服务定向原则汇总

服务定向原则汇总

本专题分六部分探讨服务定向原则，主要探讨如何将服务定向原则应用于构成服务的自动化逻辑。如何越过单个服务层面，应用作为范例的服务定向并形成能够封装整个企业领域的服务层。著名作家 Thomas Erl shares 通过他的第二本 SOA 专著的部分节选和我们一同分享他关于服务定向设计实例的独特见解。

服务定向简介

我们建立了一个服务定向设计范例。目前这些范例为我们提供了八个普遍原则。

- **服务定向原则第一部分：服务定向简介**

服务合同和松耦合

我们要通过讨论清单上的前两个原则：必须使用服务合同和创建服务间叫做“松散耦合”的关系来深入探讨这些原则。

- **服务定向原则第二部分：服务合同和松耦合**

服务抽取性和服务重用

服务合同及松耦合直接相关的服务定向的一个方面就是抽取性。只有通过抽取性我们才能控制基础服务逻辑向外部世界展现的那部分。

❖ 服务定向原则第三部分：服务抽取性和服务重用

服务的可发现性与可组合性

无论一个环境里是否真正存在一个服务登记处，我们将要讨论发现服务的必要性。我们还要近距离观察一个十分重要却经常被人误解的特征即服务的组合性。

❖ 服务定向原则第四部分：服务的可发现性与可组合性

服务独立性和无状态性

我们将要描述服务的无国界性和独立性，并挖掘其深层的内涵，宣传一个服务基本服务的特定设计特征。

❖ 服务定向原则第五部分：服务独立性和无状态性

原则的相互联系和服务层

我们将关注面向服务设计原则间的主要关系，以及怎样通过使用服务抽取层来进一步更好的应用范例。

❖ 服务定向原则第六部分：原则的相互联系和服务层

服务定向原则第一部分：服务定向简介

分六部分探讨服务定向原则，这是第一篇，著名作家 Thomas Erl shares 通过他的第二本 SOA 专著的部分节选和我们一同分享他关于服务定向设计实例的独特见解。书名是“面向服务的架构：概念，技术和设计”书中额外还附有评论。

通常采用 SOA 技术的人都希望只要成功实施 SOA，那些和面向服务技术平台相关的利益就能得以实现。但是，要想真正且有效的实现 SOA 转变这个长期且具有战略地位的目标必须采用和自动化逻辑设计一致的方法。

在建立一个面向服务的方案之前，先要了解是什么令一个独立的服务适合 SOA 支持其战略目标。换句话说，应该在工程的生命周期前期提出这个问题，即确保服务真正是面向服务建立起来？

答案取决于一个设计范例，这个设计范例把面向服务的结构模型和原来其他的模型区别开来。这个范例是面向服务的，其模拟商业自动化逻辑的方法已经成为一套为人们普遍接受的原则。我们可以在一个典型的面向服务的环境中，在应用、定位并形成原始成分（服务、说明、信息）时发现这些原则。

在这一系列的文章中，我们主要探讨如何将服务定向原则应用于构成服务的自动化逻辑。

在文章的后面，我们将讨论如何越过单个服务层面，应用作为范例的服务定向并形成能够封装整个企业领域的服务层。现在让我们先从解释什么是服务定向入手。

服务定向及其分类

服务定向影响很大，它产生影响的根源是软件工程理论，例如"separation of concerns."（相关分类）。这个理论在一个概念的基础上提出的，即把大问题分解成一系列较小的、单个的问题。这样的话，处理大问题的逻辑也能被分解成较小并与之有关的小集合。该逻辑的每一部分都能解决一个特定的问题。

我们已经在既定的范例中实施了这个理论，比如面向对象和基于构件的方法。服务定向被认为是实现相关分离的独特方法（尤其在最近）。更具体地说，关键的服务定向原则为如何实现分离提供了一个独特的方法。通过实施这个理论，为 SOA 建立了一个基本范例。如果你研究与目前 SOA 实施相关的普遍特征，你就会发现许多这些特征都和如何将这些相关分离有关系。

服务定向的普遍原则

公共 IT 机构、供应商、以及咨询公司对于服务定向的构成持不同的看法。作为 SOA 系统公司发起的现行行业分析措施的一部分，很长一段时间里我一直在进行对服务定向前景的研究和评估工作。该项目意在识别并描述一套由所有主要 SOA 平台所支持的普遍原则。因此形成了服务定向现时世界的定义。我们这一系列文章的焦点和中心都是围绕那些被 SOA 行业普遍接受并被相应的供应商所支持的原则所展开的。

最近创立的八个原则：

- 服务共享一个正式契约
- 服务是松散耦合
- 服务提取潜在逻辑
- 服务可以组合
- 服务可以重用

- 服务是独立的
- 服务是无国籍的
- 服务是可发现的

这八个原则里，其中独立性、松散耦合、抽象性、以及对一个正式契约的需求可以被看做是形成 SOA 基础的核心原则。尽管八个原则支持或被其它原则支持，但这四个原则是另外四个得以实现的直接条件。研究这些原则间相互关系的非常有趣，这些研究还为服务定向带来的独特动力提供了一个更深层次的视角。

值得注意的是 Web 服务本身就支持这些原则的子集，这就表明了为什么 Web 服务技术平台被认为是适合于构建面向服务的方案。在文章的后面，我们将要讨论这些原则间的关系，以及其中的一些原则是怎样通过使用 Web 服务在内部实现的。

下一步如何

要想全面了解服务定向是如何影响面向服务架构的，我们需要研究该应用在构成 SOA 基础部分的含义。除去这项研究，在该系列文章的第二部分我们将关注松散耦合以及四个原则中前两个有关正式契约的使用。

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

服务定向原则第二部分：服务合同和松耦合

在前面的文章里，我们建立了一个服务定向设计范例。目前这些范例为我们提供了下面八个普遍原则。

- 服务共享一个正式契约
- 服务是松散耦合
- 服务提取潜在逻辑
- 服务可以重用
- 服务可以组合
- 服务是可发现的
- 服务是独立的
- 服务是无国籍的

这些原则是“普遍的”，因为它们代表了行业层面的关于服务定向的观点。该服务定向由 SOA 系统公司执行的研究所决定。在文章的第二部分，我们要通过讨论清单上的前两个原则：必须使用服务合同和创建服务间叫做“松散耦合”的关系来深入探讨这些原则。

服务合同的用途

服务被定义成使用一个或多个服务描述文档。在 Web 服务领域，技术服务描述文档是典型的 WSDL 定义和 XSD 模式。该文档是又一越来越具重要性的文档形式。每个文档都可以被分成服务元数据，每一个服务元数据都提供和服务相关的信息。我们可以把服务描述文档整体看做是建立了一个服务合同——一套必须被满足并被能为潜在的服务请求者所接受的条件，以确保成功完成通信和互动。

- 服务端点
- 每个服务操作
- 每个由操作支持的输入和输出信息
- 每个信息内容的数据表示模型
- 服务和操作的规律和特点

因此，服务合同界定了大量的解决方案环境的底层架构，甚至可能提供的语义信息，这些语义信息能够解释作为方案一部分的服务是怎样完成一个特殊任务的。总之，该信息确立了该协议的条件，服务的用户都应该遵循这个条件。

因为许多协议共享这个合同，合同的设计就尤为重要。同意合同的服务请求者便依赖于这个定义。因此，在合同最终发行前，应进行仔细的维护和校对。

在服务定向中，服务合同代表一个基础原则。因此，它能支持并使其它原则得以执行。例如服务抽象性、组合性、可发现性以及接下来要讨论的松散耦合。

怎样才算是松耦合呢？

没人能预测一个 IT 环境将如何演进。自动化方案将如何发展，是集成化还是随着时间的推移被取代，这些永远都不会被精确的规划出来，因为促使这些变化的要求对于 IT 环境来说是外在的，应用服务定向的主要目标就是能够用高效的方法回应无法预测的变化。通过在服务之间建立松耦合联系，我们就能实现其灵活性。

对于一个传统的分布式结构模型来说，SOA 建立在一个概念之上，即把方案逻辑划分到许多逻辑单位之中。这些逻辑单位能够被集中在一起令商业工作自动化。一个将 SOA 和其它架构区别开的特征就是这些逻辑单位是如何被要求联系在一起的。

这就是耦合的来由，软件程序间的耦合可被看做是代表测量依赖性的一种方法。依赖性越强，耦合就越紧。而无依赖不表示一个去耦状态。我们强烈推荐最大限度地减小 SOA 里逻辑（服务）单位的依赖性。这种特定的关系被称作“松散耦合”并且通过将服务和请求者间的依赖性限定到服务合同所表达的信息范围内，同时在设计协议时不必要只针对某个特定的服务请求者，就可以实现松耦合。

如果坚持这个原则，新建的服务就会明显的独立于其它服务。这样就使正在标准化 SOA 的机构能够积累一系列服务，作为相对独立的逻辑单位，这些服务可以按照需要被配置到新的结构中。当为了建立一个面向服务的企业而创建多种服务时，这些服务可以被分成特定的服务模型。这就使他们在自己封装的逻辑方面更为专业。

在标准化一套服务模型时，能够成功抽取更大的域。这就使机构能够通过创建代表不同域的服务层而在企业的特定领域有效的建立具有战略意义的松耦合联系。通过这样做，松耦合带来的利益会得到增大—最明显的是它促进了机构灵活性—。这就展示了作为范例的服务定向并没有局限于服务设计层面。它的原则可以在企业整个领域得以广泛应用。

下一步

在文章的第三部分，我们将继续探讨另外两个原则，其中一个是服务抽象性。我们将扩展有关松耦合的解释，并研究怎样通过让基础逻辑和被服务封装的技术独立演进把抽象性和促进组织灵活性联系在一起的。

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

服务定向原则第三部分：服务抽取性和服务重用

与我们之前谈的服务合同及松耦合直接相关的服务定向的一个方面就是抽取性。只有通过抽取性我们才能控制基础服务逻辑向外部世界展现的那部分。

抽取功能和技术

提及服务接口层面的提取，原则上是鼓励人们建立类似黑盒的服务，并有意隐藏潜在用户的基本详细资料。通过规范的使用服务合同可以完成抽取。将服务的公开信息限制在服务合同指定的范围内，就可以极大程度的在私人（隐藏）信息和公开（可消费）信息间实现分离。

这里对逻辑服务代表事物的数量没有限制。一个服务可能只是执行一个简单的任务，或者在整个自动化方案被用作网关。对一个服务能使用的自动化方案的来源也没有限制。

举个例子，单个服务可以揭示多种不同基础系统的逻辑。事实上，当我们在向服务模型标准化迈进时便建立了一个和营业个体及和商业活动相关的功能环境，人们希望在充满旧方案的环境下，一个服务能够普遍揭示依赖许多不同系统的功能。

服务接口层提取是分布式平台提供的固有的品质，例如组件和基于服务的 Web 架构。Web 服务的应用是协同的，因为它提升了可提取的层次，使其远远超过了功能层面。Web 服务从基础自动化逻辑中提取专有的实施细节，这使潜在用户免于和特定的供货技术相连接。尽管我们把抽取看作是服务的一个特征，但事实上却是集中抽取基础逻辑的单独操作。服务就是这些操作的容器。任何既定服务抽取的水平很大程度上取决于每个服务操作的水平。

这就要强调服务合同的设计。服务合同上表达的越多，我们抽取的内容就越少。服务合同越一类化，服务的客户就会更不具体，过程就越缓慢。这就决定了我们选择从服务合同中要表达的(而不是提取的)重用的潜能

通过重用促进灵活性

不管是否是即时要求重用，服务定向支持所有服务中的重用。这个基本的原则迫使我们尤其注意每一个自动化逻辑的交付单位，我们称其为“服务”

最初的战略目标和重用有关，即用可重复价值将每个服务定位成一个 IT 资产。随着可重用资产的增加，要少建设而多使用我们已有的一切，完成新业务自动化要求的机会也在增加。

人们希望通过减少建立自动化逻辑的时间，改进机构对变化的反应能力。通过减少共同努力，自动化要求的完成有望更有助于提高成本合算的效率，令提高 IT 开发环境的效率成为可能。这听起来像是无理的要求，但是为了实现这些利益，许多机构都在创建高度可重用服务清单上面进行了巨额投资。

这一原则有利于各种形式的重用，其中包括应用程序互操作性，组成和建立跨领域服务或公用服务等。正如我们以前设立的服务一样，一个服务就是一个相关业务的集合。因此，由单个操作封装的逻辑必须可重复使用，以保证其作为一种可重复使用的服务。

对意义深远的重用的强调，也突显了作为一项执行方案 Web 服务的适用性。通过行业标准通信框架，使每一个服务都可以得到使用，并可以大大拓宽重用的潜力，，因为服务封装的逻辑现在已经可以为服务请求者所用，这些服务请求者由不同的基本技术建成。

这归根结底取决于服务合同

这些原则让我们再次思考要求使用的服务合同，合同的内容决定了什么要被抽取出来，什么不用抽取。通过设计内容我们能决定没被抽取部分的类属和重用性。这就很有必要把一个服务看成是一项投资。建立面向服务方案逻辑往往更昂贵更耗费时间，因为人们需要考虑即时战略要求以外的事情。因此，鉴别服务定向打算完成的任务在证明投资合理方面非常非常重要。

下一步做什么

我们的文章已经进行一半儿了，希望能够进一步阐释这些独一无二的特征、要求、以及服务定向范例能带来的潜在利益。

在第四部分，无论一个环境里是否真正存在一个服务登记处，我们将要讨论发现服务的必要性。我们还要近距离观察一个十分重要却经常被人误解的特征即服务的组合性。

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

服务定向原则第四部分：服务的可发现性与可组合性

我们继续探索服务定向，我们现在关注两个原则，这两个原则似乎没有受到应有的重视。当涉及到服务设计时，人们的目光都集中在和 SOA 营销普遍相关的设计特征上，就是松耦合和重用。这些都很重要，但却不是实现 SOA 转变过程长远目标的关键，我们还需考虑得更多。在文章的第二和第三部分，我们讨论抽取和战略性的使用服务合同是怎样描述另外两个原则的，这些在一定程度上有助于实现可重用的松耦合服务。但是，如果不能被那些负责创建新客户的人所发现，即使是重用率最高的服务也不管用了。另外，如果不能形成有效的组合，即使是最松散的耦合重用的潜力也十分小。

服务的可发现性

服务可发现性这个特点能够帮助避免建立冗繁的服务或者执行冗繁逻辑的服务。因为每个服务操作都提供一个潜在的可重用自动化逻辑。和服务相关的元数据不仅需要充分描述服务的整体意图还包括单个操作提供的功能。

服务定向和可发现性相关，但又不同，在结构层面，服务可发现性指结构提供发现机制的能力比如一个服务登记簿或目录。这些扩展都成为支持 SOA 实施整体基础设施的一部分。

在服务层面，可发现性原则指的是单个服务的设计。所以不管具有可发现性的产品或延伸在它周围的实施环境是否存在，单个服务被设计得能够尽可能被发现。

原因是这里不需要服务登记簿，因为没有足够的服务目录来保证有一个服务登记簿，服务应该被设计成高度可发现资源。这样，当服务文件增加的时候，人们就可以更好的管

理服务的改良过后的统治，因为每一个服务都配备了足够的元数据用来恰当的表达其意图和能力。

服务组合性

随着服务文件增加，服务的组合在所难免，并且成为建立面向服务方案这个设计的很重要的一个方面。主要的原因是这个特殊的原则非常重要，以至于它能以这些组合中现有成员、控制器的身份参与其中。

任何服务都可组合的要求同时也强调服务操作的设计。可组合性是重用的另一种形式，因此需要用标准方式（和恰当程度的颗粒性）来设计操作以便最大限度的增大组合的机会。

普通 SOA 扩展强调组合相关性的和谐一致。这里，面向服务的业务流程可通过一个组合语言表达出来。例如 WS-BPEL，将流程本身归类为一个由父进程代表的服务组合。希望服务高度组合需求的和即时组合要求是否存在无关。

组合的可发现性

我们文中解释的每个原则都相互关联。例如，服务组合性，和其他几个原则一起应用的程度有关。

甚至可发现性也和有效组合相关。服务提取的基本规则就是一个服务能代表来自一切被支持源的逻辑范围。如果服务封装了其它，我们就有了一个组合。为了建立一个有效的组合，服务设计者需要一个方法，这个方法能找到能够用作组合部件的最适合服务。另外，一旦组合得以完成并被部署，代表这些服务的潜在用户能从意识到其存在、目的及其潜质方面获益。

发现支持所有这些情况，并使其得以实现，因此加速了服务定向的进程。

下一步如何

目前为止，我们讨论的许多原则都关注于服务合同的设计和利用。在下面的文章里，我们将要描述服务的无国界性和独立性，并挖掘其深层的内涵，宣传一个服务基本服务的特定设计特征。

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

服务定向原则第五部分：服务独立性和无状态性

机构继续以服务的形式建立企业自动化逻辑，这就极大的需要增加服务在运行时间操作的可靠性和效率。在用大量可重用服务组合一项服务时，这个需求就更加重要了。

可重用服务和并行操作

重用是 SOA 的核心部分，其作用非常重要，一些与企业相关的 SOA 变化的战略目标都和其有直接联系，以便成功实现自动化逻辑的重用。因此，我们要保证交付的服务不仅拥有重用逻辑，同时在用于现实世界时，还能被重用。

总之，我们要增加重用的机会。每一个被划分为“重用”的服务能为潜在的大量客户程序所用。结构是可以预测的。随着时间的流逝，我们需要同样的服务来促进不同业务流程或服务的自动化，这个服务可能成为众多服务组合的一部分。

这最终翻译成大使用量、不可预测的使用方案和一个我们非常有兴趣准备的运行条件；并行存取的运行环境。当一个服务被两个或更多的服务用户同时访问时，就会产生服务实例。这些的发生很大程度上取决于负责运行服务的供应商运行时刻平台。

但是，为了构造基础服务逻辑以便方便并行存取的条件和其它和与依赖性相关的条件，我们需要通过几个重要的步骤来为服务的设计定型。

这涉及到两个设计原则，我们将在文章中简要介绍。

- 服务是独立的

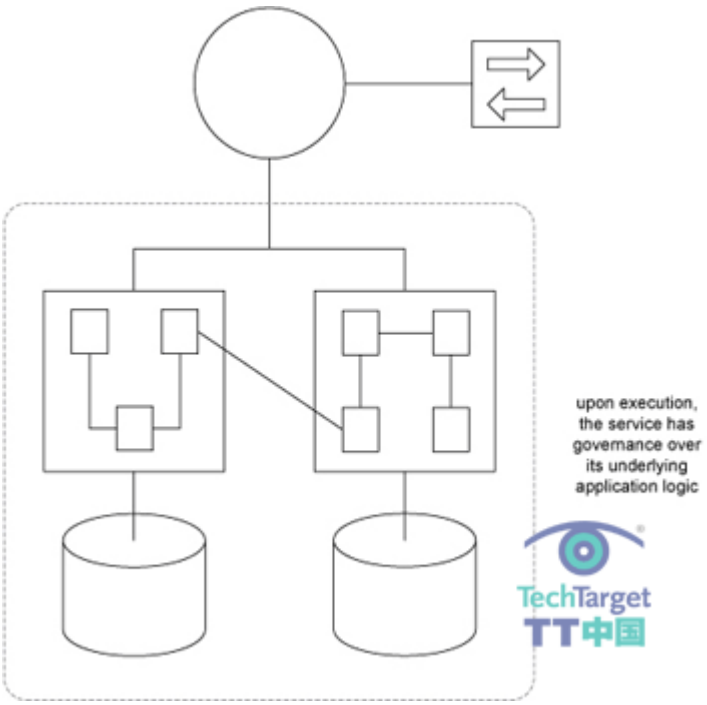
• 服务是无状态的

如果创建一个使用某个特定服务的程序，我可能不知道或不关心有多少其他的人正在使用这项服务。我会想基于服务合同和辅助 SLA 里的内容，这个服务能做些什么。因此，我会依靠服务的性能来提供一个行为、实行性、和可靠性可预测层，不管其是否会被使用。

服务独立性

当涉及到分解时，服务定向带来许多不同的态度。在建立一个企业服务清单时，尤其需要强调将清单上的每个组件定位成一个构造块。

对于服务来说，要提供可靠可预测的性能，他们需要很大程度上控制基础资源。独立性代表其量度，这个原则强调单个服务拥需要有高度的个体独立性。



通过增加服务在执行环境的控制，我们减少了在企业里共享资源所要求的依赖性。尽管我们不能经常提供一个被封装的逻辑专有所属权的服务。我们主要关注的是在执行时间里不管逻辑代表什么，服务能够实现合理的控制。

因为衡量独立性的不同方法的存在，有助于将它们区别开。下面，我们单列出独立性的两个层次。

服务层面的独立性—服务间的界限清楚明晰，但是服务也可能共享基础资源。例如，一个封装旧环境的包装服务，它控制旧系统，但同时也和老用户共享资源。

纯粹独立性——服务拥有并控制基本逻辑。这是建立支持服务逻辑时最常见实例。很显然，一个含有纯粹独立服务的服务清单更令人向往。这不仅帮助我们处理伸展性问题，如并行存取条件，还能帮我们将服务定位的更可靠以应对在杠杆作用重用自动化逻辑时“单点故障”。但是它通常要求建立新的服务逻辑并需要特别部署。这就需要花费更多的时间和精力。

服务无状态性

独立性是 IT 最容易理解的一个方面，但状态信息的构成却不太透明。由于这个原因我们需要在讨论这个原则前，花点时间来定义状态管理。

状态指一件事物的特殊情况。一辆车在运动的状态下行驶，而不是在固定不动的状态下行驶。

在商业自动化方面，一个软件程序通常被认为是有两个主要的状态。

- **主动状态**

• **被动状态**

第一个状态代表被调用的或者被执行的软件程序处于一个主动状态。当程序没有使用时，就处于被动或非主动状态。

在设计程序时，我们对软件程序在主动状态下的表现很感兴趣。我们如此感兴趣，事实上，我们还有应用于程序的其它状态，该程序代表主动状态的特定类型。在我们谈到状态管理时，有两种基本状况：

• **无状态的**

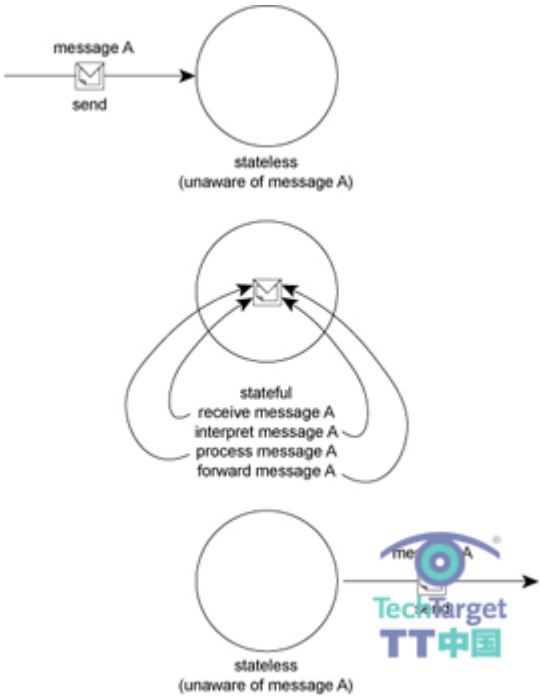
• **有状态的**

这些术语常常被用来证明一个程序的主动或运行时间状态，因为这和执行指定任务的处理有关。我们可以把这个数据称为状态信息。

一个程序可能是积极的，但其不一定参与了状态信息的处理。在理想情况下该程序是无状态的。正如你所猜想的，一个积极处理或保持状态信息的程序被认为是有状态的。

处理过程要求服务在常规基础上，增加被重用、组合和并行存取的机率，同时也要优化处理逻辑的服务。当我们建立面向服务架构时，需要特别注意状态管理。因此，我们如此强调优化架构内服务信息的管理，以至于我们有了一个用于服务设计方面的原则。

该原则规定，服务应该最大限度减少他们管理的状态信息的内容和他们有状态的期限。在一个面向服务里，状态信息通常代表现有服务活动的特定数据。当服务在处理一个消息时比如，它是暂时有状态的（图 2 ）如果一个服务负责在长时间内保持状态，其被其它用户使用的能力就会受到阻碍。



至于独立性，无状态是一个服务最佳状态。该状态能促进可重用性和伸展性。对于一个服务来说，应该尽量保持小状态。其基本服务逻辑应该考虑设计成无状态的过程。另外，架构本身就应该配有在大范围服务内支持该原则应用的延期扩展。

下一步如何？

在服务设计中，无状态性和独立性并行不悖.他们每一个都支持另一个的目标，并一起支持 SOA 的目标。在文章的最后一期里，我们将关注面向服务设计原则间的主要关系，以及怎样通过使用服务抽取层来进一步更好的应用范例。

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

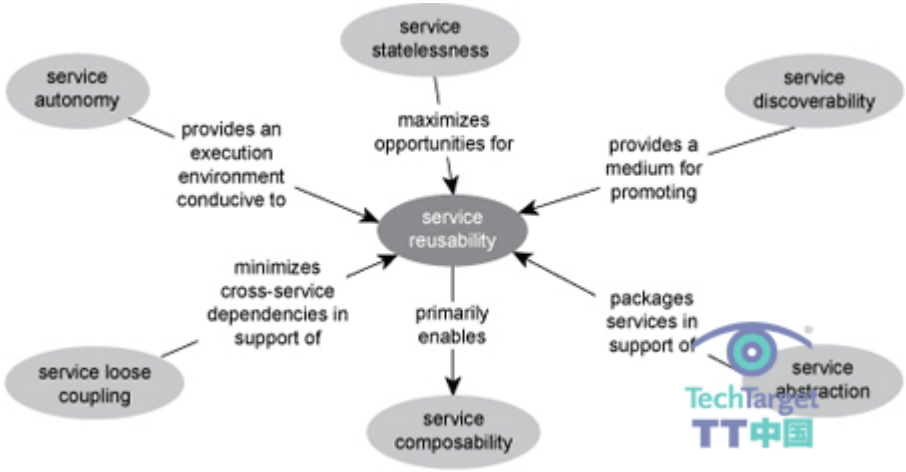
服务定向原则第六部分：原则的相互联系和服务层

既然我们已经分别介绍了面向服务的普遍原则。我们就能研究它们应用的其它方面。在文章的最后部分，我们将讨论这些原则是怎样相互联系并相互影响的。它们之间的关系非常具体并形成了面向服务范例独特的动力。我们简要的看一下这些特定的原则是怎样通过使用抽取层来让这些原则的应用超越服务层面的。

这些原则是怎样联系起来的？

要深入理解服务定向，认清应用这些普遍原则的原因和后果非常重要。通过研究原则间的相互关系，我们不仅要保证每个服务支持或被其它服务支持得以实现，同时要对什么能令服务定向和其它设计范例相区别进行评估。

拿服务重用性这个基本原则为例，在图 3 中你会看到这些椭圆符号代表的原则和箭头表示的关系。服务重用被放在了中心，因为这是我们目前感兴趣的关系。指向该原则的箭头代表其它原则产生的影响。从符号指出的箭头指明了实施可重用的自动化逻辑怎样影响到另一项原则的实现。



在该示意图里，明显可以看出有多少其他原则影响并支持服务在现实中能够实现的重用。实现重用不仅取决于设计，还和服务在现实世界的实施有关，该实施是服务定向的核心目标，也是实现 SOA 利益的关键.因此一些其他的原则显露出来支持这个目标。

我们仔细研究一下这些关系。

• 服务独立性形成了一个有利于重用的执行环境，因为服务实现了高度的独立和自治。服务的依赖性越小，可重用性的范围就越广。

• 服务无状态性支持重用，因为它能最大限度地提供服务并促进一个普通服务设计，这个普通服务设计推迟对状态管理服务范围以外的对特定活动处理。

• 服务抽取性促进重用，因为它确定了黑匣子的概念。专有数据处理详细资料被隐藏起来和潜在消费者只知道有一个由普通公共接口代表的存取点。

• 服务的可发现性促进重用，因为这可以让那些建立用户的人去寻找，发现和评估提供重用功能的服务。

• 服务松耦合建立一种内在的独立性，该独立性把一个服务从即时联系中解放出来，这就使实现重用更为简单。

• 主要是由于重用，服务组合性才得以实现。当这些被组合的服务是为了重用而建立时，通过将现有服务组合，自动化要求就能够得以实现。（从技术角度来说，可以建立一个服务，该服务的唯一目的就是被其它服务组合，但是我们普遍强调重用。）

研究原则间的关系可能很复杂。这不仅是一个原则是否被应用的问题，同样也是其被应用程度的问题。

在该系列文章的前面，我们知道，每一项原则可以落实到某一个程度。这可以使量度一个原则实际实现的程度及其可以在何种程度上影响到其它原则十分具有挑战性，。不过，根据第一手的经验，我可以告诉你，反复浏览面向服务的分析与设计过程可以使我们熟悉服务定向，并最终明智的判断这些原则在何种层次上相互联系。

抽取，松耦合和服务层

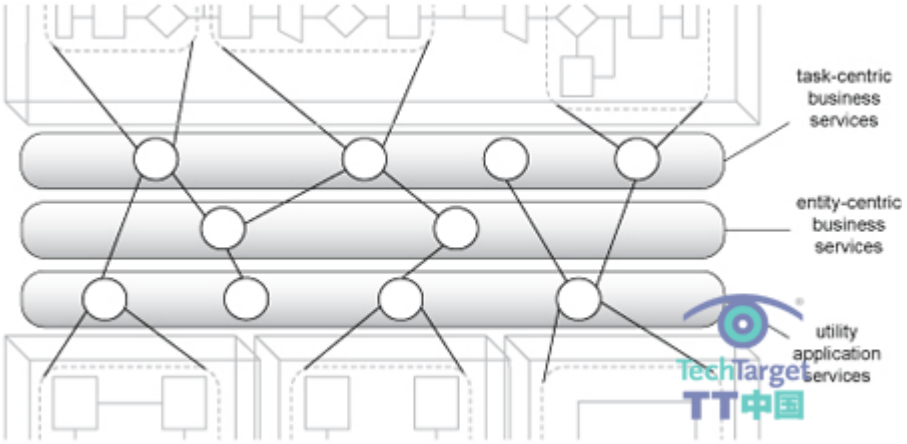
学习这些关系使得我们要考虑到一个既定服务的单个原则的应用以外的东西。但是，要在整个企业范围成功的应用服务定向，需要进一步研究超越现实服务界限的一些特定的原则。

我们已在由六部分组成的业务分析和 SOA 系列文章的第二部分介绍了服务模型。简要的说，作为一个整体，服务模式，用预先定义的设计特点为各类常见的服务建立样板。在业务分析和 SOA 的文章中，我们的重点是有业务环境的服务模式，其中包括：

- 以实体为中心的业务服务
- 任务为中心的业务服务
- 过程服务

另一个我们应该简要介绍的模型本意不是以业务为中心的。我们将其看作是一个应用或一个基础设施服务，这个基础设施服务普遍代表一个处理逻辑的实体，该逻辑用于解决跨领域问题。常见的例子包括通知，事件日志，异常处理和数据格式转换。

如图 4 所示的情况，涉及三个服务层，底层是把先前所描述的服务模式称作实体应用服务。



使用服务模型使我们能在大范围内实现功能抽取的概念。基于同样模型的一组服务和其有共同的特点可以在企业里集体抽取一个域。这样就能有效的建立一个服务抽取层。

这些抽取层不用局限于先前我们提到的那些服务模型。一个机构可以定义自己的一套服务模型。每一项服务模型都可以具体到它喜欢如何分割逻辑企业域。举例来说，一个商业服务模式，可以专为一个公司业务部门所建立（如人力资源部或销售部）。基于该模型的服务就会有功能边界，这些功能边界被局限到机构的那个部门，并且共同封装一个和该部门相对应的域。所有的这些任务将服务可抽取性这个原则提升到了一个新的层面，因为我们应用的服务超越了单一服务界限，代表服务的集合。

使用域级别的抽取，让我们也可以应用远远超出服务界限的松耦合。封装域的服务层代表企业内部的功能逻辑体，这些功能逻辑体为了能让普通业务流程和任务自动化，需要相互作用。就我们所知，松耦合通过让服务相对独立的演进令 SOA 受益。一旦应用于企业领域，服务抽取层在一个机构的所有部门建立了松耦合。因此，在支持高效、跨域交互作用时，能够准许域更加独立地演进。

结论

在目前市场里，围绕“SOA”一词的使用有很多歧义。了解那些被认为是"以服务为本"的事物的真正含义是什么，比以往更加重要。对服务定向设计范例和其原则的丰富知识，可以帮助我们更好的理解。这使杠杆作用面向服务的计算机处理技术为了支持你的战略目标可以提供些什么，并能使评估“面向服务”的产品和平台的合法性所需的清晰度得以实现。

我希望您能喜欢我们对服务定向的范式的介绍。若需要对我们讨论的有关原则有更详细的了解，希望您访问 www.serviceorientation.org 。

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