Broadview
www.broadview.com.cn

# SOA实践

## ——构建基于Java Web服务和BPEL的企业级应用

余浩　朱成　丁鹏　编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

# SOA实践

## ——构建基于Java Web服务和BPEL的企业级应用

余浩　朱成　丁鹏　编著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

## 内 容 简 介

本书是一部以案例为中心来介绍 SOA 设计及开发的技术书籍。书中以实例说明如何设计和实现基于 SOA 的系统，以及如何解决 SOA 架构设计与实施过程中所遇到的实际问题，并讨论分析 SOA 带给系统的益处。

本书对 SOA 相关知识的讨论涵盖了面向服务的原理、关键协议与标准、设计与应用的全部过程。本书共分 8 章，第 1 章对 SOA 基本概念和原则进行了讲解，第 2 章介绍本书核心案例 SOAgent，第 3 章和第 4 章讲述面向服务的分析和设计过程，第 5 章针对 SOA 平台及相关技术进行介绍，第 6 章详细介绍 SOAgent 基本服务的实现与应用，第 7 章和第 8 章介绍 BPEL 技术。

本书的读者对象是有一定经验的软件开发人员，企业级信息系统架构师，SOA 项目设计及实施人员，广大 SOA 研究与爱好者，以及对 SOA 感兴趣的高年级计算机及相关专业的学生。



**图书在版编目（CIP）数据**

SOA 实践：构建基于 Java Web 服务和 BPEL 的企业级应用 / 余浩，朱成，丁鹏编著. —北京：电子工业出版社，2009.1
ISBN 978-7-121-07790-6

I. S… II. ①余… ②朱… ③丁… III. 互联网络－网络服务器 IV. TP368.5

中国版本图书馆 CIP 数据核字（2008）第 178487 号

责任编辑：高洪霞
印　　刷：北京智力达印刷有限公司
装　　订：北京中新伟业印刷有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×980　1/16　印张：20.25　字数：406 千字
印　　次：2009 年 1 月第 1 次印刷
印　　数：4000 册　定价：45.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

# 前　言

## 写作背景

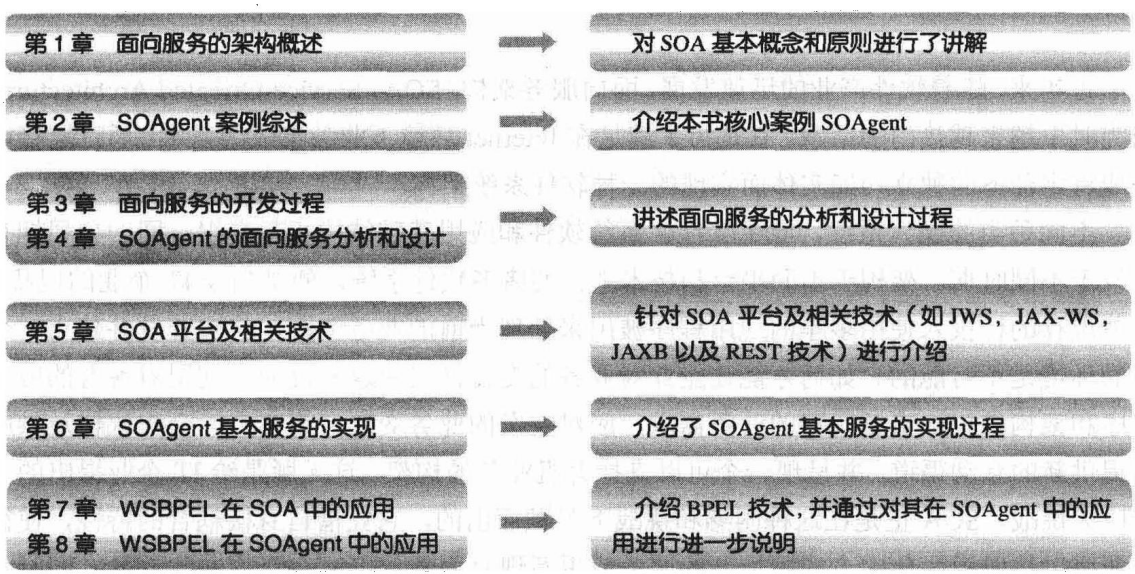
近年来，随着软件产业的迅速发展，面向服务架构（SOA，Service-Oriented Architecture）成为时下越来越热门的话题。它是为了满足在 Internet 环境下业务集成的需求，通过连接能完成特定任务的独立功能实体而实现的一种软件系统架构。

不同种类的操作系统、应用软件、系统软件和应用基础结构相互交织；同一公司拥有开发于不同时期、架构于不同平台和技术之上的诸多软件系统，便是当今 IT 企业的现状。一些现存的已投入使用多年的应用程序被用来处理当前的业务流程，从头开发并建立一个新的环境是不可能的。如何才能让企业对业务的变化做出快速的反应，利用对现有的应用程序和架构的投资来解决新的业务需求，应对突发的业务变化，为客户、合作伙伴及供应商提供新的互动渠道，并呈现一个可以支持有机业务的构架，这无疑是给 IT 企业提出的一个巨大挑战。SOA 正是在这种困惑和挑战下脱颖而出的，它凭借自身松耦合的特性，使得企业可以按照模块化的方式添加新的服务或更新现有服务，以解决新的业务需要，同时它还提供选择，从而可以通过不同的渠道提供服务，并可以把企业现有的或已有的应用作为服务，从而增加了原有系统的可用性，进一步增强了企业的服务效能。

## 本书特点

本书从一个具体的案例 SOAgent 出发，阐述如何从无到有构建一个基于 SOA 的系统平台，并通过将它和传统开发过程相比较，讨论 SOA 及具体相关技术为企业提供的解决方案。本书的案例描述了一个在 Web 2.0 环境下逐步兴起的商业模式，即商品的网络搜索、直销和配送。SOAgent 将这几个流程进行整合，通过对 Amazon，Yahoo 等大型门户销售网站所提供的商品检索和销售 Web Service 的封装集成，让用户可以方便地通过 SOAgent 搜索并购买其他网站的产品。同时，SOAgent 也为其他门户网站提供搜索接口，以使得这些

门户网站可以整合 SOAgent 的搜索及销售接口。双方都可以因此获利，SOAgent 借以推广其销售渠道，第三方门户网站则可提供更多的增值服务。在配送方面，SOAgent 把原先的配送系统撤销，取而代之的是使用并包装 FedEx，DHL 及 UPS 等提供的服务。为了优化业务整合和应对需求变化，引入 BPEL 对业务流程进行描述并配置，从而在不改变流程及服务实现的前提下，仅通过对 BPEL 的修改便实现业务流程的重组。




## 致谢

本书的编写得到了中国电子工业出版社和德国企业经济信息研究所的大力支持，在编写的过程中，我们也得到了很多人的帮助和鼓励，在此要特别感谢 Steffens 教授以及张子頔女士。同时，感谢中国电子工业出版社的李冰编辑和高洪霞编辑的支持。对于本书中可能存在的错误、问题以及疏漏之处，敬请广大读者批评指正。

余浩，朱成

2008 年 9 月

# 目　录

# CHAPTER 1

# 面向服务的架构概述

## 1.1 什么是服务（Service）

在开始任何工作之前，需要有一个共同的基础。这个共同的基础就是对一些基本概念的统一定义，没有这些定义，任何讨论都失去意义。在了解面向服务的架构时，同样如此。

服务是面向服务的架构中的核心概念，不理解服务的概念，就无法理解面向服务的架构，所以首先需要定义服务的概念。但是目前为止对服务这个概念没有一个统一的定义，不同的组织机构对它有不同的理解。下面列出服务的一些定义。

W3C（World Wide Web Consortium）将服务定义为："服务提供者完成一组工作，为服务使用者交付所需的最终结果。最终结果通常会使使用者的状态发生变化，但也可能使提供者的状态改变，或者双方都产生变化"。该定义给出了服务涉及的双方，即服务提供者和服务使用者。

OASIS（Organization for the Advancement of Structured Information Standards）则将服务定义为一种访问某一个或多个能力的机制，这种访问使用预先定义好的接口，并与该服务描述的约束和策略一致。该定义给出了服务的重要元素，即接口、约束和策略。

在 Wikipedia 中，服务是指自包含、无状态的业务功能，通过良好定义的标准接口，接受多方的请求，并返回一个或多个响应。服务不应该依赖于其他的服务，并与使用的技术无关。该定义给出了服务的重要特征，即自包含和无状态。

以上三个定义都试图从抽象的角度给出服务的定义，但都只是描述了服务的某一方面，并没有全面地刻画出服务的特点。为了更深刻地了解服务的概念，需要了解服务的特点。一般来说，服务具有以下 9 个特点：

（1）服务是可重用的。

（2）服务都有服务合同。

（3）服务之间是松耦合的。

（4）服务隐藏了具体的逻辑。

（5）服务是可组合的。

（6）服务是自治的。

（7）服务是无状态的。

（8）服务是可被发现的。

（9）服务一般是粗粒度的。

以下将逐一介绍服务的这些特点。

### 1.1.1 服务是可重用的

可重用性指的是，为某一目的而实现的模块也能被用于其他的目的。具体到服务这个层面，就是服务所包含的逻辑不仅可以被用于最初设计时的目的，也可以被用于其他的目的，即服务可以出现在不同的应用中，它是上下文无关的。

可重用性也是软件业一直追求的目标。回顾软件发展的历程，在面向过程的开发方法中，重用是通过函数来实现的，这些函数实现了一些通用的功能。函数一般被组合为函数库，这些函数库可以被用在不同的系统中，从而实现重用。在面向对象的开发方法中，重用的基本单位为对象。对象封装了数据和对这些数据操作的方法，因此实现了一个更高层次的抽象。特别是对象之间组合形成的框架，不仅实现了功能上的重用，还实现了对象之间的关系的重用。但是，面向过程和面向对象的重用都局限于具体的实现语言，因此它们都不能实现跨语言的重用。随着组件的概念及基于组件的开发模式的提出，在跨语言重用方面得到突破。组件可以跨越具体的编程语言和编程平台实现一定程度上的重用。但是组件的概念更偏向于技术，一般粒度都很小，不能直接满足企业业务的需求，随后便出现了服务的概念，它可以和企业业务直接对齐，提供粗粒度的功能，其底层实现可以通过组件来完成。从软件的发展历程可以看出，抽象的程度一直在提高，直至和企业业务对齐。显然，服务提供了一个更高层次的重用。

服务的可重用性是服务的很重要的一个特点，它能带来很多好处。首先，最直接的好处就是降低了开发成本，提高了投资回报率。企业在开发新的应用时，可以先检查是否现有的服务部分或全部满足新的应用需求，如果存在这样的服务，则不需要进行该部分的开发，从而节省了成本。其次，由于利用了现有的服务，开发时间也相应缩短。此外，服务的重用也提高了系统的质量，因为之前开发过的服务已经经过多次的测试和实践的检验，确认是可靠的服务。如果新的应用都是通过现有的服务组合而成的，显然，其质量能够得到保证。

服务的可重用性在服务设计阶段要重点加以注意，它给设计带来了新的挑战。在设计服务时，不仅要考虑到现有的应用，还要把眼光放长远，考虑到将来可能出现的新应用，使得设计出的服务具有一般通用性。

服务的可重用性也使得创建一个中心服务库成为必要，在中心服务库中可以储存企业现有的所有服务。在企业需要新的应用时，可以先检查该中心服务库，看是否存在现有的服务部分或全部满足企业新的应用需求。

### 1.1.2　服务都有服务合同

服务合同指的是服务消费者和服务提供者之间的约定。服务消费者和服务提供者必须

遵守该约定，服务才能顺利进行，并得到双方预期的结果。如果任何一方违反该约定，得到的结果则是不可预期的。显然，服务合同是服务提供者和服务消费者之间交互的前提，没有服务合同，服务双方的交互根本不可能发生。此外，服务合同也必须是完整的，即服务的消费者和服务的提供者只需要服务合同，而不需要其他额外的信息，就可以进行交互。

服务合同一般由服务提供者提供，作为对它所提供的服务的描述。服务的消费者在消费该服务前，必须理解该服务合同，并按照服务合同的要求与服务提供者交互，才可以使双方最终得到预期的结果。

服务合同在计算机行业里并不是新鲜的名词，它的历史可以追溯到早期的 Application Programming Interface (API)。API 提供了一种简单的技术层面的服务合同，描述了服务的功能方面的要求，规定了服务的名字、调用该服务所需的参数个数和类型，以及返回参数的类型。服务的消费者如果遵守该服务合同，提供正确的参数个数和类型，就会得到预期的返回值。

在面向服务的架构的范畴内，服务合同有着更高的抽象层次，它不仅包含了对技术层面的要求，也包括了对服务的非功能性层面（比如性能，可靠性等）的要求，甚至还包含了语义方面的信息。在采用 Web 服务技术来实现服务时，服务合同一般包括以下 4 部分。

（1）WSDL 文档：用于描述服务的端点、服务提供的操作，以及每一个操作的输入和输出信息。

（2）XML Schema：定义交互的数据类型。

（3）WS-Policy 文档：描述没有在 WSDL 中定义的服务元信息。

（4）Service Level Agreement(SLA)：描述服务的非功能性层面的要求和语义信息等。

在设计服务合同的时候要特别小心。首先，服务合同必须提供服务交互所需的全部信息。其次，由于服务合同代表了服务消费者对服务提供者的一种依赖，服务提供者必须根据服务合同提供服务。在服务升级的时候，服务提供者必须保证能够继续按照原有的服务合同提供服务，同时给新的服务消费者按照新的服务合同提供服务。这里需要对服务合同进行很好的维护和版本管理。

### 1.1.3　服务之间是松耦合的

耦合是指两个事物之间的依赖关系。如果一个事物和另外一个事物依赖很紧密，就称为紧耦合，反之则称为松耦合。耦合这个概念一般和内聚联系在一起。内聚指的是一个事物内部各部分之间的关联程度。联系紧密称为高内聚，反之则称为低内聚。一般来说，高内聚对应着松耦合，低内聚则对应紧耦合。

在计算机领域，耦合和内聚都属于衡量软件质量的重要标准。高质量的软件要求模块是高内聚和松耦合的。比如在面向对象的方法中，类的设计准则即是如此。对类的要求为高内聚，即一个类只针对一项任务，而不是分别完成许多不相关的任务。而类之间则要求松耦合，类只暴露必要的信息，满足类之间的通信要求即可。

服务的松耦合带来的好处是不言而喻的。如果两个服务之间是松耦合的，那么它们之间的联系很弱，这样服务就可以相对比较容易地变化，而不会对其他相关的服务带来太大的影响。如果服务之间联系太过紧密，则容易出现牵一发而动全身，从而导致一个服务的变化引起多数甚至全部服务的变化，从而使变化实际上难以实施，因为变化的后果是不可预计的。

面向对象的方法中虽然通过类的运用实现了一定程度上的松耦合，但是它的一个主要特征，即继承，在一定程度上造成了子类和父类之间的紧耦合，父类的改动可能对子类造成不可预计的影响。考虑例程 1-1 所示的例子 [引用 3]。

**例程 1-1　在 Java 中用 ArrayList 实现栈**

```
class Stack extends ArrayList
{ private int stack_pointer = 0;
public void push( Object article )
{ add( stack_pointer++, article );
}
public Object pop()
{ return remove( --stack_pointer );
 }
public void push_many( Object[] articles )
{ for( int i = 0; i < articles.length; ++i )
push( articles[i] );
```

```
    }
}
```

看上去似乎一切都没有问题，但是考虑到如果用户利用继承直接调用 ArrayList 的 clear 方法，所有的元素都会被清空，由于基类完全不知栈指针的问题，所以导致该栈对象处于一种未知的状态。由此可见继承所导致的紧耦合的问题。鉴于此，服务完全抛弃了继承的关系。

服务之间的松耦合是通过上文介绍的服务合同来实现的，即服务之间只依赖于服务合同。服务合同从一个比较抽象的层次描述了服务的接口，而并没有暴露任何关于服务内部实现逻辑的信息，因此服务逻辑的实现可以随意改变。只要服务合同保持不变，服务逻辑实现的改变对服务之间的交互没有影响。

### 1.1.4 服务隐藏了具体的逻辑

上文已经提到，服务之间是松耦合的，即服务之间只依赖于服务合同。服务合同只提供服务之间交互所需的必要信息，而不能暴露任何关于服务内部逻辑实现的信息，这样服务就像一个黑盒子一样，隐藏了具体的逻辑，外界只能看到服务合同。

### 1.1.5 服务是可组合的

组合指的是两个或多个事物相互合作，构成一个更复杂的事物，这个更复杂的事物一般具有更强大的功能，能提供单个事物不能提供的功能。20 世纪 80 年代有一部很受欢迎的美国动画片“变形金刚”，片中分为正邪两派，分别为汽车人和霸天虎。它们作为个体都是具有强大的功能，但是更厉害之处在于，它们可以全体组合，从而构成一个无敌的巨无霸。汽车人组合成的大力金刚比它们作为个体的威力更大。这里当然要有一个前提，那就是这些汽车人是可以组合的，在设计时就要考虑到这个因素。

同样对服务而言，可组合性也是一个很重要的特点。通过服务之间的组合，可以构建新的服务，这个新的服务能完成单个服务不能完成的任务，满足用户特定的需求。同时通过服务的组合，也得到不同粒度的服务。基本服务粒度相对比较细，组合得到的服务则为粗粒度的服务。如图 1-1 所示，服务 1，2，3 组合得到一个组合的服务。

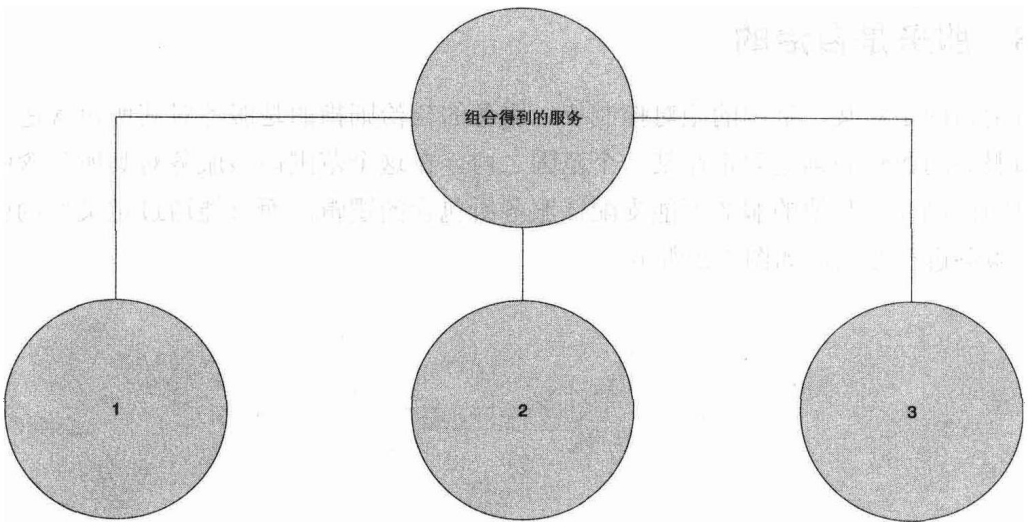



图 1-1　服务的组合

在面向对象的方法中，对象之间的组合是通过关系、aggregation 及 composition 来实现的，使用 UML 可以对这些不同的组合建模。在具体实现时，组合是通过编程来实现的，这样得到的系统有一个致命的缺点，它的对象组合的代码和对象实现的代码是混在一起的，不能分开。如果新的需求要求的仅仅是组合代码的改变，它将导致整个代码的变动。在面向服务的架构中，则避免了这一点。服务的组合是使用一种标准的语言来描述，和服务的实现分开，这样在仅仅需要组合的逻辑发生改变时，对服务并没有任何影响，从而减少了工作量，降低了开发成本，实现了对服务最大程度的使用。

在面向服务的架构中主要有两种形式的组合，即 Orchestration 和 Chereography。二者的主要区别在于可执行性和控制模式。Orchestration 定义了可执行的流程，流程和外界信息交换的顺序由 Orchestration 来集中控制。Chereography 则只是点对点交换的协议，仅仅定义了合法的信息交换顺序，进行信息交换的双方是平等的，并不存在一个进行集中控制的服务。通过这种方式定义的流程一般是不可执行的，而是存在多种实现，只要它们满足定义好的合法的信息交换顺序就可以。Chereography 当然可以用 Orchestration 来实现。二者都有对应的规范，Orchestration 为 Web Services Business Process Execution Language（WSBPEL）规范，Chereography 则为 Web Services Chereography Description Language（WSCDL）规范。

### 1.1.6 服务是自治的

自治指的是对某一范围的绝对控制权。服务的自治则指的是服务对其所包含逻辑的控制，即服务的逻辑必须被限定在某一个范围之内，在这个范围内该服务对其所包含的逻辑有绝对的控制权。其他的服务不能支配该服务所包含的逻辑，而只能通过定义好的服务合同与该服务进行交互。如图 1-2 所示。

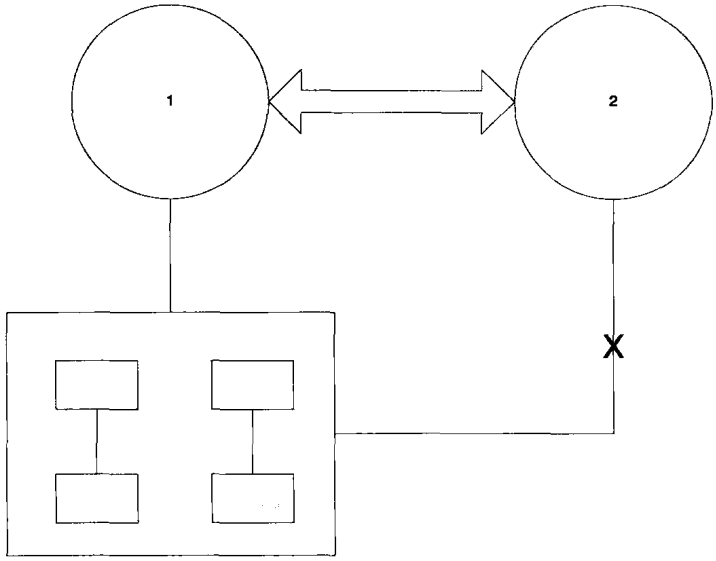



图 1-2 服务的自治

服务的自治性强调了服务对所包含逻辑的排他占有性，其他服务不能支配该逻辑，但是这并不排除其他的不属于服务范畴的事物对该逻辑的访问和支配。考虑企业的历史遗留系统，在实现面向服务的架构的时候，根据需要，遗留系统的部分或全部功能被包装为服务，这样在服务这个层次上，服务是自治的，即其他服务不能支配该服务的逻辑，但是这并不排除这些遗留系统原有的客户端对这些逻辑的访问，只不过它们不属于服务的范畴。所以服务的自治一般分为两种，一种为服务级别的自治，即从服务的层次上看，服务是自治的。另一种为纯粹的服务自治，新构建的服务一般为这种。此外也可以看出，服务级别的自治存在风险，因为服务不是唯一可以访问逻辑的，这样可能会导致冲突。在面向服务的架构中尤其要加以注意。企业的遗留系统是企业的资源和财富，在实施面向服务的架构