HZ BOOKS

Borland INPRISE
核心技术丛书

李维 著

Delphi 6/Kylix 2

SOAP/Web Service 程序设计篇

机械工业出版社
China Machine Press

Borland INPRISE
核心技术丛书

李维 著

Delphi 6/Kylix 2 SOAP/Web Service 程序设计篇

机械工业出版社
China Machine Press

本书是专门讨论Delphi 6中SOAP/Web Service新技术的使用书籍。主要内容包括：SOAP/Web Service技术介绍、SOAP功能规范、各种SOAP/Web Service应用技术和架构的讨论、结合数据库的SOAP/Web Service应用系统的开发等等。

本书内容深入浅出，实用性强，是一本SOAP/Web Service技术的完整指南。本书所附光盘包括书中的所有示例代码。

图书在版编目（CIP）数据

Delphi 6/Kylix 2 SOAP/Web Service程序设计篇/李维著. -北京：机械工业出版社，2002.3

（Borland/Inprise核心技术丛书）

ISBN 7-111-09909-5

Ⅰ.D… Ⅱ.李… Ⅲ.Delphi语言－因特网－程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字（2002）第011053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）

简体字改编者：陈剑瓯

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3月第1版第1次印刷

· 787mm×1092mm 1/16 · 25.75印张

印数：0 001-5 000册

定价：65.00元(附光盘)

序

科技的进步真是非常迅速，从1999年Delphi 5推出之后，软件发展的趋势不断地演变。Web应用已经成为主要的应用，而多层架构也逐渐被许多系统所采用，特别是结合Web应用和分布式架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Delphi 3第一次以多层架构做为发展的主轴，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。这使我们不禁要佩服那些Delphi研发人员的眼光了，特别是所有Delphi程序员都久闻大名的Anders Hejlsberg以及当初坚持Delphi 3中要加入分布式功能的Zack Urlocker，笔者很庆幸能够有机会恭逢其时，相信许多读者也历经了这场革命性的信息科技演变。

Delphi 6提供的新功能是延续Delphi 5的自然发展，并且融合了目前许多最重要的软件技术，让Delphi开发人员能够及时地使用Delphi 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（Delphi 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最新的高效率且跨平台的数据访问引擎，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。这些新的软件技术每一个都非常精彩和实用，足让Delphi的软件人员能够开发主从架构、Web应用和分布式多层应用系统。当然这些技术也都足以写成专门的书籍，详细地说明如何运用这些技术。

本书是专门讨论SOAP/Web Service技术的实用书籍，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中介软件技术看得出来。不但Java将把SOAP/Web

Service定义进核心，Microsoft的.NET也是以SOAP/Web Service做为核心的技术。Delphi 6不但是第一个完整支持SOAP/Web Service技术的开发工具，而且Delphi还在不断地改善SOAP/Web Service方面的功能，让它们更强大，也确保Delphi 6的SOAP/Web Service技术能够顺利地与所有使用其他开发工具开发的SOAP/Web Service应用系统相互沟通。此外Delphi 6和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技术架构。

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。在本书的头两章中将会说明为什么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与现在使用的各种组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做到的事情。接下来，本书会使用Delphi 6来实际开发SOAP/Web Service应用系统，让读者能够先掌握实际的开发能力。在第4、5、6章中本书将以实际的范例来介绍SOAP的功能规格，让读者能够切实地了解什么是SOAP、SOAP设计的概念以及SOAP的技术细节。

从第7章开始，本书将涉及高级的SOAP/Web Service技术，开始讨论各种SOAP/Web Service应用技术和架构。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Delphi 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上实际使用Delphi 6调用由其他开发工具开发的SOAP/Web Service应用系统，让读者真正地领略SOAP/Web Service的威力，了解SOAP/Web Service提供的强劲集成能力。相信在读者阅读完本书之后一

定能够切实地掌握SOAP/Web Service技术，准备下一轮的挑战。

使用Delphi一直是令人非常高兴和舒服的事情，因为不但可以使用Delphi开发各种应用系统，也能够不断地学习到最新的软件技术，提高个人的价值。在Visual Basic停止开发第7版而以VB.NET来代替，PowerBuilder的发展速度也越来越缓慢的时候，Delphi仍然不断地快速进步。它是现在最佳的Windows原生开发工具，将和即将推出的C++Builder 6一起成为Windows下最好的RAD工具，而Kylix 2也已经是Linux下市场占有率最高的RAD工具。未来Borland将会持续地发展.NET下的Delphi，继续为使用Delphi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我的读者这么多年来不断地鼓励和支持我写作，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一代的应用系统开发环境。

李维于新店

目　录

第1章 SOAP和Web Service的概念

记得是在4、5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++调用执行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是属于特定的平台，都太复杂，并且受限于特定的语言、开发工具和操作系统，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++的文章就是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，并且是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本封包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本格式封装对于远程函数的调用，然后由某一种机制来真正调用远程的程序代码，然后返回函数结果。至于远程的CalculateInterest函数是在什么地方执行？哪一个机器和操作系统？以及是使

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。

这是多么一个令人心动的想法，而这个想法在数年之后的现在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仍然在不断的进步之中，相关的规范和技术也正如火如荼的开发之中。相信这个构想会很快实现并被改善。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成为现在重要的软件趋势。

1.1 Internet/Intranet和开发模式的演进

从1、2年前到现在影响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。第一轮重要的技术革命是HTTP，它为Internet/Intranet和电子商务都带来了重要的影响。第二轮是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便可以开发应用系统，Java几乎提供了跨越平台的机会。第三轮则是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是令人头痛的问题。

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Internet/Intranet应用系统仍然使用了其他的技术，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Internet/Intranet应用系统也无法互相沟通和集成。例如ASP无法

直接调用Servlet或是JSP，而专为Java提供的EJB组件模型也无法由ASP轻易调用，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。从这一点来看，XML似乎是非常成功的技术。

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。因此在下一轮的Internet/Intranet技术演进中，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的概念。

1.2 调用和数据的集成机制

既然XML提供了数据交换的标准，因此当不同的Internet/Intranet应用系统在互相集成时，就可以直接使用XML技术来做为彼此沟通封装的标准。当Internet/Intranet应用系统需要调用彼此提供的服务时，不需要知道对方实现的技术，只要以XML封装彼此约定的服务进入点，就可以顺利地使用对方的服务。这样不但达成了标准交换数据的机制，也可顺利使用XML集成各种不同的Internet/Intranet应用系统。

这个概念可以使用图1-1来说明。例如当在Windows平台上执行的ASP想要使用在Linux平台上执行的服务时，只要双方约定了提供服务的标准接口，那么ASP便可以使用XML封装标准的调用，然后通过Internet/Intranet传递到远程的Linux中。而远方的Linux在收到这个以XML封装的请求时，便可以解析此XML封装的信息，执行ASP需要的服务之后，再同样使用XML封装执行的结果，再传递回远方ASP。

当ASP收到Linux返回的XML封装结果之后，再从XML中取出执行结果即可完成工作。

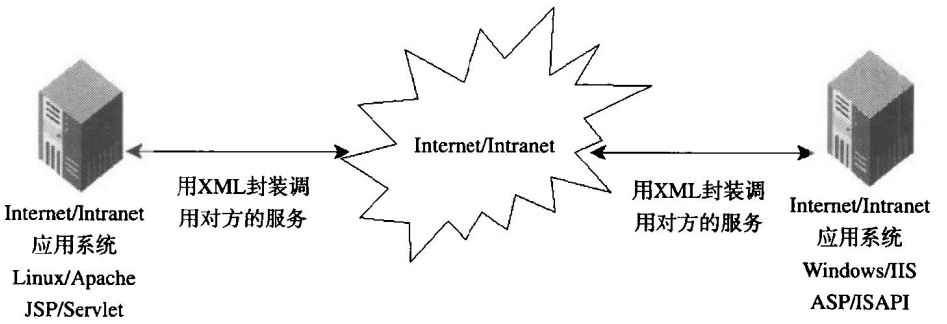

图1-1 使用XML封装集成Internet/Intranet应用系统的机制

从上面的说明中可以知道，要完成不同Internet/Intranet应用系统的集成工作，那么必须解决下列问题。

1) 标准的数据交换技术，即XML。

2) 如何封装调用的服务。例如如果上述的Linux提供了各种服务，那么远方的ASP如何封装服务的调用？如何封装传递的参数？如何传递简单类型的参数，例如字符串和整数值？又如何封装复杂的数据类型，例如记录数据、影像、图形数据等？

3) 使用什么实体沟通通信协议？

4) Internet/Intranet应用系统如何约定彼此提供的服务？

5) 不同的Internet/Intranet应用系统又如何找到它需要的服务？

6) 服务是由什么技术来实现的？

上述问题可以使用许多技术来解决或是克服，但是这些技术可能是属于特定厂商的技术，或是没有获得大多数人的接受。不过现在情况已经改变了，这些问题的答案可以被现在逐渐成为标准技术的SOAP和Web Service解决。

本书稍后的内容会详细说明什么是SOAP和Web Service，不过简单地说SOAP定义了如何交换类型和具有结构的信息，它是一个Wire Protocol并且使用XML做为封装信息的标准。而Web Service则是使用SOAP做为通信的标准，并且提供外界标准的服务接口以便让各种客户端应用程序能够通过SOAP调用服务接口，进而使用Web Service提供的功能。因此SOAP标准可以回答上面的第2个问题，Web Service则可以回答第4个问题。

由于SOAP只是一个Wire Protocol，因此在实际进行沟通时还是必须使用特定的通信协议(Transport Protocol)，例如HTTP、SMTP或是IIOP等。在SOAP的功能规格中定义SOAP如何绑定使用HTTP进行实体的通信。因此这又回答了上面的第3个问题。上面的第5个问题则可以由本书稍后介绍的UDDI(Universal Description Discovery and Integration)机制来回答，至于最后的问题则是本书后半段的重点，使用Delphi 6实现Web Service来提供集成的服务。

1.3 异构平台和通信协议

由于SOAP使用了XML做为封装和交换信息的标准，因此可以在各种不同的平台中使用，只要不同的平台支持并且能够解析和处理SOAP的封包。由于SOAP只是一个定义如何交换数据的Wire Protocol，因此它可以使用现在任何的标准通信协议做为实体沟通的机制。在SOAP的功能规格中是选择以HTTP通信协议做为第一个实现的实体通信协议，因此由SOAP封装的信息可以包含在HTTP的封包中进行沟通。

除了HTTP之外，软件厂商当然也可以实现其他的通信协议来传送SOAP封包，例如CORBA的厂商可以使用IIOP做为实体的通信协议。当然，其他的通信协议也可以做为SOAP的实体通信协议，例如下一个最有希望被实现的通信协议就是SMTP。图1-2说明了SOAP和各种实体通信协议的关系。

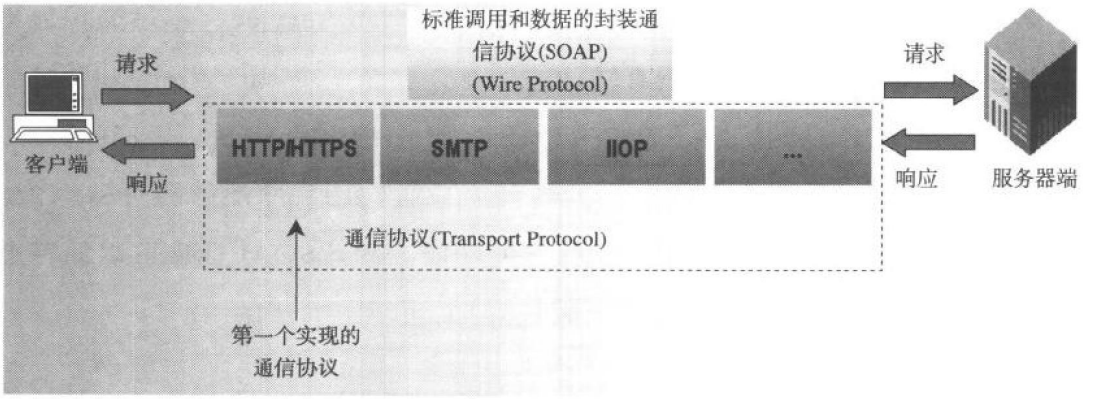

图1-2 SOAP沟通架构以及SOAP和实体通信协议的关系

由于SOAP选择了以HTTP做为第一个实现的传送通信协议，而HTTP又是现在所有平台和操作系统都已经接受的标准通信协议，因此通过使用SOAP和HTTP软件开发人员可以轻易地集成各种平台以及实现模型，包含了COM/MTS/COM+、CORBA和EJB等。

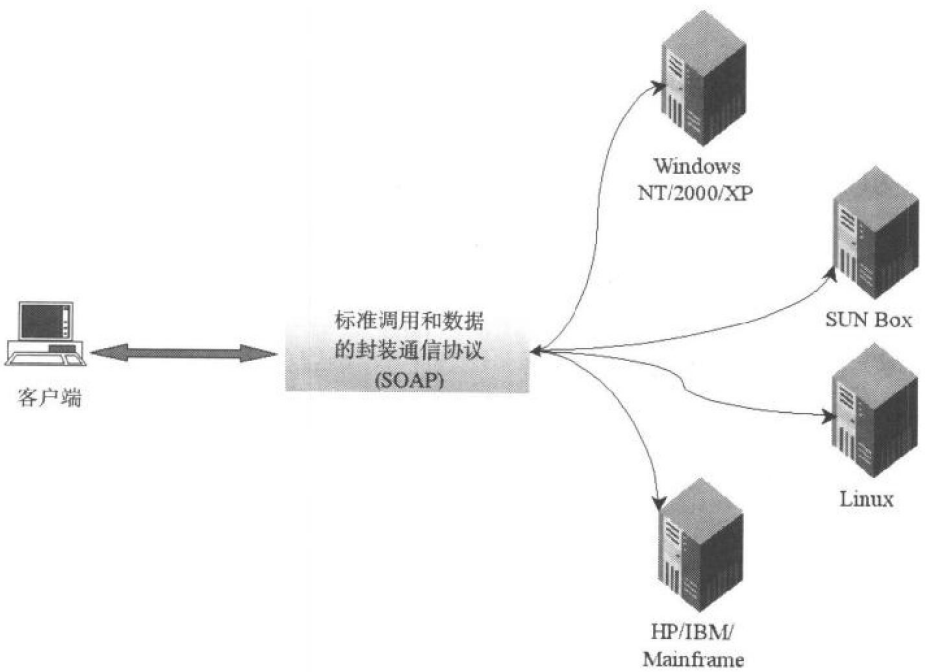

图1-3 使用SOAP机制可以集成各种不同的平台

图1-3就说明了这种可能，各种不同的客户端可以使用SOAP标准调用和使用各种不同的后端平台提供的信息。对于客户端来说只需要使用SOAP这个标准就可以进行远程服务的调用，而不需要知道后端是使用什么技术实现的。

现在SOAP几乎已经被所有的实现语言所支持，例如Java有Class Library，C/C++有函数库，VB有MS Soap Toolkit，Delphi有Soap组件和向导等等。因此现在软件开发人员就可以使用SOAP技术来集成异构平台，或是开发分布式应用系统。不过即使是如此，软件开发人员仍然面对一个非常困难的问题，那就是虽然通过SOAP软件开发人员可以封装交换信息，但是不同的实现技术如何调用其他实现技术提供的服务呢？

这个问题也就是引发软件服务观点的基础，也是Web Service要解决的问题。

1.4 软件的服务概念

应用系统的演变从纯粹以处理数据为中心，慢慢地转变到使用关系数据库储存和分析数据，再到Information On Demand，逐渐地改变使用数据的方式。在Internet/Intranet技术出现之后，数据的使用性更加多样化了，而且必须集成各种不同的使用需求以及不同平台的功能，因为通过Internet/Intranet世界已经逐渐成为一个网网相连的大型计算世界。

但是如何集成各种不同平台、不同操作系统和使用不同实现技术提供的功能却成为一个非常困难的问题。因此虽然CORBA是一个为了解决集成异构平台的组件技术，但是当CORBA要集成其他的组件模型时，例如COM/COM+，也需要各种Bridge技术来帮助。例如CORBA-和COM Bridge软件等。

但是这些Bridge技术通常都实现在特定的组件厂商之上，而且也限制用户必须使用特定的组件模型版本等。此外Bridge软件通常也是属于特定厂商

的专属技术，并不是一个开放的标准。由于这些原因，因此要使用Bridge技术在Internet/Intranet这个多元的应用环境中是非常困难而且不切实际的。

从前一小节中我们了解了SOAP是封装交换信息的标准。既然我们已经可以使用SOAP来封装标准交换信息，那么如果能够再定义一种机制，这个机制可以让各种不同的实现技术都能够遵循，以对外提供实现技术所提供的功能；那么客户端便可以使用SOAP技术来封装交换信息，并且使用这种技术来调用实现技术的标准服务。如此一来不管后端的实现技术是使用COM对象，或是使用CORBA/EJB对象，或是只是一个单纯的CGI应用程序，客户端都可以使用SOAP和标准功能来调用这些实现程序代码，那么不就可以很简单地集成Internet/Intranet上各种不同的实现技术了吗？

图1-4说明了这个令人兴奋的集成架构。在图中各种不同的后端实现技术以一个标准的服务接口对外输出它提供的功能，而各种不同的客户端则使用SOAP标准来调用这个标准的服务接口以使用后端实现程序代码提供的功能。结合SOAP和服务接口标准，不但可以轻易地集成各种实现技术和异构平台，更棒的是这种标准架构非常简洁，而且易于理解和采用。

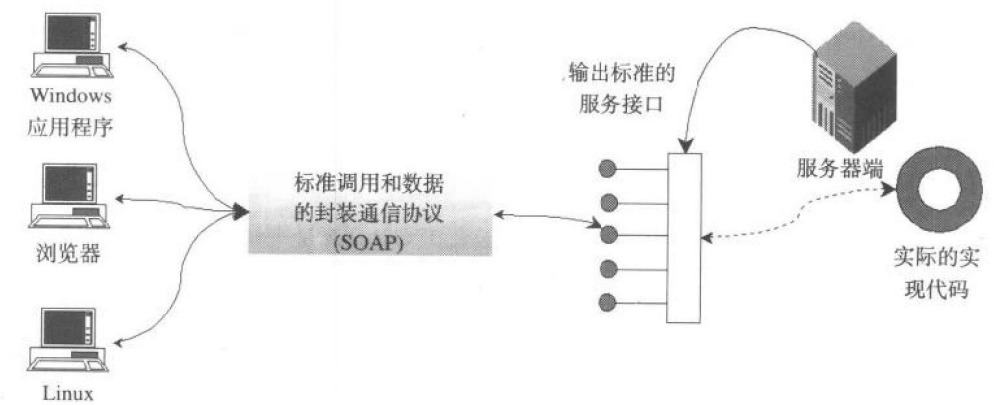

图1-4 不同的客户端通过SOAP和标准接口使用远程Web Service提供的服务

而以上所说明的就是Web Service的基础概念，也就是Web Service要解决

的问题。

1.5 Web Service的技术

事实上Web Service的概念非常简单，而且Web Service使用的基础技术大都是目前已经被广泛接受的开放标准。因此当不同的厂商在实现Web Service技术时，只是使用现有的标准技术，再加上特定语言或是平台需要的支持功能。

基本上Web Service的概念是使用一个标准的输出接口来定义实现程序代码提供的功能，以便让外界可以通过这个标准的输出接口来调用，而所谓的标准的输出接口就是WSDL(Web Service Description Language)。

WSDL是一个由XML组成的文件，这个文件内容描述了实现程序代码对外提供的函数原型，也就是各种可供调用的函数名称以及参数等信息。在实现程序代码提供了WSDL之后，客户端就可以通过WSDL来调用实现程序代码提供的服务，至于客户端如何通过WSDL调用会在本书稍后的章节中详细说明。

由于实现程序代码提供了WSDL之后，就可以接受外界的调用，因此实现程序代码并不会区分是客户端调用或是其他的后端实现程序代码来调用。只要是通过WSDL来调用，实现程序代码就可以提供它的服务。因此图1-5就说明了这个情形。

从图1-5中可以知道，当实现程序代码用WSDL提供了标准服务接口之后，任何客户端或是其他的实现程序代码都可以通过这个接口调用以取得服务。

图1-6则是结合说明SOAP和Web Service在Internet/Intranet中运作的原理。客户端通过SOAP封装需求信息，通过Internet/Intranet传递到后端的服