Internet 新技术丛书

HZ Books

SOAP

XML跨平台 Web Service开发技术

SOAP: Cross Platform Web Service Development Using XML

与平台无关、与开发工具无关，是不是您一直追逐的梦想？

附赠 CD-ROM

(美) Scott Seely 著

杨涛 杨晓云 王建桥 高文雅 等译

机械工业出版社 China Machine Press

PH PTR

Pearson Education

Internet新技术丛书

SOAP：XML跨平台Web Service开发技术

SOAP: Cross Platform Internet Development Using XML

（美） Scott Seely 著

杨 涛 杨晓云 王建桥 高文雅 等译

机械工业出版社
China Machine Press
PH PTR
Pearson Education
培生教育出版集团

SOAP作为一种综合各种计算机技术的通信协议，能够用多种程序设计语言在多种操作系统下和多种计算机平台上实现。本书介绍了使用SOAP所必需的XML相关内容，集中讨论了SOAP的技术标准、一个简单的SOAP客户和服务器，并用大量篇幅完整地介绍了一个运行在UNIX和Windows操作系统上的大型SOAP应用程序的开发和实现过程。

本书内容翔实、实例深刻细致，许多章节附有练习题，可帮助读者更好地掌握相关论题的知识。随书所附光盘包括所有的资源代码，以及一个完整的网上拍卖系统的实现程序清单。

本书版权登记号：图字：01-2001-4770

图书在版编目（CIP）数据

SOAP: XML跨平台Web Service开发技术/（美）塞利（Seely, S.）著；杨涛等译. -北京：机械工业出版社, 2002.4

（Internet新技术丛书）

书名原文：SOAP: Cross Platform Web Service Development Using XML

ISBN 7-111-09517-0

Ⅰ. S… Ⅱ. ① 塞… ② 杨… Ⅲ. ① 无线电通信-通信协议，SOAP ② 可扩充语言，XML-程序设计 Ⅳ. ① TN915.04 ② TP312

中国版本图书馆CIP数据核字（2002）第008282号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：张鸿斌

北京第二外国语学院印刷厂印刷·新华书店北京发行所发行

2002年 4 月第 1 版第 1 次印刷

787mm × 1092mm 1/16 · 19印张

印数：0 001-4 000册

定价：45.00元 (附光盘)

序　言

自从有了两台计算机，如何让它们进行通信就成为人们研究的课题。针对这一问题，人们已经提出了几十种，甚至上百种解决方案，每一种都有它自己的长处和短处。但如何让两台计算机彼此之间的通信策略达成一致，仍然是令人头疼的问题。每个人都想让对方来适应自己的策略需求，因此形成了所谓的“通信战争”：CORBA对DCOM、DCOM对RMI、信息方案对RPC方案，等等。

SOAP（Simple Object Access Protocol，简单对象访问协议）就是在这种混乱的“通信战争”状况中诞生的。SOAP并没有试图去解决所有的问题，它只是定义了一个简单的基于XML的通信格式。但就是这个简单的目标，再加上其强大的扩展机制，SOAP实现了人们一直追求的目标，成为一种综合各种计算机技术的通信协议——它能够用多种程序设计语言在多种操作系统下和多种计算机平台上实现。不论是计算机、操作系统还是程序设计语言，只要它能够生成和处理XML（XML本身只不过是一种纯文本），就能用上SOAP。自从出现了SOAP，几乎所有主要的软件开发商都推出或者发布了与之相关的实现产品。我们已经能够见到许多种与SOAP有关的产品了，其中包括独立的SOAP软件、内建有SOAP功能的Web服务器/应用服务器/通信工具，甚至能够见到使用了SOAP技术的网络消息中间件（middleware）。目前，微软公司、IBM公司、Apache公司、Sun公司等重量级企业都在努力往它们的应用软件、操作系统、程序设计语言产品里添加更多的SOAP支持，因此我们可以有把握地断定，SOAP很快会成为未来的主流。

随着W3标准化进程的发展，SOAP的技术标准也会不断地得到补充和完善，计算机和网络技术方面的变化和发展每天都会发生。但这不应该成为我们不在自己的应用软件里尝试和使用SOAP技术的理由。事物肯定会变化，标准也会被修改，但这将是一个循序渐进的由量变到质变的过程，每一次的改动都不会很大，而相应的SOAP产品会把有关的细节都隐藏在其内部，不会影响我们的实际应用。

我最早是在专门讨论SOAP技术的DevelopMentor的邮件表（如果读者有兴趣加入这个讨论，请访问http：//discuss.develop.com /soap.html站点）上“遇见”本书作者Scott的。他在这个邮件表上不知疲倦地帮助其他人了解和学习SOAP技术——很明显，他认为这是一种非常重要的技术。因此，当我得知他开始编写这样一本书的时候非常高兴。在这本书里，他为我们大家提供了大量的实用开发技巧。也正是他，告诉我目前已经有了那么多采用了SOAP技术的解决方案，而这些解决方案的实际通信效果都非常不错。

Scott和我都认为SOAP将成为一种重要的技术，我希望大家在阅读和学习这本书的过程中能够体会到这一点。你可能是一位使用着Apache公司SOAP产品的Java程序员，可能是一位使用着微软公司SOAP Toolkit（SOAP工具包）的程序员，还可能是一位使用.NET Web Services网络服务或者其他开发工具的C#程序员，我希望通过这本书的学习，你能加入到我

们的行列里来，在自己的应用软件里用上SOAP技术。让我们一起前进吧。

——Kent Sharkey
微软公司“.NET”网络技术专栏作家

参加本书翻译的还有：张玉亭、韩兰、李京山等。

目　录

第二部分 相关技术

第三部分 案例研究：网上拍卖系统

附 录

第一篇

Jianqiaomishuzhengshukaoshizhidingjiaocai

剑桥秘书证书考试指定教材

有效使用打字机或字处理机

文字处理

位。图1-2中的算盘代表的是数字23。

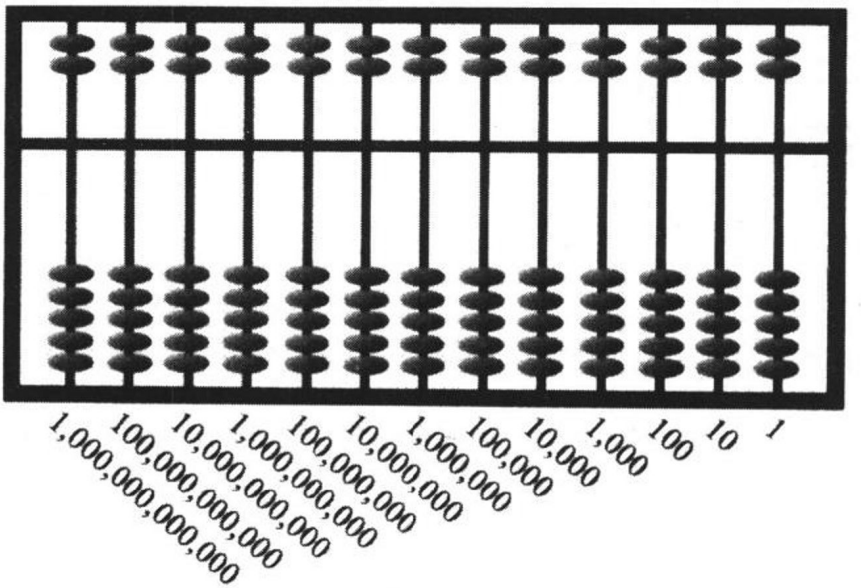


图1-1　一个有13根算盘轴的算盘

算盘特别适合用来完成数字的加减法。算盘打得好的人往往会胜过使用计算器的人。在用算盘做加减法的时候，人们通过把算盘珠拨上拨下来完成计算：如果横挡下方的5个珠子都拨到横挡上去了，就要把同一根算盘轴上横挡上方的一个珠子拨下来，而那5个珠子就要被拨回到算盘底部；同样，如果算盘横挡上方的两个珠子都拨到横挡上去了，就要把它们拨回到算盘顶部，同时要把这根算盘轴左侧那根算盘轴上的一个珠子拨到横挡上去。图1-3给出了用算盘完成“7 + 23”的计算工作，最后的结果是30。如果是进行30减7的计算，把我们刚才说的过程反过来就行了。

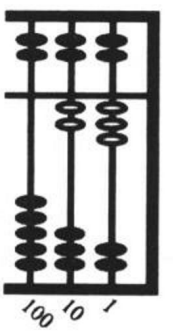
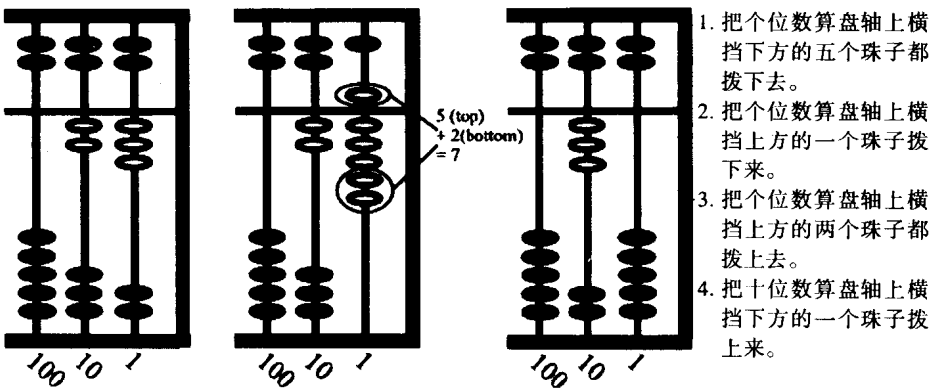


图1-2　用算盘表示的数字23（白色珠子）

你弄明白了吗？我们换一种方法来解释一下算盘的工作原理。例如用人的手指来模拟一个人体算盘，普通算盘也是基于同样的道理。算盘横挡下方的5个算盘珠可以用人一只手的5个手指来代替；横挡上方的两个算盘珠表示人的两只手，每只手有5个手指。我们的人体算盘的用法是这样的：如果一只手的5个手指都用完了，我们就开始用另外一只手来计数。如果代表个位数的那个人数到了10，这个人就要把他的手指重新归为零，由代表十位数的人伸出一根手指表示有了一个10。这个计数过程反复不断地继续下去，直到“算盘”上的手指都用完为止。

因为算盘在加减计算方面非常方便有用，所以这种工具直到今天还有人在使用。但算盘的结构使它在乘除计算方面效果不是很好。从原理上讲，乘法就是反复多次地加一个数字（比如25 × 4 = 25 + 25 + 25 + 25）；用算盘也能做除法，但非常复杂和耗时。很明显，如果要进行一些复杂或者大型的计算，算盘就帮不上什么忙了。但它确实可以用来完成一些分布式的计算工

作，你可以让两个或者更多的人来处理同一批数字以核对计算结果是否正确，你还可以把大批量的计算工作分配给许多人，让每个人用算盘完成一部分计算。当然，我们现在已经有了更快的计算办法。我们下面就来看看前人为了快速完成计算工作而发明的一些办法。



第一步：输入23。 第二步：加上7。 第三步：调整进位，得到结果。

图1-3 用算盘把23和7加起来

1.2 早期的计算器

随着我们对周边世界的了解一天天地深入，我们对自然界中的数学关系也有了越来越深的认识。这些知识逐步积累起来，人们就需要有一些能够帮助完成计算工作的工具。大约在1600年，伽利略开始在他的物理观测活动中引入数学概念。比如说，他发现不同重量的物体从同一高度落到地面上时的速度是一样的。一颗石子落到地面所需要的时间与从同一高度上落下的木球所需要的时间也完全相同。为了给他观测到的各种物理现象做出合理的数学描述，就产生了对新计算工具的需求。伽利略为自己的工作发明了许多种计算罗盘。

伽利略并不是对自然现象做出数学解释的惟一人选。约翰·纳皮埃尔就因发明对数而名垂青史，他在1614年发表了对数方面的初步成果。对数在快速乘除计算方面很有用，但手工计算对数需要花费很长的时间。为了节省时间，纳皮埃尔发明了一种被称为“纳皮埃尔机”的计算装置，它称得上是计算尺方面的老前辈。这种工具使用起来很不方便，因此人们转而寻求新的更好的办法。在纳皮埃尔发表关于对数的第一篇论文的七年之后，威廉亚姆·欧特雷德于1621年发明了计算尺。从那时起直到20世纪70年代，计算尺一直是数学系学生的必备之物；从20世纪70年代开始，因为计算器成为一种人人都买得起的普通消费商品，计算尺才逐渐退出了历史的舞台。

我的一个同事很有幽默感。他在墙上挂了一个工程师应急工具箱。这个工具箱很值得玩味，它体现了工程计算和数学的基本概念。这个工具箱装在一个半英寸深的镜框里，玻璃后面是一把计算尺、一支铅笔和一个算盘。在这些东西的上方有一块牌子写着“应急工具箱：在停电等紧急情况下，打碎玻璃”（如图1-4所示）。这虽然有些可笑，但也发人深省。计算机是能帮助我们做很多事情，但人们利用非常简单的工具也能完成不少工作。

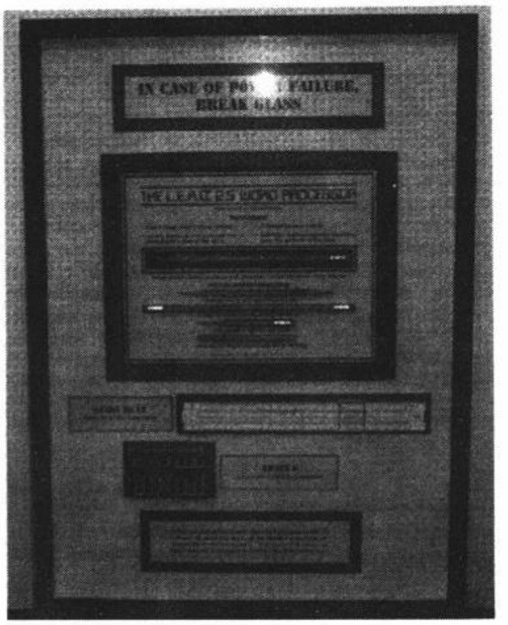


图1-4　应急工具箱

还有其他一些人也忙着在发明加快计算工作的办法。在1623年，一个名为威尔海姆·施卡德的德国天文学家制造了一台能够完成加减法的机械式计算器。这台计算器在使用者的帮助下甚至还可以完成乘法和除法运算。虽然施卡德制造的机器一台也没有留存下来，但他的笔记却保留至今。许多人根据他的笔记造出了计算器，这些机器工作得都不错。

多年之后，其他数学家对施卡德的四用计算器进行了改进。布莱瑟·帕斯卡对齿轮装置进行了简化。微积分的发明人之一高特菲尔德·威尔海姆·冯·莱布尼兹改进了乘法计算装置。1673年以后，对莱布尼兹的设计就很少再有人改进了。这种情况一直持续到1970年人们发明了更好的桌面计算器——电子计算器——才告结束。

1.3　可编程计算机器

接下来的故事要从传奇计算人物查尔斯·巴贝奇的横空出世开始讲起。巴贝奇在计算方面有三件事广为人知：分析机（Analytical Engine）、差分机（Difference Engine）以及他与Ada Lovelace的友谊。在这三个故事发生之前，他对数学还有一个巨大的贡献：他说服英国人接受使用世界其他地区使用的数学符号。他和几个朋友把一位法国数学家拉科罗尼斯的著作从法文翻译为英文。英国长期以来一直使用着牛顿数学记号，而其他大陆使用的都是莱布尼兹记号。两位大数学家分别建立的记号系统都很好，可使用莱布尼兹记号的人要比使用牛顿记号的人更多。翻译一篇领先的微积分论文虽然只是很简单的一个举动，但它却促成了英国的数学家开始接受并使用欧洲其他国家使用的数学语言。这使欧洲数学家之间的信息交流有了极大的改善。除了帮助数学界使用一致的数学记号之外，巴贝奇还设计了英国便士的铸造图案、编制出第一套精确的人寿保险费率表、为科学杂志撰写了许多篇文章，总之，在他感兴趣的许多领域都留下了

不少成果。

巴贝奇对计算界的第一个重大贡献是差分机（Difference Engine）。巴贝奇注意到大多数方程都可以用一种名为“差分法”的数学方法进行求解。表1-1就是用来计算数字平方值的差分表。表中的第一列是数字本身，第二列是该数字的平方值。第三列给出的是第一差分值——比如说，$1^2 = 1$，$2^2 = 4$，它们的第一差分值就是“4 - 1 = 3”。第二差分值永远等于2。有了这些知识，我们就可以快速计算出后续数字的平方值了。比如说，7^2将是“2 + 11 + 36 = 49”，而8^2是“2 + 13 + 49 = 64”。只要明白了这个道理，你就可以无限次地重复这个过程，还可以通过第二差分值来检验计算出来的平方值是否正确。如果第二差分值不等于2，我们就会知道在计算过程中出现了错误。再换个角度来看一下，任一给定数字的平方值有下列的形式：

$(n + 1)^2 = n^2 +$ (n的第一差分值) + 2

表1-1 [1,6]平方的一次差分值

值	值的平方	第1差分值	第2差分值
1	1		
2	4	3	
3	9	5	2
4	16	7	2
5	25	9	2
6	36	11	2

类似的方法可以用来求解普遍的高阶表达式——你只需要找出哪个差分值会重复其自身就行了（第一差分值、第二差分值、第三差分值等等）。巴贝奇正确地认识到能够求解任何高阶表达式值的机器将要比只能求解特定高阶表达式值的机器更有价值。在1822年6月14日，巴贝奇向英国皇家天文学会（the Royal Astronomical Society）提交了一份研制这样一台机器的图纸。这份报告还附有一台简单的能够计算平方值、立方值和一个高阶表达式值的“概念样机”（这台“计算机”使用电传打字机作为人机界面。那个高阶表达式是$x^2 + x + 41$，这个公式可以用来产生大数值的素数）。

英国皇家天文学会对巴贝奇的这个想法很感兴趣，决定资助巴贝奇着手制造这样的差分机（Difference Engine）。但身为工程师的巴贝奇过于精益求精，他的做法在他的资助者看来几乎是浪费金钱：巴贝奇不是先制造出一台能够工作的东西然后再进行改进，在差分机的制造过程中不断地修改着它的设计，而这些修改经常会因为技术上又有了新的改进而从头开始。而巴贝奇在处理人际关系方面又做得很缺乏技巧，这就进一步激化了双方的矛盾。因为资金来源问题和政治方面的压力，差分机的研制工作在1833年完全停顿了下来。那台未完全的机器现在陈列在伦敦的大英科学博物馆里。

差分机的研制工作虽然没有成功，但此时的巴贝奇又有了另外一种新的构想，即后人所称的分析机（Analytical Engine）。他设计的机器可以存储1 000个50位的数字（即有了所谓的“内存”概念），这些数值通过一套齿轮系统（即所谓的“中央处理器(CPU)”概念）来进行处理。他使用由Joseph Marie Jacquard发明的穿孔卡片来控制每次运算（这些穿空卡片的作用与其在20世纪出现的电子计算机中的作用完全相同，即存储计算过程——也就是我们现在说的程序）。巴

贝奇与Ada Lovelace合作研制这台装置，可惜的是这台机器也没能最终完成。但带“内存”和“CPU”的通用计算装置的想法却给现代的人们以启迪，而他的想法和设计与研制第一台电子计算机的工程师们惊人的相似。

1.4 电子计算机

在巴贝奇时代和第二次世界大战之间，人们又发明了许多种计算机器。到了战争开始的时候，电子机械式的计算机已经出现了。在今天看来，这些计算机更像是以穿孔卡片为输入介质的计算器，在当时已经能完成许多计算工作了。它们帮助人们进行各种统计调查的分析工作和各种财务方面的计算。当然，这些“计算机”在今天看来都存在很多问题。它们最大的两个缺陷是计算精度不足和运算速度缓慢。如果这些“计算器”的机械制造精度不足或者齿轮出现了磨损，就无法保证计算的精度。而齿轮传动的速度存在着极限，计算速度因此受到了很大的限制。（类似的问题也存在于电子计算机上，这也是为什么人们正在研制光子计算机的原因。）

第二次世界大战促使人们开始思考如何制造出运算速度更快，运算结果更准确的计算机器。全世界的人们都想到了要用电子开关来代替机械开关。英国于1943年12月制造出了一台名为“克鲁索思”（COLOSSUS）的机器，它被用来破译德国军方的通信密码。在大西洋的彼岸，美国人制造出名为ENIAC的机器来计算各种投射式武器的弹道表。战争初期使用的机械式计算机器计算出这些表格需要三天的时间，而那些电子计算机只要30秒就可以完成同样的工作。

战争结束后，研制战时计算机的人们开始把注意力转向如何把那些机器用在和平时代。到1970年的时候，人们已经发明出各种各样能够扩展计算能力的设备。我们有了传输控制协议和因特网协议（Transmission Control Protocol / Internet Protocol，即TCP/IP）、以太网、ARPANET网，有许多台计算机分布在世界各地，运行着各种各样的操作系统。在一台单独的计算机上，不同的程序通过共享内存、管道以及其他装置彼此进行通信。即使到了今天，我们仍然要在同一台计算机上分布运行不同的程序任务——你每次把一份电子表嵌入到一份字处理文档里去的时候，就会做这样的事情。随着计算能力的持续扩展和网络连接性能的不断提高，人们进入了一个新的舞台——让所有这些计算任务和谐地运行在许多台机器上。

1.5 分布式计算

感谢大家听我唠叨了这么多关于计算的历史。我们从那些古老的工具和构思开始已经走了这么远，而且我们还将继续不断地前进下去，每当我想到这一点，就有些控制不了自己的思绪。具备ENIAC同样功能的计算机器已经不再需要占据好几个房间大的地盘，计算机键盘的面积比芯片大得多了。这是多么不可思议的发展变化！更激动人心的是，从计算技术的发展历史我们可以清楚地认识到对这些机器还可以提出更高的要求，而这些要求并非遥不可及。这是许多既有聪明才智，又充满进取精神的人彼此借鉴各自的成功经验而不断努力的结果。分布式计算在计算技术发展史上自有其一席之地，我们将要在本书里学习的SOAP技术就是在它的基础上产生和发展起来的。

在我们的现实生活中存在着各种各样分布式计算的例子。早期的分布式计算可以说基本上

都是一些“人海战术”。数学家会利用对数表来简化一些复杂而又耗时的计算工作。每当我们利用别人工作成果的时候（比如水井的深度、房屋的高度、两点之间的距离等），实际上就是在进行分布式计算。

即使在穿孔卡片被广泛使用的时期，我们也还是离不开分布式计算。前一项工作的结果被写到一盒卡片上，计算机操作员会把这些卡片再作为输入数据馈入到另外的程序里去。人们靠“脚”而不是靠网络电缆来实现不同程序之间的衔接和数据（即穿孔卡片）的传输。

总的说来，分布式计算指的是在彼此连接着的多台计算机上分别执行不同的功能，而这些功能共同构成一个完整的程序总体。这种分布式概念使我们能够把数据放到最合理的地方去得到处理。我最得意的分布式计算的例子是天气预报（这个例子多少有些臆想的味道，实际情况也许和我想像的不一样。但我发现这个例子很有说服力，特别是在向那些从来没有分布式计算经历的人们介绍这一概念时更有用）。精确的天气预报需要处理器具备非常巨大的计算能力。预测天气变化需要有专业的气象学家和强大的计算机，我想只有国家气象局之类的机构才能应付得来。出于成本方面的考虑，给每位气象学家分别配备一个工作站，再把这些工作站连接到一台超级计算机上要比购买多台超级计算机合算得多。在我们的例子里，气象学家Beth负责写出德克萨斯州达拉斯附近地区的天气预报。为了完成这一工作，她使用了一个能够输出预计的气温变化范围、天气变化图和其他有关数据的程序。当Beth坐下来预报未来三天的天气形势时，她要运行一个程序，输入一些数据，并且绘制出需要的图表。从她的角度看，所有这一切都发生在自己的工作站上。可实际发生的情况是怎样的呢？

实际情况可能是这样的：

1) Beth向运行在她工作站上的一个客户应用程序输入了有关的数据。

2) 客户应用程序向负责处理天气预报数据的超级计算机发出一个请求。

3) 超级计算机完成有关计算并发送回被请求的数据。

4) 客户应用程序把计算结果显示在工作站的屏幕上。

这种安排有什么好处呢？又有什么意义呢？一个精心设计的分布式计算会在最需要的地方执行最需要的代码。就天气预报的例子来说，把用户操作界面放在Beth的工作站计算机上和让更强大的超级计算机来处理气象数据是最合理的做法。正是分布式计算技术使这一切成为了可能。

现在的人们几乎无时无刻不在使用着分布式计算。万维网（World Wide Web）就是我们身边最明显的例子。万维网上的内容是如此的丰富，想让每个人的计算机上都容纳有全部的Web资料是不现实的，因此我们只会在必要的时候用超文本传输协议（Hypertext Transfer Protocol，即HTTP）来请求它的一小部分，而且我们只在最合理的地方对这些信息进行处理。这其中的道理是显而易见的，因为资料肯定应该存放在其作者可以访问的机器上。这些机器通常都有高速的因特网连接，并经过专门的配置。就拿我的Web站点来说吧，我在华盛顿的家里编写文章的内容，但要把写好的文章放到一台位于犹他州的主机上，为此我还得向负责运行那台主机的公司支付费用。万维网算得上是世界上最具规模的远程过程调用（remote procedure call，简称RPC）机制了。在万维网上，你用GET命令来请求获取数据对象（通常是一个文件），而主机计算机会根据你的请求做出相应的响应。从请求方的立场看，远程主机提供的不过是一些字节流而已。这些字节流可能来自文件，也可能是动态生成的内容。举个例子，如果你只是一个普通的用户，

那么当你请求一个Java服务器页面（Java Server Page，JSP）或者一个主动式服务器页面（Active Server Page，ASP）的时候，你永远不会知道你正在查看的页面是由哪些指令生成的。你只是用一个名字从某个服务器上请求获取一个数据对象而已，页面的具体构成情况可能要取决于当时的时间、英国伦敦的天气或者你在那个Web站点上以前的所作所为等因素。

计算技术从自然科学里吸取了大量的营养。比如说，在巴贝奇和他的伙伴翻译拉科罗尼斯的著作之前，英国使用的数学语言与世界其他地区使用的是不一样的。当数学家们开始使用同一种记号的时候，他们就能更好地理解彼此双方的工作和想法。类似的事情在计算技术的发展过程中屡见不鲜。现在，要想销售一种不使用TCP/IP的网络解决方案已经是越来越困难了。分布式计算也有它自己的标准。我们将把这一小节的注意力集中到与RPC调用有关的技术标准方面，包括分布式计算环境（Distributed Computing Environment，DCE）、分布式组件对象模型（Distributed Component Object Model，DCOM）和通用对象请求协调体系结构（Common Object Request Broker Architecture，CORBA）。需要提醒大家注意的是SOAP并不是只能用来处理RPC调用，它还能够对工作流和其他类型的消息进行处理。

1.5.1 DCE

DCE是开放软件基金会（Open Software Foundation，简称OSF。这是一个由计算机硬件和软件供应商组成的技术论坛）在1990年提出来的，而分布式计算在DCE正规化之前已经出现很久了。DCE针对的是OSF组织中各家供应商的各种产品，对这些产品的用途和网络服务进行了标准化。自从人们使两台计算机相互“交谈”开始，分布式计算就出现了。分布式计算需要有一些网络服务来加强其安全性和可靠性，而DCE就是针对这类服务的规定。这些服务包括：

- 调用式目录服务（Cell Directory Service）：分布式计算环境里的许多资源都是有名字的，比如用户、计算机、应用程序等，调用式目录服务提供了访问这类资源的办法。它在功能上类似于微软公司的Active Directory和Novell公司的Directory Service。
- 分布式时间服务（Distributed Time Service）：保证一个DCE环境中的所有时钟都是同步的。因为同一个DCE环境中全体计算机的时钟都是同一时间（误差不超过1毫秒），这就使分布式应用软件的可靠性得到了加强。有了统一的时间信息，不同的过程就能按它们被调用的先后顺序正确地得到执行。
- DCE安防（DCE Security）：确保系统中的对象（应用软件、用户或者硬件等）只能访问你想让它们访问的内容。
- 远程过程调用（RPC）：允许应用程序调用另外一个进程中的函数功能。所谓“另外一个进程”可以在也可以不在发出调用指令的这台机器上；被调用函数的有关特征—包括输入、输出以及输入/输出参数等要用接口定义语言（Interface Definition Language，IDL）来描述。
- 线程（Threads）：线程是为服务器进程实现的，它们的作用是使服务器进程能够同时处理多个请求。

在这些服务当中，RPC是程序员们最感兴趣的东西（我的意思并不是想贬低安全问题的重要性，安全问题也是大事。但在我学习新技术的时候，如果我知道它已经考虑了安全问题，我就

会长出一口气："好了，又少了一件需要我担心的事情"。所以，我在这里也就不在安全问题上多花笔墨了）。客户和服务器之间交互动作如图1-5所示。这个图第一眼看上去是相当简单的：你先用IDL语言对这个接口进行定义，安排好客户桩（client stub）和服务器桩（server stub），然后编写一个调用有关代码的程序。其实在一个DCE环境的RPC调用功能的具体实现里隐藏着许多复杂的因素。为了最大限度地灵活适应多变的环境，这些RPC实现允许开发人员指定诸如数据的字节存储顺序。一个数字的字节存储顺序是由它在内存中的存储方式决定的。不同种类的CPU所采用的字节存储顺序是不同的，采用哪一种字节存储顺序通常与CPU设计者采用的优化技术有关。字节的升序存储方式和我们平时用笔写出数据时顺序是一致的；而字节的降序存储方式会把一个字的低位字节存放到高位内存位置里去，按我们日常的书写习惯来看两个字节正好是前后颠倒的。请看下面的例子，当我们用32bit来写出数值"3"时，两种不同的字节存储顺序是这样的：

1) 字节的升序存储方式：0x00000003。　高位半字的值是：0x0000
　　低位半字的值是：0x0003

2) 字节的降序存储方式：0x00030000。　高位半字的值是：0x0003
　　低位半字的值是：0x0000

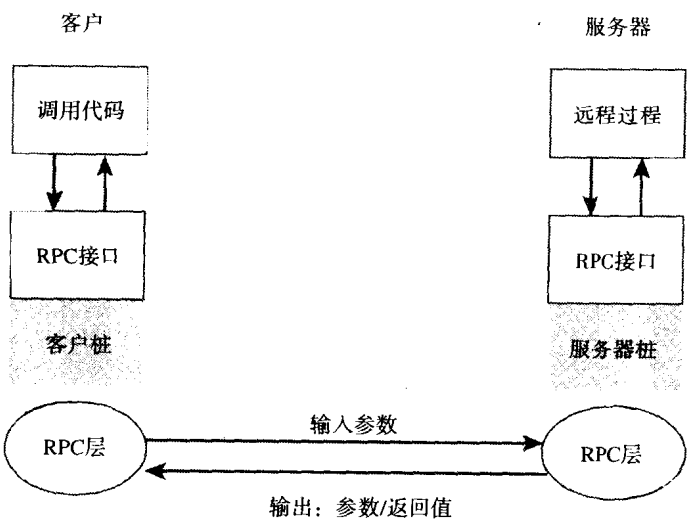


图1-5 客户/服务器模型下的RPC基本框图

灵活的适应性是有代价的。如果一台降序存储机器没有识别出一个以升序存储格式表示的数字，就可能把数值"3"错误地解释为数值"196 608"（数值"196 608"的降序格式正好是"0x00000003"）。出现这类数值识别错误的原因通常在于程序员对RPC的有关规定理解得不够透彻，在开始实现RPC功能时想当然地认为客户和服务器都将是同一种平台（即操作系统和CPU）。为了提高RPC的处理速度，开发人员可能会用自己编写的代码（这种代码被称为"定制引导器"）来把数据发送到线路上去。请看下面这段C语言风格的结构定义：