中 SOAP/Web Service 功能开发负责人）和 Jean-Marie Babet（BCB 6 中 SOAP/Web Service 功能开发负责人）的身份和角度出发，向我们讲述 Rick 如何在 D6 中从 SOAP/Web Service 规范出发，逐步形成实际的概念，然后再形成各种实际的结构，深入到编程的细节，解决了四大关键技术（处理 XML 之技术、封装包传递技术、Object Pascal 和 SOAP 包之间的相互转换技术和 HTTP 的请求处理技术）的实现，从而完成了在 D6 中加入 SOAP/Web Service 的任务。

接着，Babet 在 D6 的基础上，为了让 BCB 的程序员能够以最小的代价来顺利地使用 SOAP/Web Service，为了让 BCB 6 中 SOAP/Web Service 实现为“First Class”[1]，使用了若干巧妙的技术来完成 C/C++和 Object Pascal 之间的无缝桥接。这些巧妙的技术牵涉到了使用深入的 C/C++语言技巧，编译器的支持

[1] “First Class 功能的意思是指 Babet 需要让 C/C++的程序员觉得 C++Builder 的 SOAP/Web Service 核心功能就像是使用 C/C++直接实现的一样，而不是使用 Object Pascal 实现的。”以上摘自书中原文。

以及 Smart Interface 的机制，以便让 C/C++程序员在负担最小的情形下方便地开发 SOAP/Web Service 应用程序。

这一章是很有趣也很优雅的一章，也必定是非常令人愉快的一章。循着大师们的思路，看着大师们的代码，难道不会让你有“茅塞顿开”或“于我心有戚戚焉”的感觉吗？而这些感觉不正是令人陶醉的吗？

从这一章中，我们也可以看到为什么 BCB 的开发总是要落后 Delphi 一段时间，但却总是要比 Delphi 更完善、更先进一点。

第 14 章“到 Internet 上使用 Web Service”，向我们介绍了如何通过自己开发客户端来使用目前可以在 Internet 上找到的现成的 Web Service 服务器。在本章的三个例子中，分别调用了由.NET、Delphi、GLUE 开发的 Web Service 服务器。我们可以再次感受到 SOAP/Web Service 的威力和通用性。其中，第三个例子所调用的服务器更是十分复杂。读者应该在阅读完本章后尝试用 BCB 6 来开发一些客户端程序，去调用不同的服务器，以提高自己的 SOAP/Web Service 功力。

第 15 章“建立和部署 Web Service 应用系统”，介绍了 SOAP/Web Service 应用系统的分发和部署。与 CORBA/DCOM 等应用系统的客户端分发不同，SOAP/Web Service 应用系统基本上不需要什么客户端的分发，一个 EXE 文件已经足够，也不需要进行什么复杂的授权处理、DLL 注册等步骤。所以，建立和部署 Web Service 应用系统的重点是建立 UDDI Registry。通过 UDDI Registry，我们可以更好地管理 Web Service，也能够让用户搜索并使用我们的服务。UDDI Registry 彻底解决了 Web Service/WSDL 中点对点的对应关系所带来的问题：一旦 Web Service 数量增加到一定程度，那么管理 Web Service/WSDL 就成为一种负担。

通过本章，读者也能够学习到 Microsoft UDDI SDK 的基本概念和使用方法。

这十五章的内容涵盖了很多东西，但是正如李维在后记中所言：“还没有经历过在写完一本书之后还似乎有种尚未完成的感觉。”SOAP/Web Service 实在太新，发展实在太快。任何一本新书出版之时，必定有一些地方和最新的发展已经脱节。所以，读者不应该满足于只阅读书籍，如果有兴趣的话，应该去一些站点、新闻组多看看，从而掌握最新的潮流和发展动态。或许，你也有能力为 SOAP 的发展做一些贡献。

SOAP/Web Service 给我们带来的前景是无比美妙的，我们可以开发出 Service Provider 去综合目前的 Content Provider 而为我们的用户提供新的服务；也可以开发出 Content Provider 来基于 Service Provider

而提供综合性的内容；更有 Service Designer 正蓬勃而起，让我们结合 Service Provider 和 Content Provider 来设计属于我们自己的全新的 Service。

SOAP/Web Service 的出现给我们带来了全新的应用开发概念，这已经不是单纯意义上的技术层次的提高，而是整个概念层次的重组。其重要性，大概只有 RDBMS 的出现可以与之相提并论。

它的出现，还必将给整个软件产业带来巨大的冲击（或者不如说是契机）。开发新的服务，提供新的内容，都可能成为软件产业新的增长点。而在整个 Web Service 的应用环节中，也还有很多地方可以得到改进和提高。

例如、文章中提到的如何提高执行效率方面的问题中，就提到了缓存机制；而服务器端对传递来的 SOAP 包进行 XML 解析也是非常耗时的工作，是不是会出现一个类似 JVM 的机制(XVM？)来提高 XML 的解析和执行速度？这些都是可能成为新的话题和发展方向的地方。

综观全书，内容翔实，文笔流畅，看得出李维先生写作时文思澎湃；对概念的描述严谨而不生硬死板，加之以生动的实例和图表，能帮助我们更好地掌握概念。《C++Builder 6 SOAP/Web Service 开发》这本书再一次很好地展现了李维先生深厚的技术功底和大家风范。

TR@SOE

2002 年 6 月于苏州

序

科技的进步真是非常迅速。从 1999 年推出 C++Builder 5 之后，软件技术在不断地蓬勃演进。Web 应用已经成为主流，而多层结构也逐渐被许多系统所采用，特别是结合 Web 应用和分布式结构的应用系统早已悄悄地出现在你我的日常生活中。想想数年前 C++Builder 3 第一次以多层结构作为发展的主轴，到现在不过数年的时间，分布式应用系统已经成为事实，而且是愈来愈多应用系统使用的主流技术。我们不禁要佩服那些 C++Builder 的研发人员的眼光了，特别是所有 C++Builder/Delphi 程序员家喻户晓的 Anders Hejlsberg，以及当初坚持 C++Builder 3 要加入分布式功能的 Zack Urlocker。笔者很庆幸恭逢其时，相信许多读者也历经了这场丰富的信息科技革命的洗礼。

C++Builder 6 提供的新功能是延续 C++Builder 5 的自然发展，并且融合了许多目前最重要的软件技术，让 C++Builder 的开发人员能够及时地使用 C++Builder 6 开发现在和未来的应用系统。这些重要的软件技术包括：WebSnap，C++Builder 6 新一代的 Web 开发技术；DataSnap，Midas 的最新版本，加入了跨平台以及 XML 支持的功能；DataExpress，Borland 最新、高效率、跨平台的数据库存取引擎，可结合 DataSnap 开发多层应用系统；以及本书讨论的重点，SOAP/Web Service 技术。这些新的软件技术都非常实用，足以提供 C++Builder 软件人员开发主从结构、Web 和分布式多层应用系统。当然，这些技术也都足以写成专著，详细说明这些技术如何运用。

本书是专门讨论 SOAP/Web Service 技术的实用性书籍，因为笔者认为 SOAP/Web Service 将会是现在和未来最重要的软件技术和发展趋势，这可以由目前所有的开发工具和应用软件技术看得出来。不但 Java 将把 SOAP/Web Service 定义成核心，Microsoft 的.NET 也是将 SOAP/Web Service 作为核心技术。C++Builder 6/Delphi 6 不但是第一个完整支持 SOAP/Web Service 技术的开发工具，而且 C++Builder 还在不断改善 SOAP/Web Service 方面的功能，让它们更强大，也确保 C++Builder 6 的 SOAP/Web Service 技术能够顺利地和所有使用其它开发工具发展的 SOAP/Web Service 应用系统相互沟通。此外，C++Builder 6 和 Delphi 6/Kylix 2 将拥有相同的 SOAP/Web Service 技术核心，因此，Borland 也提供了一个跨平台的 SOAP/Web Service 技术结构。

由于 SOAP/Web Service 的重要性，因此，笔者认为应该写一本完整的书籍来介绍它们，而不是以一个简单的章节来带过。在本书一开始的两个章节中将会说明为什么 SOAP/Web Service 技术会被提出并且得到快速地发展，也会比较 SOAP/Web Service 和现在使用的各种控件模型以及通信协议，讨论为什么 SOAP/Web Service 可以解决以往无法轻易达成的事情。接着本书会使用 C++Builder 6 来实际开发 SOAP/Web Service 应用系统，让读者能够先具备实际的开发能力。在第 4、5、6 章中本书将以实际范例来介绍 SOAP 的功能规格，让读者能够切实了解什么是 SOAP、SOAP 设计的概念以及 SOAP 的技术细节。

从第 7 章开始，本书将进入高级的 SOAP/Web Service 技术，开始讨论各种 SOAP/Web Service 的应用技术和结构。例如，如何在 SOAP/Web Service 应用系统中处理复杂的数据类型，以及如何开发结合数据库的 Web Service 应用系统，说明如何使用 SOAP 追踪工具，以及如何结合 SOAP/Web Service 和 COM+，开发分布式 SOAP/Web Service 应用系统。本书也会讨论如何调整 SOAP/Web Service 应用系统的执行效率，让读者不但能够使用 C++Builder 6 开发 SOAP/Web Service 应用系统，还能够让应用系统执行得非常有效率。最后本书带领各位到 Internet/Intranet 上使用 C++Builder 6，调用由其它开发工具发展的 SOAP/Web Service 应用系统，让读者真正领略 SOAP/Web Service 的威力，了解 SOAP/Web Service 提供的强劲集成能力。相信读者阅读完本书之后，一定能够切实掌握 SOAP/Web Service 技术，准备迎接下一波的技术挑战。

本书与笔者另一本著作《实战 Delphi 6/Kylix 2-SOAP/Web Service 程序设计篇》（机械工业出版社出版，2002.4）不同的地方是：在 Delphi 的书中并没有讨论到 UDDI 程序设计，以及如何分发 Web Service 应用系统，而这些重要的技术问题都已在本书之中进行充分的讨论。此外，在结合 Web Service 和 COM+ 的技术方面笔者也使用了 ATL 来说明，当然，所有的范例程序都是使用 C/C++撰写的，同时使用了 C++Builder 中 Smart Interface 的机制来开发 SOAP/Web Service。

使用 C++Builder 一直是令人非常高兴和舒服的事情，因为不但可以使用 C++Builder 开发各种应用系统，也能够不断学习最新的软件技术，充实个人的价值。在 Visual Basic 停止开发第 7 版，而改以 VB.NET 来代替，PowerBuilder 的发展速度也愈来愈缓慢的时候，C++Builder 仍然在不断地快速进步。它是现在最佳的 Windows 原生开发工具和即将推出的 C++Builder For Linux 一样，将同是最好的 C/C++ RAD 工具。未来 Borland 将会持续发展.NET 下的 C++Builder，继续为 C++Builder 的软件开发人员提供最好的可视化开发工具。

最后还是要谢谢许多关心笔者的读者，这么多年来不断地鼓励和支持我写的书籍，也特别要谢谢周筠编辑，若没有她的努力和坚持，这本书是绝对不可能出版的。希望这本书也真的能够帮助那些想要了解 SOAP/Web Service 新技术的读者，顺利地进入新世纪的应用系统开发环境。

李维

2002 年 4 月于新店

目 录

1

SOAP和Web Service的概念

记得是在四五年前，我第一次在 C++ Report 中看到了一篇令我耳目一新的文章，这篇简短文章的内容是讨论如何以 C/C++调用在远程机器中的程序代码。虽然当时已经有 RPC 和 CORBA 等技术，但是这些技术在那时还属于特定的平台，都太复杂，并且受限于特定的语言、开发工具以及操作系统，因此，要使用这些技术开发分布式应用系统不是简单的事情。

而那篇 C/C++的文章就在讨论是不是有一种机制，能够让调用远程程序代码的工作不受制于任何的操作系统、平台和开发工具，并且能够以标准的方式来进行这个工作。当时这位作者就提出了类似如下的语法，允许客户端的 C/C++应用程序使用这个文字描述来调用远程一个称为 CalculateInterest 的函数来计算利息：

```
<Method>CalculateInterest
    <Parameter>10000</Parameter>
    <Parameter>0.05</Parameter>
    <Result>Float</Result>
</Method>
```

相信上面的语法非常容易理解，它的意思是调用远程 CalculateInterest 函数，并且传递两个参数，第一个参数的数值是 10000，而第二个参数则是利率 0.05，而这个函数则希望返回一个结果类型为 Float 的返回值。

这个构想是以特定的文字格式封装对于远程函数的调用，再由某一种机制来真正的调用远程的程序代码，最后返回函数结果。至于远程的 CalculateInterest 函数是在什么地方执行？是哪一个机器和操作系统？以及使用什么开发工具实现的？这些问题客户端并不需要知道，客户端只是希望能够使用 CalculateInterest 函数提供的服务来完成客户端应用程序的工作。

这是一个多么令人心动的想法，这个想法在数年之后的今天终于由本书稍后讨论的技术 SOAP 和 Web Service 提供了初步的实现。而 SOAP 和 Web Service 仍然在不断地快速发展之中，相关的规格和技

术也正如火如荼地在研发之中。相信这个构想会很快地被实现并且得到改善。在继续讨论之前，也许让我们回顾一下最近几年软件技术的发展就可以了解为何这个构想会在不知不觉之中逐渐地成为现在最重要的软件发展趋势。

1-1 Internet/Intranet 和开发模式的演变

从两三年前一直到现在影响全世界最重要的技术就要算是 Internet/Intranet 了。Internet/Intranet 不但改变了许多人的生活，也影响了世界运转的方式。各种商机和软/硬件的发展也由 Internet/Intranet 主导。Internet/Intranet 的出现也为软件的开发方式带来了多元的形式，软件人员开始使用各种工具来开发 Internet/Intranet 应用系统。第一波重要的技术是 HTTP/HTML，它为 Internet/Intranet 和电子商务都带来了重要的影响。第二波是 Java 语言的兴起，Java 带来了跨平台的契机，让软件人员可以使用 Java 在各种平台上使用单一的语言和环境开发应用系统，Java 几乎提供了跨平台的机会。第三波则是 XML 技术的兴起，它提供了标准的数据封装技术，让数据的交换跨越了各种平台、操作系统和开发工具。通过 XML，各种应用程序在交换数据时再也不会令人头痛了。

虽然 Java 和 XML 在应用程序和数据交换方面几乎提供了完整的解决方案，但是在 Internet/Intranet 多元化的时代中，Java 也无法提供所有的解决方案。许多的 Internet/Intranet 应用系统仍然使用了其它的技术（例如 ASP、PHP 等技术）来实现，当然也包含了 Delphi，C/C++等。即使是在 Internet/Intranet 的世界中，许多的 Internet/Intranet 应用系统也无法相互沟通和集成。例如 ASP 无法直接调用 Servlet 或是 JSP，而专为 Java 提供的 EJB 控件模型也无法由 ASP 轻易的调用，即使我们仍然可以使用 XML 技术在这些不同的实现技术之间交换数据。实现技术和应用程序之间无法互相的沟通和集成在 Microsoft 的.NET 正式发表之后，情形会更为严重，因为根据许多的调查结果都显示未来 Java 和.NET 将平分天下，各领风骚。但是在 Internet/Intranet 世界中，我们仍然急需能够解决这个问题的技术。

造成这种现象的原因是因为在这些 Internet/Intranet 应用系统中，仍然是以特定的软件技术为中心，以 XML 技术交换数据为辅。因此在下一波的 Internet/Intranet 技术演变中，势必将以突破不同实现技术为重点，让各种实现技术也能够像 XML 一样能够轻易地集成。不管各种 Internet/Intranet 应用系统的实现技术为何，我们需要的是一种标准接口，能够让它们彼此可以调用和使用对方提供的功能或是**服务**，这就是 Web Service 兴起的概念。