SOA 设计模式汇总

SOA 设计模式汇总

本专题分六部分探讨 SOA 设计模式，当初设计面向服务架构的一大初衷就是降低服务间耦合度，由此提高服务的灵活性和自由度，这样一来每个服务都可以不受羁绊，更好的得到发展。实现理想的松耦合程度，一直是设计中讨论的议题，议题通常是围绕服务合同和依靠服务合同的用户编程展开的。

Façade 服务

Façade 服务模式的实施过程中，还有一种情况可以得到有效控制。核心服务逻辑的单个实体要求多个合同（这种情形和另一个叫做 Concurrent Contracts 的模式有关）。

- ❖ 本周 SOA 模式：Façade 服务

可知环境

如果一个服务是可重用服务的话，还应该称其为服务吗？就这一模式来说，答案是否定的。在服务建模和设计阶段，不可知服务受到了广泛的关注。

- ❖ 本周 SOA 模式：可知环境

域库存

企业范围的协调性一直是人们多年来不断追求和努力的目标，这种状态全力支持 SOA 以及面向服务架构所包含的一切内容。

❖ 本周 SOA 模式：域库存

服务规范化

当你设计数据架构时，会很容易得出不同的数据库或者数据库表。该数据库表中包含相同或相近的数据。所有这些记录在册的数据可以帮助更好的维护数据，解决质量问题。

❖ 本周 SOA 模式：服务规范化

服务分解

一个服务本身就是一个服务生命体，需要独立发展。我们在编制 SOA 设计模式目录时，了解到不仅在设计阶段，甚至是服务生命周期的后续维护阶段也会涌现许多模式。

❖ 本周 SOA 模式：服务分解

典型模式

SOA 设计模式目录中，没有什么比典型模式更容易理解，也没有什么比典型模式更难实践了。此外还有一些富有争议性的模式。

❖ 本周 SOA 模式：典型模式

本周 SOA 模式：Façade 服务

当初设计面向服务架构的一大初衷就是降低服务间耦合度，由此提高服务的灵活性和自由度，这样一来每个服务都可以不受羁绊，更好的得到发展。实现理想的松耦合程度，一直是设计中讨论的议题，议题通常是围绕服务合同和依靠服务合同的用户编程展开的。

对于服务设计师来说，在服务实现内部还是有机会建立抽取中间层的，这个过渡层可以帮助减小其内部活动部分的耦合程度以便调节服务本身的演进和治理。这些中间抽取层是由 Façade 服务应用所创建的，该设计模式关注服务的内在设计。

设计服务时，你还要留心几个耦合类型：合同与逻辑耦合，例如，该耦合会生成一个服务合同，而服务合同源自底层服务逻辑（因此会产生依赖性），一旦逻辑发生改变，合同也会相应做出改变。

通过在核心服务逻辑和服务合同之间创建中间处理层，Façade 服务就可以帮助我们避免这些情况的发生。Façade 准许服务合同与底层逻辑保持去耦关系，因此可以帮助避免修改核心业务逻辑。这对功能改变和性能改变都适用。性能变化可能是因为实施服务重构模式所引起的（通常是偶然的）。Façade 会对内部改变做出补偿，这样就没必要改变服务合同了。因为合同中的功能性能没有受到影响，在两种情况下，服务会在内部得到演进，同时现存的服务客户也会免于任何负面效益的影响。

该模式的实施过程中，还有一种情况可以得到有效控制。核心服务逻辑的单个实体要求多个合同（这种情形和另一个叫做 Concurrent Contracts 的模式有关）。在这种情况下，对应每个服务合同都会创建一个 Façade 服务组件，这样可以保证从合同层中抽取出核心服务逻辑。这样就可以避免为了适应不同的合同增加或者扩大核心服务逻辑，并且当

逻辑收到功能变化和影响或者受制于性能补偿时，还能进一步利用前面所提到的功能收益和性能补偿。

Façade 服务模式会产生一个带有明确抽取层的完美服务架构，但是随着服务逻辑物理分配的增加，在这之前还要进行额外的数据处理。服务逻辑分散性的增强也可以看做是服务设计的复杂性在不断增加。切记，处于 façade 层的逻辑可以帮助减轻这些问题所带来的影响。

总体来说，尽管该服务模式的实施会使服务内部的数据处理职责得到有效分离，由此所得到的庞大管理收益能够适应服务长期发展，并巩固与服务客户计划的关系。

本周 SOA 模式所包含的原创内容和独家观点全部出自 SOAPatterns.org 社区网站以及“SOA 设计模式”(Erletal.，国际准书号：0136135161，Prentice Hall 出版社，2009)一书的所有作者和撰稿人的辛勤努力，该书最近的标题叫做“Prentice Hall Service-Oriented Computing Series from Thomas Erl(www.soabooks.com)2009 版权所有 SOA 系统公司。

(作者：Thomas Erl 译者：杨君 来源：TechTarget 中国)

本周 SOA 模式：可知环境

如果一个服务是可重用服务的话，还应该称其为服务吗？就这一模式来说，答案是否定的。在服务建模和设计阶段，不可知服务（具有可重用特性，并且每个 Agnostic Context 都可以提供多功能逻辑）受到了广泛的关注。但是任何只关注不可知服务逻辑的做法都是不完全的。

可知逻辑代表既定业务流程或者任务的所有功能。换句话说，可知逻辑本身是单一功能逻辑因此不能重用。可知逻辑最为普遍的形式和其它服务的组合有关。

当把众多服务合成为一体时，业务决策驱动功能的原始层通常会创建业务任务的工作流。因为这种类型的功能通常是针对某个具体的任务，因为被认为具有单一功能（或者可知），并进一步被分成独立的任务服务（由 Process Abstraction 相关模式创建的服务模型）。

在大多数服务定向解决方案中，可知逻辑需要同不可知逻辑分开，但是为什么还要将其置入服务中呢？简单来说“不需要将其置入服务中。”“你可以把这个原始逻辑置入一个丰富客户，单片机系统当中，或者其它服务定向软件程序中，服务组合依然可以实现。Non-Agnostic Context 不会要求你创建服务，只会告诉你这样做的好处。

最近单一功能服务：理解 Non-Agnostic Context 及其实施策略这篇文章中刊登这些好处。文章作者：Herbjorn Wilhelmsen。在这里，我们并没有重复只是简要列举其中的一些好处：

- 众多供应商

- 不受变化影响
- 治理集中(把所有这些放在一起)
- 最佳服务组合
- 分担用户计算机的任务
- 业务调整(关注点分离)

为了实现服务组合最佳化，现在我们简要看一下上述第四条。

服务定向将组合这个概念提升到了一个新的高度。它不仅关注如何将分散的软件程序成功的组合成自含程序，同时还关注灵活和自适应组合。这意味着在设计服务时不仅要考虑到服务需要参与到大型组合中去，同时还有让其能增加扩展并重新配置现有组合，并且不管业务需求发生怎样的变化，还要参与到新的组合中去。

有效重组这些服务（例如每个 Capability Recomposition 模式）取决于服务定向在机构层面最大限度启动业务灵活性的能力。服务定向的设计原则可以通过将服务塑造成高效组合成员实现这一目标。

还有一种方法是从组合的角度来看待这个问题。一个服务组合就是一个机械实体，这个机械实体由分散移动部分构成，这些分散移动部分需要在运行时间合力工作，以便执行主要业务任务。那些无法通过服务定向严密考验的那部分组合会成为整个组合质量和性能的薄弱环节。

在非服务定向程序和平台中置入可知逻辑会降低服务组合的效用和潜能，有时会令这些效用和潜能大打折扣。较为大型的、复杂组合更是如此，这些组合服务内部依赖相互的行为要求，并且服务性能要求也得到了扩大。

Non-Agnostic Context 模式可以解决这几个要求，因此单一功能逻辑可以被封装到服务中。这些服务得到调整和最佳化，以便参与现有的服务组合，以便帮助它们进一步改良。

（作者：Thomas Erl 译者：杨君 来源：TechTarget 中国）

本周 SOA 模式：域库存

企业范围的协调性一直是人们多年来不断追求和努力的目标，这种状态全力支持 SOA 以及面向服务架构所包含的一切内容。那些已经实现这一目标的机构，可以更好的实现标准化。这些机构所取得的成就，使其它机构的成果显得逊色了许多。但是，没有实现协调性，并不意味着你无法成功实施 SOA。

在某些领域，很多人认为，SOA 措施是孤注一掷的，它要求在整个企业范围内全部转换成 SOA 模式。对于那些抱有这种想法的人来说，一想到要顺应全球数据模型，全部都屏住了呼吸，那些 IT 管理者一想到要将权利下放给其下属部门，就寝食难安，那些叛逆的开发商整日被标准监督人员（配有符合行业标准的防爆装备）所围绕。

有些人被误导，将 SOA 项目和“大爆炸”方法联系在一起。这给 SOA 带来了务虚有的罪名，这和面向服务计算这一哲学思想大相径庭。你不必为了获得收益，就在整个企业范围内实施 SOA。正是出于这个原因，域库存模式诞生了。从战略规划的角度来说，没有什么模式能够像域库存模式一样，完全实现 SOA。

在面向服务架构的过度时期，你将负责决策 SOA 的使用范围。只要在跨筒仓领域是有意义的，就能在一定程度上实现与服务定向计算相关的策略效益。换句话说：你无需在整个企业范围内采用 SOA 来获得收益。你可以创建一个能够代表 IT 企业部门的域。这个部门要容易管理。并在域内实施 SOA。机构中的另一个小组可以创建自己的域，并独立的实施 SOA 项目。

基本概念构成了该模式的基础。同时还可以在企业范围内提供一个真实可靠的解决方案。

在 SOA 设计模式目录中，这些域以服务库存的方式展现出来。服务库存是在现实世界中实施你在服务目录或者服务组合中所记录的一切。在实施域库存模式时，每一个模式都按照其特有的方式被定义。实际上，很多大型机构也设计出了许多域库存。

在最近 InformIT OnSOA 频道的播客访谈节目中，分析师 Joe McKendrick 问到在创建“服务群岛”时，域库存模式和其它模式有什么不同之处，。换句话说采用这种模式，会不会使我们倒退回基于部门划分的、集成应用环境，这种环境令我们在一开始就陷入困境。答案是否定的。该模式是要建立“服务大陆”每个服务库存的范围是通过具有战略意义的适当平衡和易管理治理而定义的。

标准警察叫道“等一下”“要是不同域的服务之间需要交流，不还是有集成的需要吗？”是的长官，情况的确如此，当跨域服务需要组合时，仍然需要使用传统集成技术。例如，数据模型转换，协议桥接，以及数据格式转换技术。但是在项目的最初阶段就可以实施这个模式，这些转换和桥接层被认为是企业技术蓝图既定的一部分。该技术蓝图主要是围绕域库存而进行的。因为这些技术都在意料之中，他们的作用和影响也可以提前预测。当集成要求在解决方案交付后，意外出现时，就可以有效的缓和了他们的需求。

这里有许多定义服务域组成的方式。这个方式可以是建立在权利域基础上，也可以建立在地理位置基础上，或者大型的旧环境基础之上。其中最“健康”的方法就是将域库存边界和真正的业务域紧密的连在一起。面向服务技术企业的宏观视图可以和逻辑域或者可能存在的业务领域同步。这样可以便利长期定位调整业务技术。

此外还有许多实施与库存模式的方法。例如，域服务库存可以将 SOA 实施划分为几个阶段。同时，在最初定义后，还有机会扩大个人域，有利于发展增加新的服务。

不论是用那种方法，关键是要了解这个模式所展示的选项。因为这样可以帮助所有机构采用 SOA，这也是成功实施面向服务计算商业价值的最佳方法。

(作者：Thomas Erl 译者：杨君 来源：TechTarget 中国)

本周 SOA 模式：服务规范化

当你设计数据架构时，会很容易得出不同的数据库或者数据库表。该数据库表中包含相同或相近的数据。所有这些记录在册的数据可以帮助更好的维护数据，解决质量问题。这样建立起来的数据规范成为了最受欢迎的数据建模实施。从本质上来说，数据规范化的目标就是尽可能降低数据的赘余度。这样就迫使每一个使用特定数据类型的应用必须通过一个位置访问服务规范。因此，通过降低数据的冗余度，数据规范可以促进时间重用。

可重用性也是服务定向的主要目标之一。服务定向的其中一项原则就是确保服务的质量得以实现。服务规范是众多支持服务可重用性模式中其中的一种模式，但是，服务规范的目标远不止支持服务可重用性。同数据规范一样，服务规范模式意在降低冗余度，减少不必要的消耗，以避免治理负担（维护相似服务逻辑体，并使相似服务逻辑体保持同步，同时复制服务逻辑）。

要想实现这个目标，服务规范首先要在服务之间划定界限，这样服务就不会重叠。和数据规范不同，服务规范并没有受限于数据。服务规范主要考虑的是基本服务界限的规范化。因此，你经常会感觉自己在服务建模时期正在实施这种模式，这个时期也是服务最初概念化的时期。

要想理解服务规范实施，最重要的一方面就是要弄清楚开展规范化任务的范围。正如我们在这一系列文章中所提到的，通过网域清单，可以在一个 IT 企业里建立许多由独立标准服务和管理服务所组成的集合。这些服务清单（有时被称为“服务大陆”）准许你实现服务定向目标。

服务清单蓝图通常都是在分析和建模期间确定下来的，一个特定蓝图的界限决定了服务规范的实施范围。这意味着，只要是在域服务清单范围内发生（而不是某个特定的服务清单）你就可以覆盖服务界限和冗余的服务逻辑。

这个由服务规范确立的原则融入了服务建模流程以及全局服务交付方法。避免功能覆盖是我们需要考虑的首要问题，也是专用流程步骤的基础。（尤其是迭代实施的建模流程）。当许多不同的小组平行工作，为相同的服务清单建立服务时，需要跟踪并协调这个问题。

尽管我们采取了多种措施，还是无法避免所有的功能重叠。服务蓝图以及具有相同功能的服务就有可能丢失某些东西。或者，硬约束限制了这种模式的完全应用。例如，不同的服务需要封装那些自身无法规范化的。在这种情况下，嵌入式逻辑或者确立的逻辑不可避免的会导致一定程度的冗余度。这时就出现了性能问题。你可能会遇到一种情况，就是完全规范化的服务会使运行的等待时间延长，解决这个 问题的唯一方法就是给服务设计一些非规范化方法。

在现实世界中，你可以这样解释这个模式“在一个服务清单中，没有两个服务界限可以重叠，如果二者发生了重叠，必须要有一个足够的理由！”，这也是服务规范化的总体目标，为支持服务定向目标建立一个坚实的基础。

（作者：Thomas Erl 译者：杨君 来源：TechTarget 中国）

本周 SOA 模式：服务分解

一个服务本身就是一个服务生命体，需要独立发展。我们在编制 SOA 设计模式目录时，了解到不仅在设计阶段，甚至是服务生命周期的后续维护阶段也会涌现许多模式。

很多项目都会有一个普遍场景：

• 在 SOA 实施的初期，服务的模拟和设计会受到基础设施和技术的制约。这些限制条件迫使我们缩小服务组合的规模，降低跨服务信息交流的程度。结果，每个服务包含了更多的逻辑并且颗粒度更为粗糙。

• 基础设施得到了提高（主要是因为新平台进行了升级或是拥有了更好的软件等）。我们的服务组合是由粗颗粒服务构成的，并通过就环境的参数交付这些服务。但是，我们意识到服务可以拥有细颗粒度（性能更为优良，构成更为有效）因为基础设施可以支持规模更大的服务组合。

为了适应当前的这一情况，在服务被部署到一个或者是多个细颗粒服务中后，服务分解模式提供了一种分裂服务的技术。

当然，许多版本控制和变动管理的参与者都对这个方法很感兴趣。我们怎样才能分离带有合同的服务，而又不会令使用这些服务的客户程序受到影响呢，同时实时运行还要依赖服务的现状？

为了解决这些问题，我们需要其它的 SOA 设计模式来帮助我们解决服务结构模式问题。

• 代理功能——当逻辑从一个服务转移到另一个服务当中时，我们就会利用这个模式维持服务合同所描述的原有功能。

• 服务表层—为了支持准许代理功能，这一多用途模式可以被用来建立数据处理的表现层，连接原有服务和新服务。表面组件可以促进新建服务的相应功能。这样一来，便可以代表原有服务的客户，成为服务客户。

当原始服务逻辑移到新地方时，很可能产生行为变化。如果同时实施以上两个模式和服务分解模式，表层逻辑就可以弥补这种变化。

要想成功分解一个服务，需要一个必要的前提即得出的颗粒度更为细致的服务要有完全不同的功能背景。当模拟并设计这些新服务时，所有可用的服务定向原则和模式都应该被看做是带有其它新服务的原则和模式。同时还需要实施像服务规范化这样的基础模式，以确保新服务可以和现有服务清单上的其它服务结合在一起。

服务分解的后续时期还有一个共同问题，但是既定的一套功能无法和新服务的功能环境完全吻合。这意味着新服务只能要求原始服务功能代表一部分。

这里有几个解决这个问题的方法，包括一个代理功能混合应用，在这个应用中原始服务保留了一些逻辑但是还是会呼叫处于其它位置的新服务。但是，在原有服务的建模初期，还要考虑另外一个模式，以便满足将来的分解要求。这个模式叫做被分解功能，主要是让我们预先思考，一个既定的粗颗粒服务环境是如何被分成众多新颗粒环境的，并且和主要服务功能相对齐。

在 SOA 设计模式目录中，服务分解模式在分类上是一种治理模式，尽管它和服务设计之间的联系更多。所有的这些模式都可以帮助扩充现有的服务集合（支持服务组合和服务重组），并使其更为合理化。

(作者：Thomas Erl 译者：杨君 来源：TechTarget 中国)

本周 SOA 模式：典型模式

SOA 设计模式目录中，没有什么比典型模式更容易理解，也没有什么比典型模式更难实践了。此外还有一些富有争议性的模式。实际上，典型模式的操作潜能也是决定服务清单架构规模和状态的基本要素。

这一切都归结为建立基准互操作性。典型模式可以确保服务带有合同，这些合同可以在标准数据模型（模式）的基础上共享商务数据。典型的数据模型 1 认为不同应用应该被整合在一起，以便在共同数据模型的基础上共享数据，而典型模式则要求提前在服务合同里创建普通数据模型（Hohpe、Woolf）。因此，要想成功实施这种模式，必须建立并执行这些设计标准。

在讨论数据模型的标准化之前，我们先来描述一下“基准互操作性”。

当服务和服务用户程序发生交互作用时，通常会依据某些结构和一套规则来传送并组织数据（通常以信息的形式）。这个结构以及相关的规则构成了数据的形式表达法（或者模型）。

我们用不同的数据模型代表不同类型的数据。这时服务在数据共享方面就会遇到问题，因为数据模型是不兼容的。为了解决这个问题，我们会运用数据模型转换这一技术，数据模型映射逻辑正是通过这一技术被开发的。这样数据服务之间交换的数据就可以在运行时间被转换成和数据模型相符的数据。这一技术获得了巨大的成功以至于人们想要开发与其相应的数据模型转换模式。

但是，伴随着数据模型转换技术的诞生，数据模型转换的过渡使用又成为了一个现实问题，这些问题往往和架构复杂程度，开发任务的增多以及运行时间效率要求有关。它会极大地影响大型服务组合，如果你贴近中间件，真的会听到运行时间延迟所发出的吱吱扭扭的响声。

我们将在数据模型转换模式这一系列文章里，向大家陆续介绍具体细节和相关问题。对于我们来说，目前最为重要的是弄清楚，应用典型模式的首要目标是要避免应用数据模型转换。

这时我们还是要观察数据标准以及其应用的规模。不同的项目小组在不同的时间建立典型模式，这就要求每个项目小组都要使用事先定义好的数据模型，这个要求似乎很简单，但是真正做到却并不容易。很多机构都在为标准化的数据模型的执行和治理而斗争——以至于引发机构权利之间的争斗、个人的不满情绪以及一致性和变化管理之类的技术难题。

还有为什么典型模式都要和网域清单一起应用。将应用、实施以及标准化数据模型的治理限制到可管理规模的服务清单的范围内，这一点增加了成功实现这一模式的潜能。

典型模式简要概述了建立在筒仓基础上，集成企业向服务定向的过渡。这一模式解决了一个大问题，同时也对其持续的应用做出了更大的承诺。

本周的 SOA 模式系列文章包含了 SOAPatterns.org 社区网站、“SOA 设计模式”（Erl et al.，ISBN：0136135161，Prentice Hall，2009）一书的作者以及投稿人的原创内容和独到见解。其最新出版一书的标题为“Prentice Hall 出版社服务定向技术丛书，作者 Thomas Erl”（www.soabooks.com）

（作者：Thomas Erl 译者：杨君 来源：TechTarget 中国）