第 6 章　Web Service 技术

本章首先介绍 Web Service 技术的基本概念，读者阅读之后将会对 Web Service 技术有大致的了解；然后介绍 Web Service 中的两个重要协议 WSDL 和 UDDI 的技术细节；最后一节会给出使用.NET 平台构造的 Web Service 例子。读者可以根据自己的知识结构和实际需要进行选择性的阅读。

6.1　什么是 Web Service

Web Service 是解决应用程序之间相互通信的一项技术。严格地说，Web Service 是描述一系列操作的接口。它使用标准的、规范的 XML 描述接口。这一描述中包括与服务进行交互所需要的全部细节，包括消息格式、传输协议和服务位置。而在对外的接口中隐藏了服务实现的细节，仅提供一系列可执行的操作，这些操作独立于软、硬件平台和编写服务所用的编程语言。Web Service 既可单独使用，也可同其他 Web Service 一起，实现复杂的业务功能。

抛开这些深奥复杂的定义，用一句话概括目前广泛使用的 Web Application 和 Web Service 的区别，那就是：Web Application 是面向用户的，而 Web Service 面向的则是计算机。面向计算机的特性赋予了 Web Service 巨大的潜力和广阔的应用空间，Web Service 也因此成为各大厂商追捧的对象。

让我们来看一个日常办公中的例子。某公司 A 通过因特网和公司网站提供了网上订购的业务，B 公司的业务员通过这个平台订购了 A 公司的一些产品，并将这些订购计划输入到 B 公司的信息系统中。于是这位业务员首先打开浏览器，进入 A 公司的网站，在网上填写订单并提交，然后将这份订单按照 B 公司所要求的格式重新填写一遍，再提交到本公司的信息系统中。同样的一份订单，虽然表现形式不一样，但所包含的信息是一样的，这位业务员为产生相同的信息做了两次工作。这种重复的工作不但意味着工作量的增加，也造成更多出错的可能。而计算机本身具有很强的数据处理能力，将这种样式转换的工作交给计算机是再合适不过了，但采用传统的 Web Application 平台却很难做到这一点。传统的 Web 面向的是用户，如果非要将 Web 页面的订单转换为信息系统中的格式则必须增加新的转换程序，而且对于不同的公司需要不同的转换程序，如果页面样式发生了变化，程序也要做相应的调整。如果采用 Web Service 这些问题都可

以迎刃而解，通过 Web Service 的一系列技术标准（如 WSDL、UDDI、SOAP 等），计算机可以自动地完成转换工作。Web Service 面向计算机和程序的特点可以让程序以更低的代价、更简单的方式集成到一起，降低企业实施电子商务的成本，同时 Web Service 的松散耦合方式也有助于以增量方式开发、部署分布式计算环境。

由于人们普遍认同了 Web Service 的技术潜力，促使 Web Service 技术不断发展。到目前为止支撑 Web Service 的基础标准工作已经基本完成，一些技术力量雄厚的公司也实现了 Web Service 供用户调用，如：google 就提供了一个检索的 Web Service。不过仍有很多 Web Service 标准如管理、安全、QoS、路由等仍在制订中。

6.2　Web Service 模型

在 Web Service 模型的解决方案中共有 3 种工作角色，其中服务提供者（服务器）和服务请求者（客户端）是必需的，服务注册中心是一个可选的角色。它们之间的交互和操作构成了 Web Service 的体系结构。服务提供者定义并实现 Web Service，然后将服务描述发布到服务请求者或服务注册中心；服务请求者使用查找操作从本地或服务注册中心检索服务描述，然后使用服务描述与服务提供者进行绑定并调用 Web Service。图 6-1 表示了 Web Service 模型的 3 种角色及它们之间的操作关系。

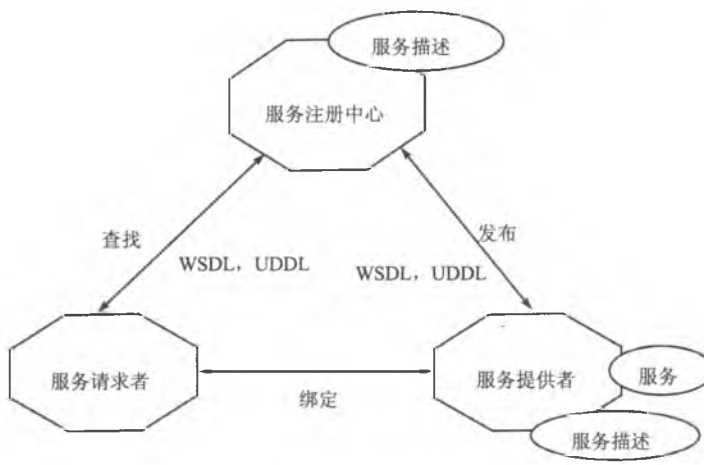

图 6-1　Web 服务角色、操作和构件

下面，我们分别介绍 Web Service 模型中的角色与操作。

（1）服务提供者。即 Web Service 的所有者，如企业、ICP 等。该角色负责定义并实现 Web Service，使用服务描述语言对 Web Service 进行详细、准确、规范的描述，并将该描述发布到服务注册中心供服务请求者查找并绑定使用。

（2）服务请求者。即 Web Service 的使用者。虽然 Web Service 面向的是程序，但程序最终的使用者仍然是企业或用户。从体系结构的角度看，服务请求者是查找、绑定并调用服务或与服务进行交互的应用程序。服务请求者角色可以由浏览器来担当，由人或程序（如另外一个 Web Service）来控制。

（3）服务注册中心。服务注册中心是连接服务提供者和服务请求者的纽带，服务提供者在此发布他们的服务描述，而服务请求者在服务注册中心查找他们需要的 Web Service。不过在某些情况下，服务注册中心是整个模型中的可选角色，如使用静态绑定的 Web Service，服务提供者可以把描述直接发送给服务请求者。在没有服务注册中心的 Web Service 中服务请求者可以从其他来源得到服务描述，例如文件、FTP 站点、Web 站点、广告和服务发现（Advertisement and Discovery of Services，ADS）或发现 Web 服务（Discovery of Web Services，DISCO）。

对于 Web 服务模型中的操作，包含以下三种：发布服务描述、查找服务描述、根据服务描述绑定或调用服务。这些操作可以单次或反复出现。

（1）发布。为了使用户能够访问 Web Service，服务提供者需要发布服务描述使服务请求者可以查找它。

（2）查找。在查找操作中，服务请求者直接检索服务描述或在服务注册中心查询所要求的服务类型。对于服务请求者，可能会在生命周期的两个不同阶段涉及查找操作，它们分别是：在设计阶段，为了程序开发而查找服务的接口描述；在运行阶段，为了调用而查找服务的位置描述。

（3）绑定。在绑定操作中，服务请求者使用服务描述中的绑定细节来定位、联系并调用服务，从而在运行时与服务进行交互。绑定可以分为动态绑定和静态绑定。在动态绑定中，服务请求者通过服务注册中心查找服务描述，并动态地同 Web Service 交互；在静态绑定中，服务请求者实际已经同服务提供者达成默契，通过本地文件或其他的方式直接同 Web Service 进行绑定。

6.3　Web Service 使用流程

目前，Internet 上已经有了不少 Web Service 可供使用，比如，google 提供了一个用于检索的 Web Service 免费供爱好者学习（每天可以使用 50 次）。那么该如何使用这些 Web Service 呢？图 6-2 所示是 W3C 所制订的 Web Service 使用流程标准。

在图 6-2 的每一道线中都标有一个数字，数字的大小代表了消息发生的先后顺序。在使用 Web Service 时，首先需要服务提供者将 Web Service 的描述信息提交到服务注册中心（即图 6-2 中所谓的发现服务中）。当服务请求者需要使用 Web Service 时，它首先通过发现服务查找需要的 Web Service，这就是图中的第二步。当找到合适的 Web Service 后，发现服务将返回请求者所需要的 Web Service 描述。在此之后，服务请求者

并不是马上同服务提供者进行 Web Service 的调用，而是首先需要同服务提供者统一各自的语义，以保证可以相互理解对方的请求和响应。当然，服务请求者可以按照服务提供者规定的语义信息进行服务调用，不过更合理的做法是双方遵循一个共同的行业标准，这个标准可以由一些相关的行业协会制订。当一切准备工作都完成后，服务者就可以直接同 Web Service 提供者进行交互，调用 Web Service。

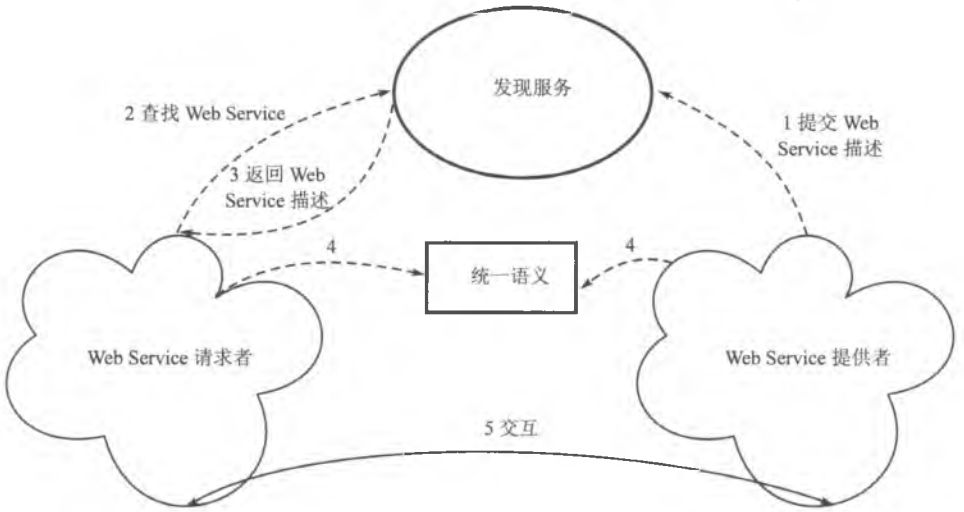

图 6-2　Web Service 使用流程

6.4　Web Service 协议堆栈

2001 年 4 月，W3C 联盟召开了第一次 Web Service 专题研讨会，其目的是为探索 W3C 应向哪个方向发展，才能更好地实现 Web Service 架构的标准化，会议期间第一次提出了“Web Service 堆栈”的构想。如今，随着 Web Service 技术的发展，2004 年 2 月 11 日，W3C 提出了最新的 Web Service 协议栈，其内容如图 6-3 所示。

在协议堆栈的下层为网络通信部分，Web Service 继承了 Web 的访问方式，使用 HTTP（S）作为网络传输的基础，除此之外 Web Service 还采用了其他的传输协议如 SMTP、FTP、JMS、IIOP 等。在消息处理方面，Web Service 使用了 SOAP 简单对象访问协议作为消息的传送标准。在此之上是 Web Service 描述语言 WSDL，用以描述 Web Service 的访问方法。位于最顶层的是与 Web Service 和应用程序，以及 Web Service 之间相互集成相关的协议，其中包含发现、集成等若干方面。在这一层，我们将介绍 UDDI 协议，UDDI 也是 Web Service 领域中赫赫有名的动态发现协议。除了底层的传输协议外，整个 Web Service 协议栈是以 XML 为基础的，XML 语义的精确性和灵活性赋予了 Web Service 强大的功能。除了这些基本协议，还有一些需要讨论的问题，那就是安全

和管理，这两大问题不是 Web Service 可以独立解决的。比如，在安全方面就需要同 PKI、LDAP 等相结合。

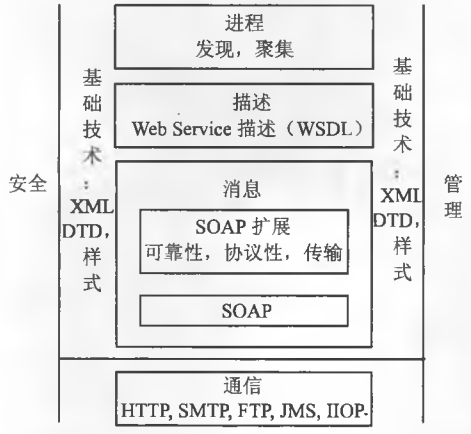

图 6-3　Web Service 协议栈

1．简单对象访问协议

W3C 于 2000 年 5 月 8 日发表了 Simple Object Access Protocol（SOAP，简单对象访问协议）1.1 版本，这是一种基于 XML 的协议，通过 SOAP，应用程序可以在网络中进行数据交换和远程调用。图 6-4 显示了 SOAP 在网络应用程序之间交换数据的方式。

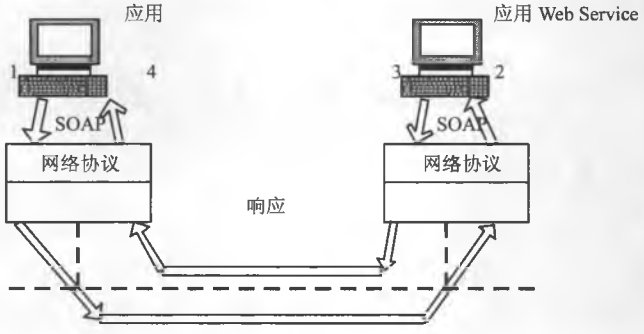

图 6-4　SOAP 工作模式

看到这里，有的读者会问：SOAP 实质上是一个基于 XML 的 RPC 标准，它与同为 RPC 标准的 CORBA、COM/DCOM 有什么区别呢？首先让我们再回顾一下 CORBA 和 COM/DCOM。

CORBA（Common Object Request Broker Architecture）公共对象请求代理体系结构是由 OMG 组织制订的，是一种标准的面向对象应用程序的体系规范。由对象请求代理 ORB（Object Request Broker）、对象服务、公共设施、域接口和应用接口这几个部分组成。其核心是对象请求代理 ORB，ORB 提供了一种机制，使对象可以透明地发出请求

和接收响应。分布的互操作对象可以利用 ORB 构造互操作应用。ORB 可看做是在对象之间建立客户/服务关系的中间件。基于 ORB，客户可以透明地调用服务对象提供的方法。该服务对象可以与客户运行在同一台机器上，也可以运行在其他机器上通过网络与客户进行交互。ORB 截取客户发送的请求，并负责在该软件总线上找到实现该请求的服务对象，然后完成参数、方法调用，并返回最终结果。

COM/DCOM（Component Object Model / Distributed Component Object Model）是微软公司提出的分布式组件对象模型标准，支持在局域网、广域网甚至 Internet 上不同计算机对象之间的通信。DCOM 基于 COM 的应用程序、组件、工具等来处理网络协议低层次的细节问题，因而本身不必关心太多的网络协议细节，使用户能够集中精力解决自身所要求的问题。DCOM 位于应用程序的组件之间，将组件以不可见的方式组合在一起，形成具有完整功能的应用程序。

明确了 CORBA 和 COM/DCOM 的概念以后马上可以明确的是 SOAP 同 CORBA、COM/DCOM 不是同一层面上的概念。SOAP 是一个基于 XML 的分布式对象通信协议，CORBA 是分布式应用的服务标准，COM/DCOM 则是一个组件模型。无论是 CORBA 还是 DCOM 都可以使用 SOAP 作为分布式对象通信的标准。除了概念上的区别外，SOAP 和 CORBA、COM/DCOM 还有更多的差异。

（1）采用 CORBA 或 COM/DCOM 构造的应用程序不能混用，二者不能协作。但 SOAP 却可以在二者之间建立联系，实现 CORBA 和 DCOM 的整合。

（2）SOAP 使用 XML 进行编码，是一个开放式的协议。SOAP 本身并没有定义信息的语义、服务质量、事务处理等问题，但 CORBA 和 DCOM 对这些问题都有相应的约定。

（3）SOAP 仅仅是一个对象通信协议，类似于 CORBA 中的 IIOP，是一个层次较低的协议。相比起来，CORBA 和 COM/DCOM 协议则复杂得多。

（4）SOAP 是与应用平台完全无关的。虽然 CORBA 也可以在各种平台上运行，但 CORBA 只能同采用 CORBA 标准的应用程序通信；而 COM/DCOM 则只能在微软的平台中应用。

我们可以将 SOAP 理解为：HTTP+XML+RPC。这里，HTTP 是网络中的通信协议，XML 是数据格式的协议。虽然将 SOAP 理解为 RPC 的一种并不准确，因为 SOAP 并非单纯的远程进程调用，SOAP 要强大得多，但以 RPC 的观点看待 SOAP 有助于我们理解 SOAP。

由于 SOAP 采用 XML 和 HTTP 封装通信消息，所以 SOAP 需要增加 XML 解析和 HTTP 传输的额外开销。但是 SOAP 同时也继承了 XML 和 HTTP 的优点，严格的语法格式使 XML 在 Internet 中得到了广泛应用。所以 SOAP 是一个非常优秀而且潜力巨大的协议，Web Service 正是一种建立在 SOAP 之上的服务模型。

2. Web Service 描述语言

我们都知道，在通常的开发过程中，对象接口一定具备相应的 SDK 描述文档，Web Service 也是一种对象，是一种被部署在 Web 上的对象。很自然，我们也完全需要有对 Web Service 这个对象进行描述的 SDK 文档。当然这两者不完全相同，一方面，目前在 Web 上的应用已经完全接受了 XML 这个基本的标准，WSDL（Web Service 描述语言，Web Service Description Language）是基于 XML 的标准，另一方面 Web Service 的目标是即时装配、松散耦合，以及自动集成，这意味着 SDK 描述文档应当具备被机器识别的能力。也就是说，对于使用标准化的消息格式和通信协议的 Web Service，它需要以某种结构化的方式（XML）对 Web Service 的调用和通信加以描述，实现这一点显然非常重要，这是 Web Service 即时装配的基本保证。WSDL 包含了一套基于 XML 的语法，将 Web Service 描述为能够进行消息交换的服务访问点的集合，从而满足这种需求。WSDL 定义了可被机器识别的 SDK 文档，同时 WSDL 也可用于描述自动执行应用程序在通信中所涉及的细节问题。

因为 WSDL 的目标是描述如何使用程序来调用 Web Service，所以我们可以把 WSDL 理解为 Web Service 的 SDK 标准，或者是 Web Service 的接口定义。对于服务提供者来说，他们既需要描述他们提供的 Web Service 是做什么的，还要描述如何使用他们提供的 Web Service。

3. 统一描述、发现和集成

UDDI（Universal Description Discovery and Integration，统一描述、发现和集成）提供了一种 Web Service 的发布、查找和定位方法。我们可以将 UDDI 理解为一种目录服务，Web Service 提供者使用 UDDI 将服务发布到服务注册中心，而 Web Service 使用者通过 UDDI 查找并定位服务。UDDI 除了目录服务之外，还定义了一个用 XML 表示的服务描述标准。在讨论了 SOAP 之后我们知道，通过 SOAP 和 XML 可以很方便地将不同企业的不同应用集成到一起，但仅仅有 SOAP 和 XML 仍然是不够的。SOAP 和 XML 不能够提供任何计算机平台都能支持的端到端的解决方案。UDDI 在 SOAP 和 XML 之上建立了新的层次，通过 UDDI 不同的企业可以以统一的标准描述自己的 Web Service，或查询其他的服务。

除使用 UDDI 发布和发现 Web Service 外，还有其他几种服务的发布方式，其中最简单的方式是直接发布。直接发布就是服务提供者直接将服务通过使用电子邮件附件、FTP 站点甚至光盘发布给服务请求者。直接发布一般是企业双方在使用电子商务的各项条款达成一致后进行，或在请求访问服务的服务请求者支付了费用之后进行。在这种情况下，服务请求者可以保留服务描述的一份本地副本。很明显，直接发布是一种静态的服务发布方式，灵活性很差。

直接发布动态的发布方法有 DISCO 和 ADS。DISCO 和 ADS 两者都定义了一个从给定 URL 获取 Web Service 描述的机制。然而这种简单的获取服务描述的方法也不能够完全满足动态的电子商务模式的要求，UDDI 的功能要强大得多。

UDDI 定义了一种 Web Service 的发布方式。首先 UDDI 注册中心可以为程序或程序员提供 Web Service 的位置和技术信息。服务提供者可以向专用的 UDDI 节点发布服务的描述信息，而服务的使用者可以动态地查询并连接到特定的 Web Service。我们可以将这几种服务发布技术放到坐标系中，如图 6-5 所示。纵坐标度量服务发布的能力，横坐标度量发布的灵活度。从图中我们可以看出，UDDI 的发布方式是功能最强大、灵活度最高的。

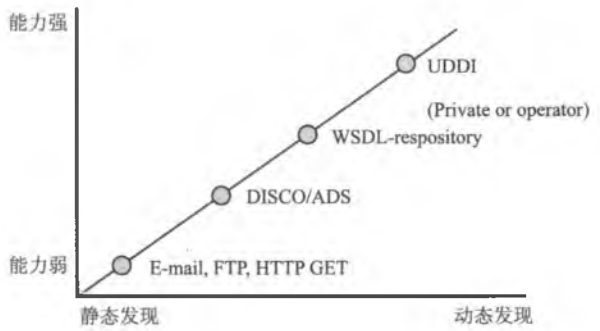

图 6-5　服务发布方式比较

6.5　XML 在 Web Service 中的应用

XML 在 Web Service 中有着非常重要的应用，可以毫不夸张地说，没有 XML 技术标准，就没有 Web Service 的出现。

Web Service 是一种部署在 Web 上的对象或组件，人们期望通过 Web Service 实现松散耦合的分布式组件互联，以适应 Internet 的计算环境。当讨论这种分布式互联策略时，遇到的最大问题就是如何将形态各异的数据结构、程序接口、操作系统、硬件平台有效地结合到一起。XML 恰好是解决这一问题的利器，由于 XML 具有严密的数据格式和灵活的表现方式，便于数据传输、转换和表现，因此 SOAP、UDDI 和 WSDL 都是在 XML 基础之上定义的。