第23卷 第3期 计算机工程与设计 2002年3月
Vol. 23 No. 3 COMPUTER ENGINEERING AND DESIGN Mar. 2002

文章编号：1000-7024 (2002) 03-0030-05 中图分类号：TN915.04 文献标识码：A

# 网络服务与简单对象访问协议

余枝强， 薛云皎， 王春森

（复旦大学计算机系，上海 200433）

**摘 要**：网络服务是基于网络的分布式应用程序的基本构造模块，这些程序是以平台、对象模型和多语言方式构建的。SOAP技术通过一种基于XML的、平台无关的方式集成现有的网络服务到应用程序中，它有助于实现大量异构程序与平台之间的互操作性。首先简要介绍网络服务的定义，以及建立基于网络的分布式应用程序所要解决的基本技术问题；然后说明SOAP是如何克服存在技术（如CORBA和DCOM）的许多缺陷；最后通过一个实例，描述怎样通过SOAP调用现有的网络服务。

**关键词**：CORBA；DCOM；SOAP；网络服务

# Web services and the simple object access protocol

YU Zhi-qiang, XUE Yun-jiao, WANG Chun-sen

(Department of Computer Science, Fudan University, Shanghai 200433)

**Abstract**: Web services are building blocks for constructing distributed web-based applications in a platform, object model, and multilanguage manner. The Simple Object Access Protocol (SOAP) improves internet interoperability with an XML-based, platform-agnostic approach to aggregate the existing web services into applications. This article defines web services and the key enabling technologies of building distributed applications across the internet. And then describes the ways the SOAP overcomes many of the limitations of existing technologies, such as DCOM and CORBA. An example of how to call a web service through the SOAP is provided in the end.

**Key words**: CORBA; DCOM; SOAP; web service

## 1 网络服务一览

通常说来，一个网络服务只是一个作为服务的应用程序，通过Internet标准，该服务能与其它应用程序集成在一起发行。换句话说，它是可通过URL定位的自动将信息返回到需要它的客户端那里的一种资源。它主要是为了解决分布在Internet上的不同的应用程序之间的通信问题而引入的。

同组件一样，网络服务提供“黑匣子”函数，调用方无需知道该服务具体是怎样实现的。网络服务提供了被称为契约的精确定义的接口，此接口描绘了所提供的服务。开发人员可以将远程服务、本地服务和定制代码组合在一起而集成应用程序。例如，某公司可以将第3方提供的用户身份验证系统、个性化服务系统、信用卡处理系统集成到自己的在线销售系统中，以使自己的网站更具竞争力。下图显示的模型说明了为了生成分布式网络应用程序应怎样链接网络服务。

现在需要的是以一种通用的方式将所有的网络服

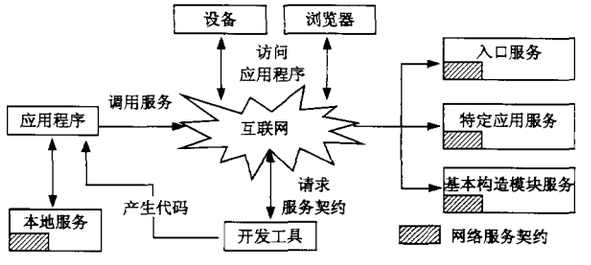


图1 网络服务模型

**收稿日期**：2001-02-08

**作者简介**：余枝强（1978-），男，福建人，硕士研究生，研究方向为软件工程。 薛云皎（1978-），男，云南人，硕士研究生，研究方向为软件工程。 王春森，男，副教授，主要研究领域为软件工程。

务无缝链接起来。通过这种方式，应用程序应该能够很容易地协调不同的网络服务，而且，创建一个新的、定制的网络服务应该变得简单。另外，开发者还应能集成任意的软件组件、应用程序、商业处理过程以及其它的任意的智能属性。这需要一种新的应用模型，它能为互联网上的分布式组件提供一组核心服务，让它们以一种快速、可靠的方式调用分布式的网络服务以及实现分布式组件互连。

## 2 目前的技术问题

历史上，解决这种分布式应用程序之间通信的方案通常是引入一种对象模型，比如微软的DCOM或对象管理组织（OMG）的IIOP或CORBA。传统上认为DCOM和CORBA都是合理的服务器端到服务器端的通信协议。但是，二者对客户端到服务器端的通信都存在明显的弱点，尤其当客户机被散布在Internet上的时候。

DCOM和CORBA/IIOP都是依赖于单个厂商的解决方案来发挥协议的最大优势。尽管这两个协议都在各种平台和产品上被实现了，但现实是选定的发布需要采用单一厂商的实现。在DCOM的情况下，这意味着每台机器都要运行Windows系统。（尽管DCOM已经被转移到其它平台，但它只在Windows系统上获得了广泛的延伸）。在CORBA情况下，这意味着每台机器都要运行同样的ORB产品。的确，让两个CORBA产品用IIOP相互调用是有可能的，但是许多高级的服务（如安全和事务）此时通常是不可交互的。

DCOM和CORBA/IIOP都依赖于周密管理的环境。两个任意的计算机通过DCOM或IIOP在环境之外被成功调用（calls out of the box）的几率是很低的，在考虑安全性的时候尤其是这样。而且，这两个协议都有相当多的深奥的规则来处理数据排列、类型信息和位操作，这使得一般的程序员在没有领会ORB产品或OLE32. DLL的情况下去构造一个简单的CORBA或DCOM调用也变得很困难。

也许对DCOM和CORBA/IIOP来说，最大的弱点是它们不能在Internet上很好地发挥作用。如果防火墙或代理服务器分隔开了客户和服务器的机器，任何IIOP或DCOM包要通过的可能性是很低的。尽管一些厂商如Microsoft、Iona和Visigenic都已经建立了通道技术，但这些产品很容易对配置错误敏感而且它们是不可交互的。

## 3 利用HTTP和XML构筑更好的协议模型

一般而言，与具体组件技术紧密结合的实现在一个受控的环境中（比如在一个单一的、统一的体系架构中）能很好地被接受，但它们在网络环境中变得不切实际。网络服务采取了另外一种途径，它使用普遍存在的网络协议和数据格式，如HTTP和XML，进行通信。支持这些网络标准的任何系统都支持网络服务。而且，网络服务契约描述的是由网络服务生成和接受的消息，而不是描述服务是如何实现的。通过把重点放在消息处理上，网络服务模型就完全可以不考虑语言、平台和对象模型。只要用于解释服务容量、消息序列和所期望协议的契约得到认同，那么所实现的网络服务及网络服务用户就可以相互不同，并且不会影响会话另一端的应用程序。

网络服务模型对体系架构的要求很低，以确保网络服务可以在使用任何技术和编程语言的平台上实现和访问。对网络服务互用性的解决可只依靠网络标准。然而，为了使应用程序更容易使用网络服务，仅仅通过标准网络协议就可以访问网络服务是不够的。当网络服务和网络服务使用者能用标准的方式表示数据和命令、表示网络服务契约、算出网络服务所提供的容量时，网络服务才容易使用。

XML是定义一个标准的、可扩展的用于提供命令和典型数据的语言的明显的一种选择。XML被专门设计为描述数据的标准元语言。简单对象访问协议（SOAP）是以一种可扩展的方式使用XML表示数据和命令的工业标准。网络服务可选择用SOAP决定消息的格式。

XML也可用来描述网络服务契约。服务契约语言（SCL）定义了一种记录网络服务契约的XML语法。由于SCL是基于XML的，所以对开发者和开发工具来说，容易生成、解释契约。另外，Disco规范为服务提供者发布网络服务契约和相应的机制描述了一个标准方式，这将使开发者或开发工具可找到契约文献。

象SOAP, SCL和Disco这样的标准有助于开发者，因为他们不需要明白和实现所使用的每一个网络服务的访问方式。支持这些标准的更好的、已充分测试的、高性能的体系架构将由开发平台提供，这会大大简化整个开发过程。目前，Apache-SOAP2.0模型以及Microsoft .NET框架都提供了这些方面的支持。

以下重点介绍SOAP协议，以及它在网络服务模型中的应用。

## 4 SOAP协议

简单对象访问协议（SOAP）是Microsoft, DevelopMentor,IBM,Lotus Development Corp.等几大公司联合工作的成果，它以一种持续的、结构化的方式在分布式

应用程序间传送数据以及实现方法调用。它建立在现有的两个技术HTTP(也可以是其它协议，如SMTP)与XML基础之上。HTTP与RPC的协议很相似，它简单、应用广泛，并且对防火墙比其它协议更容易发挥作用。HTTP的最大优点是它已被广泛使用和接受。HTTP请求一般由Web服务器软件(如IIS和Apache)来处理，但越来越多的应用服务器产品在支持原有的DCOM和IIOP协议之外也开始支持HTTP。HTTP提供了IIOP或DCOM在组帧、连接管理以及序列化对象应用等方面大部分功能的支持，它所缺少的是用单一的标准格式来表达一个RPC调用中的参数。这正是XML的用武之地。

XML是一个与平台无关的中性的数据表达协议。XML允许数据被序列化成一个可以传递的形式，使得它容易地在任何平台上被解码。它有以下几个优点：

(1)有大量XML编码和解码软件存在于每个编程环境和平台上；

(2)XML是基于文本的，相当容易用低技术水平的编程环境来处理；

(3)XML是特别灵活的格式，它容易用一致的方式被扩展；

(4) XML也支持带类型的数据表达。正在推出的XML Schema规范为描述XML数据类型定义了一个标准的词汇集。

SOAP用XML为请求和响应参数编码，并用HTTP进行传输。具体地讲，一个SOAP方法可以简单地看作遵循SOAP编码规则的HTTP请求和响应。一个SOAP终端则可以看作一个基于HTTP的URL，它用来识别方法调用的目标。象CORBA/IIOP一样，SOAP不需要具体的对象被绑定到一个给定的终端，而是由具体的实现程序来决定怎样把对象终端标识符映射到服务器端的对象。

SOAP的XML特性是把数据类型的实例序列化为XML的编码模式。在发送SOAP请求消息之前，参数被序列化成为一个请求对象。同样被响应消息接收到的响应对象被反序列化为参数。一个类似的转变同样发生在调用的服务器端。SOAP的消息中，有4种类型的元素：结构元素、根元素、存取元素以及独立元素。soap:Envelope，soap:Body和soap:Header是仅有的3个结构元素。根元素是soap:Body或soap:Header的直接的子元素。SOAP消息中的信封(soap:Envelope)含有一个可选的信封头(soap: Header)以及一个信封体(soap:Body)。其中信封体只能包含一个根元素，这个根元素用来表达调用、响应或错误对象，它的标记名和域名URI必须与HTTP SOAPMethodName头或在错误消息情况下的soap:Fault相对应。可选的<soap:Header>元素用来传载被协议扩展所使用的信息，它可以有多个根元素，与消息相联系的每个头扩展对应一个。这些根元素必须是soap:Header的直接子元素，它们的标记名和名域URI表示当前存在扩展数据的类型。存取元素被用作表达类型的域、属性或数据成员。存取元素的标记名对应于类型的域名。一个独立元素表示至少被一个多引用存取元素引用的类型的实例。所有的独立元素用soap:id属性作标记，而且这个属性的值在整个SOAP Envelope中必须是唯一的。

SOAP请求是一个HTTP POST请求。SOAP请求的content-type必须用text/xml，而且它必须包含一个请求-URI(Uniform Resource Identifier)。服务器怎样解释这个请求-URI是与实现相关的，但是许多实现中可能用它来映射到一个类或者一个对象。一个SOAP请求也必须用SOAPMethodName HTTP头来指明将被调用的方法。简单地讲，SOAPMethodName头是被URI指定范围的应用的相关的方法名，它是用#符作为分隔符将方法名与URI分割开：SOAPMethodName: urn: schemas-develop-com:StockProcs #GetLastTradePrice。这个头表明方法名是GetLastTradePrice，范围URI是urn:schemas-develop-com:StockProcs。在SOAP中，规定方法名范围的名域URI在功能上等同于在DCOM或IIOP中规定方法名范围的接口ID。

下面是一个简单的SOAP方法请求：

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml
Content-Length: nnnn
SOAPMethodName: urn: schemas-develop-com: StockProcs#
GetLastTradePrice
<SOAP: Envelope xmlns: SOAP="urn: schemas-xmlsoap-org:
soap.v1">
  <SOAP:Body>
    <m:GetLastTradePrice xmlns:m=" urn:schemas-develop-
com:StockProcs ">
        <symbol>DIS</symbol>
    </m:GetLastTradePrice>
  </SOAP:Body>
</SOAP:Envelope>
```

上述代码的前5行是HTTP请求头。其中第1行包含3个组件：HTTP方法，请求-URI，协议版本。在前面的例子中，这些分别对应于POST, /StockQuote, 和HTTP/1.1。Internet工程任务组(IETF)已经标准化了数量固定的HTTP方法。GET是HTTP用来访问Web的

方法，POST 是建立应用程序的最常用的 HTTP 方法。和 GET 不一样，POST 允许任意数据从客户端发送到服务器端。请求 URI 是一个 HTTP 服务器端软件，它用来识别请求的目标的简单的标识符。请求的第 3 行和第 4 行指定了请求体的尺寸和类型。Content-Length 头指定了体信息的比特数。Content-Type 类型标识符指定 MIME 类型为体信息的语法。请求的第 5 行是个 SOAPMethodName 的 HTTP 头。从第 6 行到最后，是用 XML 格式封装的 SOAP 消息。在 SOAP 消息中，SOAP-MethodName 必须与<soap: Body>下的第一个子元素相匹配，否则调用将被拒绝。这允许防火墙管理员在不解析 XML 的情况下有效地过滤对一个具体方法的调用。易知，上述的 SOAP 消息表示调用网络服务端提供的 GetLastTradePrice 方法查询指定股票（本例中是 DIS）的最近一次交易价格。

SOAP 响应的格式类似于请求格式。响应体包含方法的[out]和 [in,out]参数，这个方法被编码为一个响应元素的子元素。这个元素的名字与请求的调用元素的名字相同，但以 Response 后缀来连接。下面是对前面的 SOAP 请求的 SOAP 响应：

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: nnnn
<SOAP: Envelope xmlns: SOAP="urn: schemas-xmlsoap-org: soap.v1">
  <SOAP:Body>
    <m:GetLastTradePriceResponse xmlns:m=" urn:schemas-develop-com:StockProcs ">
      <return>34.5</return>
    </m:GetLastTradePriceResponse>
  </SOAP:Body>
</SOAP:Envelope>
```

这里响应元素被命名为GetLastTradePriceResponse，它是方法名紧跟 Response 后缀。要注意的是这里是没有 SOAPMethodName HTTP 头的。这个头只在请求消息中需要，在响应消息中并不需要。

SOAP 中没有关于 SOAP 服务器怎样使用请求头来分发请求的要求；这被留为一个实现上的细节。一些SOAP服务器将映射请求-URIs到类名，并分派调用到静态方法或到在请求持续期内存活的类的实例，其它 SOAP 服务器则将请求-URIs 映射到始终存活的对象，经常是用查询字符串来编码一个用来定位在服务器进程中的对象关键字。还有一些其它的 SOAP 服务器用 HTTP cookies 来编码一个对象关键字，这个关键字可被用来在每次方法请求中恢复对象的状态。重要的是客户对这些区别并不知道。客户端软件只是简单遵循 HTTP 和 XML 的规则来形成 SOAP 请求，让服务器自由地以它认为最合适的方式来为请求服务。

## 5 一个 SOAP 调用的实例

利用 Apache-Soap 2.0 模型能够比较容易地建立网络应用程序和网络服务。它支持大部分的 SOAP1.1 规范。它有一个服务端框架来发布、管理和运行 SOAP 服务，并提供一系列客户端 API 来调用 SOAP 服务。以下实例就是基于 Apache-Soap 2.0。

### 5.1 环境配置

XML 编码解码需用到 SUN 的 Java 开发工具包 JDK，IBM 的 XML 解析器 XML4J 以及 Apache Xerces。

需将 xerces.jar,xml4j.jar 加到系统的 classpath 中。另外，也要将 SOAP.jar 加入到 classpath 中，它提供了一系列 SOAP 相关的类来调用 SOAP 服务。

客户端只需以上配置。服务器端需安装一个支持 servlets 和 jsp 的 web 服务器（可利用它发布网络服务并充当 SOAP 服务器）。本例选用 Tomcat3.1。

### 5.2 发布服务

依然用前面所述的例子，查询指定股票的最近一次交易价格。实现了具有相应功能的类之后，就可以发布这项服务，以便其他应用程序可以远程调用。Apache-SOAP 提供了两种方式发布服务：

通过浏览器或命令行工具。命令行方式如下：

```
java org.apache.soap.server. ServiceManagerClient url operation arguments
```

其中 url 是 Apache-SOAP rpcrouter（它监听到来的 SOAP 请求，解析 SOAP 消息）的 URL；

operation 是 deploy；arguments 则形如 deployment-descriptor-file.xml，该 XML 文件提供了服务的相关描述，形式如下：

```
<isd:service xmlns:isd="http://xml.apache.org/xml-soap/deployment"
  id=" urn:schemas-develop-com:StockProcs ">
  //此处 id 对应于 SOAP 请求的方法名的名域 URI。
  <isd: provider type="java" scope="Application" methods="getLastTradePrice">
  //methods 指定该 SOAP 请求可调用的方法名，若有多个，以空格隔开。
  <isd:java class="samples.stock.StockService"/>
  //class 指定所要调用的具体的类名。它对应了上面的 id（SOAP 请求的方法名的名域 URI）。
```

//发布服务后，Apache-SOAP2.0 会将具体类名以及相应的对象 id 注册在 ServiceManager 中。

</isd:provider>

</isd:service>

正在推出的 WSDL（Web Services Description Language）可用来描述已存在的网络服务，比如 SOAP 的 URL、对象的端点ID、提供的方法名、具体的调用消息格式等。它也是基于 XML 格式。通过查询相应的 WSDL，程序员可以很方便地将相应的服务集成到自己的程序中。

### 5.3 调用服务

知道了网络服务的接口信息，就可用下列客户端代码调用网络服务：

```
//建立一个 soap 的调用类
Call call = new Call ();
// 设置编码模式，利用标准的 SOAP 编码。
String encodingStyleURI = Constants.NS_URI_SOAP_ENC;
call.setEncodingStyleURI(encodingStyleURI);
//设置服务定位参数，即上述的URI，SOAP服务器根据它定位具体的类
call.setTargetObjectURI ("urn: schemas-develop-com: StockProcs");
//设置具体调用的方法
call.setMethodName ("getLastTradePrice");
//创建方法调用参数
Vector params = new Vector ();
params.addElement (new Parameter ("symbol", String.class, symbol, null));
call.setParams (params);
//激活服务，url 指向 SOAP 服务器的地址。
Response resp = call.invoke (url,"");
//在本例中 url 形如 http://soapListener:8080/soap/servlet/rperouter
//其中 Apache-SOAP 的 rpcrouter 监听到来的 SOAP 请求，解析 SOAP 信封信息，然后根据
//传过来的对象URI，在ServiceManager中查询相应的已注册的对象，并激活它。
if (resp.generatedFault ()) { throw new Exception();}
else {
    // 调用成功，取得结果，进行后续处理
    Parameter result = resp.getReturnValue ();
    Float Price=(Float) result.getValue(); …… }
    //异常处理
catch (Exception e) {System.out.println("Exception");}
```

这就是利用 Apache-SOAP 发布网络服务以及调用具体服务的大致过程。为了简单起见，这个例子中没涉及到诸如传递对象引用等问题。要实现对这类消息的正确的序列化以及反序列化，需用到前述的SOAP中的独立元素，这里就不再细述了。

## 6 结束语

网络服务为在Internet上绑定应用程序提供了一个利用现存体系架构和应用程序的简单的、灵活的、基于许多标准的模型。SOAP 是一个被类型化的序列化格式，它用 HTTP 作为请求/响应消息传输协议，并用 XML 定义消息格式。通过 SOAP，网络应用程序很容易与当地开发的服务或已存在的服务集成在一起，而不用考虑开发平台、开发语言或使用的对象模型。不过，与组件对象模型（诸如 CORBA 和 DCOM）相比，SOAP 在高级服务方面（如安全与事务）涉及较少。另外，如何有效地减少网络调用延迟，也是SOAP需要改进的一个方面。目前，SOAP 规范还在不断的完善中。SOAP的一个主要目标是使用尽可能多的存在的技术。几个主要的CORBA厂商已经承诺在他们的ORB产品中支持 SOAP 协议。微软也承诺在将来的 COM 版本中支持SOAP。可以预见，随着 Internet 的发展，SOAP 将以它的简单性以及平台无关性在网络分布式计算中发挥重要的作用。

## 参 考 文 献：


[1] Simple Object Access Protocol (SOAP) 1.1 http: //www.w3.org/TR/2000/NOTE-SOAP-20000508

[2] The Programmable Web: Web Services Provides Building Blocks for the Microsoft . NET Framework http: //msdn.microsoft.com/msdnmag/issues/0900/WebPlatform/WebPlatform.asp

[3] A Young Person's Guide to The Simple Object Access Protocol: SOAP Increases Interoperability Across Platforms and Languages http: //msdn.microsoft.com/library/periodic/period00/soap0300.htm

[4] The role played by XML in the next-generation Web http: //www.xml.com/pub/2000/09/06/distributed.html

[5] Web Services Description Language (WSDL) 1.0 http: //ww-4.ibm.com/software/developer/library/w-wsdl.html

[6] http: //xml.apache.org/soap/