企业服务总线 ESB

企业服务总线 ESB

ESB（Enterprise Service Bus，企业服务总线）是传统中间件技术与 XML、Web 服务等技术结合的产物。ESB 提供了网络中最基本的连接中枢，是构筑企业神经系统的必要元素。

企业服务总线 ESB 就是一种可以提供可靠的、有保证的消息技术的最新方法。ESB 中间件产品利用的是 Web 服务标准和与公认的可靠消息 MOM 协议接口(例如 IBM 的 WebSphere MQ、Tibco 的 Rendezvous 和 Sonic Software 的 SoniCMQ)。ESB 产品的共有特性包括:连接异构的 MOM、利用 Web 服务描述语言接口封装 MOM 协议，以及在 MOM 传输层上传送简单对象应用协议(SOAP)传输流的能力。大多数 ESB 产品支持在分布式应用之间通过中间层如集成代理实现直接对等沟通。

企业服务总线（Enterprise Service Bus，ESB）的概念是从面向服务体系架构(Service -Oriented Architecture， SOA)发展而来的。SOA 描述了一种 IT 基础设施的应用集成模型，其中的软构件集是以一种定义清晰的层次化结构相互耦合，其中，一个 ESB 是一个预先组装的 SOA 实现，它包含了实现 SOA 分层目标所必需的基础功能部件。

ESB 简介

ESB（Enterprise Service Bus，企业服务总线），是过去消息中间件的发展。ESB 采用了“总线”这样一种模式来管理和简化应用之间的集成拓扑结构，以广为接受的开放标准为基础来支持应用之间在消息、事件和服务的级别上动态的互连互通。 ESB 是一种在松散耦合的服务和应用之间标准的集成方式。它可以作用于：

•面向服务的架构－分布式的应用由可重用的服务组成

•面向消息的架构－应用之间通过 ESB 发送和接受消息

•事件驱动的架构－应用之间异步地产生和接收消息

- ESB:平静之下的暗流涌动
- 企业服务总线(ESB)集成化

ESB 有哪些应用

在 SOA 实现的早期阶段，当目录仅仅由一个或者两个基于项目的服务组成的时候，ESB 看起来是英雄无用武之地。但是幸运的是，如果在企业中采用 ESB，那么服务的部署会被加速。任何一项策略都被要求能够提供随需应变的扩展性，可靠性和足够的性能等特点。从构架的角度，使用一个合理的原则避免服务混乱不失为一个好的想法。

随着企业 SOA 的逐步成熟，业务功能会从各种源头被挖掘和发现出来。这些服务的提供者可能是遗留的应用，第三方软件包或者主要解决方案提供的功能。虽然理想状态是所有这些服务都使用相同的技术，但是现实情况证明这是不可能的。很有可能 Web Service 标准仅仅是使用的技术中的一个而已。

- 事件流程处理与灵活企业应变
- 企业服务总线 ESB 的数据服务
- 重造 Java ESB:JBI 与 ServiceMix
- Web 服务完善后是否还需要使用 ESB
- ESB 如何帮助企业重组应用程序和系统

ESB 与 SOA 的关系

ESB 是逻辑上与 SOA 所遵循的基本原则保持一致的服务集成基础架构，它提供了服务管理的方法和在分布式异构环境中进行服务交互的功能。

可以这样说，ESB 是特定环境下(SOA 架构中)实施 EAI 的方式：首先，在 ESB 系统中，被集成的对象被明确定义为服务，而不是传统 EAI 中各种各样的中间件平台，其次，ESB 明确强调消息(Message)处理在集成过程中的作用，这里的消息指的是应用环境中被集成对象之间的沟通。最后，事件驱动成为 ESB 的重要特征。

- 使用 ESB 简化 SOA 复杂性
- 超越 ESB：下一步 SOA 难题
- Forrester:视 ESB 为 SOA 的本质
- Sun 紧抓 ESB 和 JavaEE 实现 SOA 应用
- 独家专访：如何看待开源 ESB 和基于 REST 的 SOA?

ESB 的实施

考虑一个 ESB 作为一个中间件平台提供许多与中央运行时处理过程和服务管理相关的特性。由 ESB 主导的服务能够以 Web 服务的形式发布，并且只要服务合约是由客户定制的(理想情况下是标准化的)，你可以非常确信地认为你能够暴露服务逻辑而不需要将合约匹配到任何底层基础服务实现细节上去。这将给你充分的自由去将服务安置到另外一个 ESB 平台上或者其它地方上。而且这个可选择性并不会让 ESB 变得孤立。

- 如何转变 EAI 到 ESB
- 实施企业服务总线 ESB

ESB：平静之下的暗流涌动

通过标准接口开始进行应用程序的开发，而不采用在应用程序开发完成之后再定制接口的方式，虽然这样做也许是很流行的，但是仍在存在很多关于如何确切的定义企业服务总线(ESB)的问题。

去年的某个时间，在消息传递方面，企业服务总线(ESB)的优势已经超过了传统企业应用程序接口(EAI)方法。

通过标准接口开始进行应用程序的开发，而不采用在应用程序开发完成之后再定制接口的方式，虽然这样做也许是很流行的，但是仍在存在很多关于如何确切的定义企业服务总线(ESB)的问题。许多大公司，如 BEA 系统公司、IBM 公司现在正迅速进入 ESB 的市场，他们已经完全依靠企业服务总线(ESB)。

“在过去的八个月里，整个市场上都充斥着企业服务总线(ESB)，”Sonic 软件公司产品市场部高级主管 Jonathan Bachman 说。“从传统的整合仲介到应用服务器，你已经开始把它们都称之为企业服务总线(ESB)。这潜在地破坏了它们之间的分类;事实上这样做使这种分类消失。”

根据目前这个状况，Sonic 公司表示了自己的作法，设法使本公司的企业服务总线(ESB)产品成为一种 ESB 衡量标准，即控制 ESB 应该是什么样子的以及它应该如何操作的。

根据 Bachman 所说，关键问题是“企业服务总线(ESB)终端要创建一个可扩展的、灵活的并且不影响服务的抽象分离。这个终端不需要通过网络集线器来发送消息。它能够在本地工作。”

接口的变动不应该要求服务的代码发生改变，而且服务的改变不应该要求接口的代码进行变动。简单扼要地说，就是它在工作时是松耦合范例。

虽然简单性并不是处处必要的。也许企业服务总线(ESB)底层的设计概念可能是很容易掌握的，但是 Bachman 提醒大家注意到在企业服务总线 Sonic ESB 中有 110 种“主要部分”。

“这样的活，要做一个比较分析是非常困难的，”他说。“当你把两个产品放在一起进行观察时，他们之间的相似之处以及区别之所在并非你一眼就可以看出来的。”

举例说明，企业服务总线 Sonic ESB 包含多平台编辑器以及调节规范。对你而言，可扩展的样表式语言转换(XSLT)风格的样板编辑器或者能够为可扩展标记语言(XML)数据创建 XQuery 表示式的能力并不是十分重要。同样，Sonic 公司为处理消息(调节)所进行的六个步骤的基本过程可能与其它的企业服务总线(ESB)所提供的大相径庭，也有可能是不同外表掩藏下的同种实质。

为了解释清楚属于方面的混淆，Sonic 公司在它的参考模型文档中添加了 11 页的术语列表。

除了标准 ESB 之外，也存在着为面向服务架构的创建和管理的增值服务。 这些增值服务包括连同其他功能一道的处理业务过程工作流、外部合作伙伴协调以及数据库访问。

当涉及企业服务总线(ESB)所提供的工具时，Bachman 很容易的指出它不是一个一次完成面向服务架构(SOA)的工具。

“企业服务总线(ESB)是为你的服务而创建的集合，它只是面向服务架构(SOA)的一部分，并不是全部，”他说。

根据 Sonic 公司所做的研发，对于“一个分布式服务架构”而言，它的关键的功能和属性包括:

新服务以及新服务类型可以通过配置实现，而不需要重新写程序实现。

一个企业服务总线(ESB)的接口变动，可以不改变代码，并且不需要在接下来的开发阶段重新编译和配置该服务。

无论 ESB 容器处于网络的任何位置，服务实例都可通过远程提供，这样就允许通过负载平衡或者多服务实例来配置新功能以及管理性能，增加灵活性。由于主机具备加工或者进入其他资源的能力，这样做还允许把服务配置在能够发挥其最佳性能的物理位置。

只要快速服务启并且及具备运行的能力，服务所需的本地访问手工配置就可以实现，甚至在网络失败的情况下都可以防止入侵 ESB 库。

要配置新的或者变动后的服务，可以通过分布式网络来进行 ESB 服务执行和配置的自动分配。

分布式 ESB 的服务容器和通讯仲介可以通过相同的底层消息基础结构进行管理，这个基础结构是用于服务通讯，并且这个功能可以为可测量性和可靠性区分通讯。

事件包括 ESB 过程所调用服务的进入和退出， 它可以甚至可以通过高度分布的配置来追踪管理过程的执行。

自动负载平衡以支持性能和可靠性目标，它是通过一个服务的多个实例实现的。

连续有效的架构提供高可靠性、多协议消息传递，它是通过从原始消息数据到第二仲介数据进行复制。这样做保证了在一个通讯仲介传输失败的情况下仍能够进行连续有规则的消息传输。

动态路由架构允许包括 Web 服务请求和相应在内的消息通过仲介的联合复选的配置实现自动路由。不考虑当本主机目的地受到安全行使权利的限制时的情况，这样做也允许消息进入企业服务总线(ESB)中的任何一个仲介，并且路由到正确的目的地。

Bachman 认识到在产品之间多样化将会出现，但是他主张使用 ESB 来展示在具体条件下这些正在变得越来越重要。新的 Java 企业整合(JBI)规范的出台多半可能证明在这一竞争舞台中的利益之所在。

(作者：Michael Meehan 来源：TechTarget 中国)

企业服务总线(ESB)集成化

大型机群趋向于保守文化，专业人员对于安全和性能问题固持己见，并且他们以相当怀疑的态度对待新技术。因此应用更新技术这一想法必须首先要得到证明，例如用于大型机应用集成的企业服务总线(ESB)，这话出自 Jody Hunt，麻省沃尔瑟姆福市，Iona 科技公司的高级产品市场经理。

这就是销售商们所喜欢的 Iona 科技公司和 Neon Systems 股份有限公司，在得克萨斯州的 Sugar Land,这两家公司正着手开始做 ESB 型的大型机应用集成产品。“与 Web 服务联系在一起的大型机领域比与 ESBs 联系在一起的大型机领域更具有发展力，”Hunt 说，“现在他们妥协的认为 Web 服务很好。”

许多公司数据仍然存储在大型机中，迄今为止在大型机领域的集成化仍然是一个棘手的问题。典型的点对点连接所必须的软硬件都是需要高昂的维护费用，而且会影响企业运营的灵活度。

Neon 最近推出 Shadow RTE 产品，称之为企业服务总线上的“on-ramp”。Iona 科技也为大型机推出了 Artix 这种产品， 称之为“可延伸的”企业服务总线(ESB)，而且到目前为止已经有效使用一年多了。这意味着在面向服务的架构(SOA)中上述两个产品都对企业范围内的企业服务总线(ESBs)起到了作用。

“我们允许其它的企业服务总线(ESBs) 通过我们来获得大型机上的信息;我们提出一个工业标准形式来与企业服务总线(ESBs)集成。” Neon 公司战略与解决方案的资深副主席，Robert Evelyn 说。根据 Neon 公司的理论，一个企业范围内的企业服务总线(ESB)是无法提供大型机所需要的设备和最优化配置的。

Shadow RTE 被配置在 IBM z/Series 大型机上，在 Neon 的大型机应用集成产品中起基础设施的作用，它提供了一个处理 Web 服务和实时事件的平台。“Shadow RTE 的绝妙之处在于，你可以以 Web 服务的方式调用数据。但是如果你没有进入 Web 服务的话，你也同样可以通过标准数据库类型调用同一数据。”Evelyn 说。

对大型机而言，Iona 公司推出的 Artix “在大型机上的功能与企业其他剩余资源上的功能是相同的，但是前者提供的是大型机运行的方式。”Hunt 说，要达到的目标就是创造包含大型机的企业连接。

“许多用户数据仍然使用大型机管理，因此，对于每个业务流程来说，用户数据库和用户信息的来源是具有决定性的;这类用户数据库中所提供的信息就是人们通过 Web 服务所提供的。假设你有一个在客户端系统和大型机之间的[IBM WebSphere] MQ(消息队列)连接;再典型一些，你也有一个针对这个连接的 COBOL 授权的消息格式。那么我们就可以在 MQ 队列的末尾创建一个 Web 服务，而不仅仅是替换以前的。这时，大型机虽然看起来没有发生任何变化，执行过程也没有改变，但是你已经可以通过 Web 服务进行访问了。我们对 Artix 的目标就是记录下这些缺口，进行分层并且替换。”

Hunt 说，对于企业服务总线(ESB)，Artix 采用了与方法无关的技术。“在一个企业中，一个企业服务总线(ESB)通过使用面向服务架构(SOA)原则将应用程序与其他的应用程序相连接。Artix 就是这样的一个接口，无论中间件层是你想要的哪一种。”

这一方法的好处，Hunt 说，“是更加先进的架构，可存取性更好，并且生产率更高。当你使用面向服务的架构(SOA)，并且有服务终端的话, 你就能够迅速地创建一个综合应用。因此你就可以摆脱大型机及 Unix, Linux 等操作系统的限制， 利用以前的资源，迅速创建新的应用。”

有了 Atix 和 Shadow RTE，大型机不再仅仅是 Web 服务的服务提供者，现在还可以是 Web 服务的使用者。“因此通过大型机你可以进入到 Web 服务的领域。”Evelyn 说。例

如，他说，大型机应用可以建立一个 Web 服务调用，从而使非大型机平台能够应用于一个业务流程。

ESB 型的方法还可以帮助移植一个大型机系统。Hunt 说，“Web 服务为移植系统提供技术，使其不会遭到很严重的破坏。”

进一步，Hunt 意识到需要让使用大型机的人们对 ESB 更加坚定。“典型地，第一个问题是围绕安全和性能;我们已经有很有说服力的答复。使用大型机的人们会首先考虑一些问题:我如何维护?如果我改变服务接口，会发生什么事情?我如何管理不同的版本? 这些问题是那些缺乏技术但是成熟的人们所不会考虑的事情。”

(作者：Colleen Frye 来源：TechTarget 中国)

事件流程处理与灵活企业应变

对于灵活的追求促进了近来日渐增多的对面向服务的架构(SOA)的采用。而且现代 IT 的集成架构的表现也正在逐步改变。过去的技术孤岛现在已经被企业服务总线(ESB)技术连接起来。企业服务总线为网络主干、通信、中介以及服务容器管理等 SOA 所需要的因素提供支持。每个综合软件厂商都在他们的产品中提供某些种形式的 ESB，ESB 事实上已经上升到了面向服务的应用软件的统一标准的地位。但是，IT 集成架构的下一步又会演变成什么样呢?

下一步将会是一种新的类型的软件，它被称为事件流处理(ESP)。ESP 已经被软件厂商和分析家成为下一个“重要的东西”，因为他能够帮助 SOA 集成架构变成智能化的和可响应的。由于 ESP 能够让业务不仅了解过去业务的状态，还能得知现在的业务状态，所以他能够让业务分别考虑业务运作和 IT 基础架构的有关事宜。

但是，难道 ESB 及其组件已经不能发布事件了吗?所有的 SOA 的基础元素都不能发布事件处理事件了吗?回答是：不。当今，ESB 及其组件也是有事件处理的能力的。不过，他们并不规定对那些已经被他们的服务发布的事件做哪些处理。而这恰恰是 ESP 的关键价值所在。

ESP 能够让一个事件驱动的 SOA 程序去解释事件模式(如果 C 紧跟 B，B 紧跟 A)，时间上(4 秒钟以内)和空间上(10 英尺之内)的事件的关系，而且，是能够实时的完成这些的。这意味着允许一家企业不断的实时的分析关键绩效指标，实时的识别威胁和机遇，并立即实施。这些能力需要一种新的计算方式——流计算——能够在事件驱动的 SOA 与实时的商务视角之间传递完整事务中缺少的环节。

ESP——用时间约束和因果关系找到 ESB 的事件模式

为了描述 ESB 和 ESP 是如何在一起工作的，让我们来用一个例子来解释：信用卡欺诈检验。目的是监督系统内的购买活动以及捕获那些可以被实时分析检验是否是欺骗活动的授权请求。这些活动充分的展示了事件处理范例的三个阶段：1 监督 2 分析 3 处理

首先，我们需要那些 ESB 中发生的事件有电子化的接口。源事件流，用下文的 A 商人表示，而商人 B 则代表通过 ESB 的通告传递给事件引擎的购买活动事件。

其次，我们需要制定一些遵照那些事件的规则。由于 ESP 引擎处理事件是异步的，事件可能源自不同的地方，而且类型也可能各异，接收的顺序也可能差异很大。事务处理语言(EPL)使用事件属性、事件发生的时间以及事件中任何能推断的因果关系作为他的基础元素，而不是将数据的结构和 SQL 的关系代数作为基础元素。一个 EPL 处理查询是部分的——例如，当 A 和 B 都是真的的时候，那么如果 C 在 N 秒内发生了，那么采取措施。实时的检测事件模式能帮助应用程序识别即时发生在业务中的事件到底是威胁还是机会，例如实时买卖股票的机会，自动操控专业化生产车间的机会或者是检测信用卡欺诈的机会。下文我们将用一个例子来解释基于事件的规则。

这段 EPL 代码以一个"ON credit_payment (user)"的事件滤镜作为开始，这个语句命令引擎去监视 ESB 中的那些表示信用卡交易的事件。随着事件通过 ESB，这个事件模式就被部分确认了。接着，那句"FOLLOWED-BY"命令 ESP 引擎去监督同一个用户后来的信用卡支付事件。如果在两分钟内发生了三次交易，那么这段代码将会将其鉴定为潜在的欺骗活动。尽管上文没有描述过，一个包含在空间逻辑的警告也许会成为所有模式之外的地域性购买交易的标记或者当出现购买地域的地理距离暗示至少有一笔交易是不合法的情况时，它也会成为后续交易的标记。这些警告阐明 ESP 的最核心的概念：因果关系的推论。ESP 已经从发生欺诈的交易的可能性的关系来判断。

语句"WITHIN"阐明了 ESP 的另一个核心的概念：时间。在这个例子中，如果第三笔信用卡支付事件并没有在第一笔交易后两分钟之内发生，那么这个活动就不会被标记为潜在

的欺诈。检视到此结束。在流计算中，基于实时的原因，灵活的企业中，任何单独的事件对于业务重要性而言的重要性都是显著降低的。遵照事件行事的机会通常是极为短暂的。除非事件处理框架能够迅速的识别出其意义并做出响应。由于后续事件使得外界环境的变化以及其他因素的缘故，利用这种情况的机会是转瞬即逝的。

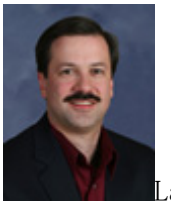
最后，我们来看一看 ESP 第三个关键概念：处理。信用卡欺骗检测应用系统等自动化系统通常在模式被检测到的时候马上调用事件驱动处理。在这个例子中，当前的交易请求被拒绝了，并且通过在 ESB 中发送一个新的起源事件把该帐号标记为欺骗管理行为。

结论

随着企业服务总线不断成为企业 IT 集成架构的主干，它提供的事件流使得实时的观测成为现实。流计算以及 ESP 工具能够提供检测时间、因果关系以及事件中的基于空间的 ESB 流中的模式的能力。通过利用 ESB 规格化的集成架构整合 ESP 规则，企业会成为真正灵活的企业。

(作者：Mark Palmer 来源：TechTarget 中国)

企业服务总线 ESB 的数据服务



Larry Fulton 作为 Forrester Research 分析师是研究企业服务架构的专家。他精通面向服务的架构（SOA），SOA 治理，enterprisearchitecture 和企业中间件。Larry 主要研究领域是 SOA 治理，SOA repositories，和企业服务总线（ESB）。他是 SearchSOA.com 的数据服务专家。

问：有些企业服务总线增加了 MDM/数据服务功能，例如（Sun, WSO2).你认为这是 ESB 应该做的吗或者应该在服务总线外部处理该功能？

答：ESB 不断演进，为连接并管理业务服务提供工具，不论我们所讨论的服务是否是由 ESB 提供主机服务的。“外在”功能的功能（就像业务规则引擎，BPEL，或者复杂的数据处理）或者使用 ESB 流组建的复合服务，JBI, SCA（或者一个组合）。

访问许多和数据相关联的功能是人们所期望的也是必然的趋势。这些与数据相关的功能支持 SOA，通过企业服务总线可以获得。我要补充的是，不管该功能是出自企业服务总线供应商之手还是其它的什么人之手，从理论上来说这些都是真实的，这就意味着希望供应商继续提供扩展 ESB 功能的方法以便为这类事物提供无缝的整合。

供应商越来越支持“hot pluggability”这样你就能添加，升级或者在需要时去除功能如果供应商能够让第三方也能够使用该功能就更好了。就像支持 MDM 和数据服务也是 hot pluggability 的。

机构要解决的问题如下：

1)当一个团体拥有了 ESB 而另一个团体拥有了数据集成时，如何来解决机构问题呢？从长远考虑，，从某种程度上来说，普通所有权是必不可少的，因为人为地区分这两个领域不会使事情变得更简单。

2）不管是在内部还是产品本身，如何处理独立的 ESB 演化途径和数据设施。你可能想利用那些对自己有用的开发项目，不论供应商希望你使用哪一个开发项目。

3)如何将这些功能结合到一起。大多数机构已经（或者将要）令从众多供应商的得来的 ESB 多样化。这是由于在项目、部门以及机关层面做出的决策所决定的。或者是因为特定产品要完成像 B2B gateways 这样特定的角色，或者是因为已购买的程序将包含 ESB，并使其成为集成和用户化功能的一部分。ESB 的整合不断完善，但是还不够完善。当你添加了较小标准功能时，和那些在 MDM/数据空间出现的事物一样，在整个企业的众多 ESB 中使功能可行本身就是一个挑战。

(作者：Larry Fulton 译者：杨君 来源：TechTarget 中国)

重造 Java ESB:JBI 与 ServiceMix

Java 的进展都是围绕着 JSR 形式的规格说明书进行的。最近，这个家族中又新添了一个成员，那就是 JBI(Java Business Integration)。它是一种企业服务总线(Enterprise Service Bus,ESB)，用于形成一种关键基础设施片段，使我们能够用 Java 实现面向服务的架构。我们将在本文中探讨 JBI 有关概念以及一种名为 SeviceMix 的开源实现。

JBI 的主要目的是提供一个基于服务的平台作为对现有 Java/J2EE 平台功能的扩展。由于 Web services 已经实际应用于 J2EE 中，而且 ESB 和 SOA 等术语与其说是技术推动力倒不如说更是市场概念，所以让我们一起来深究一下到底什么是 Java/J2EE 中所谓的“基于服务的平台”。

当前的 J2EE 部署都运行在一个基础上，那就是应用服务器。应用服务器本身由两个独立的部分组成——Servlet 容器和 EJB 容器，它们分别用于部署 JSP/Servlets 和 EJB 构件。在它们中的任何一个，你都能使用 Web services。但是，在任何环境中以分散的方式使用 services 是很困难的工作，而 JBI 的目的就是为完成这个任务提供一个专门的环境。

JBI 的最底层是一个容器，它与 J2EE 中的容器一样定义了自身的部署构件。在我们深入之前，让我们先详细了解一下 JBI 的主要构想。

首先，它关注了 Web services 的核心部分:在终端之间传输的 SOAP 消息/信封。这种数据段或标记能够包含被服务于很多应用的信息。不仅如此，根据发送端或接受端的不同，它还能协助把某个业务逻辑与数据适配。

现在，回过头来看看今天的 Web services。在 Web services 出现之前，对于很多企业应用来说，使用面向消息的中间件系统或者 MOM 实现异构系统之间的通信已经足够了。Web services 的出现同样是处于这个目的。用 MOM 的基于消息的架构和 Web services 很类似，被服务的数据也能在应用之间进行无缝操作。

这种场景可以被应用到典型的 J2EE 环境中，并通过 JMS 或 JAX-RPC 等技术进行服务。这种做法对简单部署是可行的，但正如前面提到的，当进行很复杂的设计时，现有的容器则显得很难用。于是 JBI 试图解决这个问题。

JBI 提供了一种正规消息路由器(Normalized Message Router,NMR)，说白了，就是一个地点。在这个地点，所有基于消息的数据片段——SOAP 片段、MOM 消息、HTTP 数据或其它信息——被聚合、集中、应用到业务逻辑、传输，如果有必要则被转换成其它格式并随后被分派到最终目的地。

先前的描述说明了为什么 JBI 被称为 ESB。它很适合企业级应用，因为它通过一种总线型架构的基于消息的手段到达了适应大范围的消费者和提供者的目的。现在，让我们看看除了 NMR 还有什么构成了 JBI。

和 JBI 环境直接交互的是两个部分，JBI machine 和 JBI binding。JBI machine 定义了部署构件以及在环境中管理它们的方式。本质上，它是提供商设计的黑盒，用于在 JBI 中支持他们自己的模型。另一方面，JBI binding 则被环境通过专门的业务协议与外部世界进行通信。

现在，再让我们把这些概念放到真实的 JBI 实现 ServiceMix 中。ServiceMix 是完全用 JBI 的思想开发的一个开源项目。它可以支持多种绑定，包括 HTTP、JMS、平面文件以及 RSS，还提供了一系列类似 BPEL、Schema Validation、JCA 和缓存等服务。

与 ServiceMix 协作的还有基于 JMX 的管理接口以及用来与 JBI 构件通信的对本地客户端 API。尽管本文不对 ServiceMix 功能做进一步说明，但你可以很容易地下载该软件然后继续学习。下面，让我们来看看一个可部署在 ServiceMix 上的真实的 JBI 构件，这样你会对它有点感性认识。

Listing for JBI Components - File Sender & File Receiver.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:my="http://servicemix.org/demo/">
<!-- the JBI container -->
<container id="jbi">
<property name="useMBeanServer" value="true"/>
<property name="createMBeanServer" value="true"/>
<property name="dumpStats" value="true"/>
<property name="statsInterval" value="10"/>

 <components>
   <!-- Write files to the outbox directory -->
    <component id="fileSender"
               service="foo:fileSender"
               class="org.servicemix.components.file.FileWriter">
    <property name="directory" value="outbox"/>
    <property name="marshaler">
    <bean
       class="org.servicemix.components.util.DefaultFileMarshaler">
    <property name="fileName">
    <bean
      class="org.servicemix.expression.JaxenStringXPathExpression">
```

```
      <constructor-arg value="concat('sample_', /sample/@id, '.xml')"/>
      </bean>
      </property>
       </bean>
      </property>
   </component>

   <!-- Look for files in the inbox directory -->
    <component id="filePoller" service="foo:filePoller"
          class="org.servicemix.components.file.FilePoller"
          destinationService="foo:fileSender">
      <property name="workManager" ref="workManager"/>
      <property name="file" value="inbox"/>
      <property name="period" value="1000"/>
    </component>
   </components>
 </container>

  <!-- the work manager (thread pool) for this container -->
  <bean id="workManager"
      class="org.jencks.factory.WorkManagerFactoryBean">
      <property name="threadPoolSize" value="30"/>
  </bean>

</beans>
```

JBI 规格说明书本身并没有定义构件应该如何编码。请注意，ServiceMix 用 XML 风格的语法定义了两个核心构件，关联到特定的类和属性。在把这种结构部署到 ServiceMix

以后，环境就会创建一种查询构件，负责从系统和其它构件中读取存档信息，然后把读取的值返回到文件系统。

该文件系统是 JBI 中非常基本的场景，因为信息就是从 JBI 环境(NMR)原样集中的，然后被返回到同样类型的绑定(文件系统)。但是很明显，一旦 JBI 环境有能力把任何业务逻辑适配成数据，例如 BPEL 或 schema validation，然后重定向到其它类型的绑定，例如 HTTP 或 RSS，那么就不只是返回到文件系统了。

正如你意识到的一样，当一用到 JBI，操纵及分发数据的可能性就很大，但即使 JBI 用 Java 的方法实现了 SOA，JSR-208 规格说明书还是受到了批评。IBM 和 BEA 是 Java/J2EE 领域的两个主要推动者，他们就不支持这个新项目。尽管如此，还是有像 ServiceMix 这样的小项目开发者和一些闭源开发组织打赌，一个培育良好的 JBI 市场将和 J2EE 的情况差不多。

假如你正在 Java 中使用 Web services 或者为了复杂的集成任务使用基于消息的设计，你应该对 JBI 的进展特别关心，因为它能为执行这样的任务提供更健壮的平台。就算你对 JBI/ESB 是否能加入 J2EE 协议栈还存有怀疑，继续关注还是有帮助的，因为它可能会成为应用服务器领域像 Spring 那样的事实上的“第三容器”或轻量级的容器扩展，为 Java 提供 SOA 能力。

(作者：Daniel Rubio 来源：TechTarget 中国)

Web 服务完善后是否还需要使用 ESB

Thomas Erl 是 SOA Systems 公司的创始人。该公司是一家企业解决方案提供商，专注于策略性 SOA 咨询和培训服务。Thomas 自 2004 年开始，一直是世界上最畅销的 SOA 作者。他所著的两本书《Service-Oriented Architecture: Concepts, Technology, and Design》和《Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services》受到广泛好评，主要探讨的是现实生活中的 SOA 集成和迁移问题，以及提供一种主流的 SOA 方式。Thomas 是 OASIS 组织的投票会员，在相关的研究领域非常活跃，例如 ServiceOrientation.org 和 XML 与 Web 服务集成框架(XWIF)。他出版了无数文章，包括 Web 服务期刊，WLDJ，LooselyCoupled.com 和应用开发趋势（Application Development Trends）。

问：我正在考虑广泛实行一家大型 ESB 提供商提供的 ESB(企业服务总线)解决方案，但是这时候，我产生了一个疑问，如果我们的所有应用程序都有 Web 服务的功能，那么 ESB 是不是就没有用了?同时，一时间所有的商品提供商都支持即开即用的 Web 服务接口，我是不是不再需要 ESB 了?那么 Web 服务自己能解决所有的问题吗?

答：把 ESB(企业服务总线)当作是某种中间件平台，它能够提供各种各样与集中运行时间处理和服务管制相关的功能。由 ESB 负责的服务也能被看作是 Web 服务，同时，只要服务契约是客户开发的(在观念上是标准的)，你的想法就是完全正确的，因为契约不需要与任何潜在服务实施细节进行耦合就可显示服务逻辑。这将使你拥有重新部署服务的自由，你可以将服务通过其他 ESB 平台传送，一切由你决定。但是这样的选择权没有使 ESB 被荒废。

市场上有各种 ESB 产品，同时每种 ESB 产品都有其自己的特性和缺点。我建议你仔细对比、研究一下这些产品再决定哪一种可以完全满足的你需求。这样做可以帮你衡量看 ESB 产品的花费以及 ESB 产品对你公司的影响适合成正比，投资是否合理。如果结果显示你只需要单独使用 Web 服务，而不需要使用 ESB，那么你当然可以选择这样做。当你进行选择时，记得要把眼光放长远－－如果你计划递送大量服务，而且这些服务需要被打包为复杂的组合结构，还有运行时间要求，那么建立一个稳固的、集中的平台是一个明智的选择，因为它可以帮你分担大半的运行和管理负担。同时，要注意不同的提供商用不同的方式使用 ESB，一些提供商提供一些类似于 ESB 的产品，不过它们却不叫 ESB.

（作者：Thomas Erl 来源：TechTarget 中国）

ESB 如何帮助企业重组应用程序和系统

随着面向服务构架(SOA)被人们的广泛接受和应用，越来越多的 CIO 正面临着在现有的业务和 IT 构架的基础上获得更好的回报的目标。为了成功达成这个目标，企业已有的应用程序和系统必须能够以同等的地位过渡到新的构架体系中。在这种情况下，CIO 和技术开发者所遇到的问题和面对一个刚开发的系统是完全不同的。

其实就在不久以前，我们还认为如果需要更新换代，就必须换掉所有的代码。那个时候我们还在开发不兼容 Y2K 的代码，而且也没有考虑到迁移这些应用程序会带来什么样的问题。可以确定的是，“临时”或短时期的开发所做出的努力会很快被更新和更好的代码所取代。技术的革新速度太快了，数据库管理系统、事务处理监视程序、应用程序服务器等等相关技术不断涌现，可以说，今天还是代码的东西明天就可能变成一个产品。而且一代又一代新的计算机语言不断被提出，不管是 COBLE、C、C++、VB 还是 Java，最终会变成每个人都使用的语言，而那些旧的东西会在历史上记录一笔，然后最终就不再改变。

然而在这个过程中，有些事情并不会像我们想象的那样结束。当然，某些系统会被替换掉，但是有一些更成功的系统则不会。一旦一个给定的业务区域成功的实现了自动化，不管是采用了什么解决方案，它都会一直运行下去。也许这个系统会需要更多的功能，或者是在性能、可扩展性、安全性、GUI、Web 浏览器等方面进行了一些增强，但是位于后端的核心功能，一旦被正确部署后就会一直待在那里运作。

随着 Web 服务的到来和它作为主流应用的不断认可，整个软件业最终达到了一个成熟的时期，在这种环境下，对于已投资的应用程序或系统将不会做太大的更改，而是被人们所接受，并且互相整合在一起。对于企业来说，他们不希望整个 IT 环境全部被替换掉，而是希望同时也要求现有的系统能够互相整合在一起工作，而且获得更好的效果。这样的要求，无论是对于企业高层还是对于开发人员来说，都意味着“在现有企业投资成果的基

础上获取更大的收益”。如果只是简单的将现有的应用程序和系统分散再用新的代码填补其中的空白，往往不能达到这样的要求。

将现有的系统整合在一起工作并且达到更好的目标和效果，实际上是企业服务总线(ESB)所扮演的核心角色。ESB 技术的特点能够帮助企业通过使用 Web 服务的形式保留现有 IT 资源的价值，并且使得现有的系统或应用程序能够更方便的互相通讯和共享数据，而这一切和底层的系统所使用的技术都是无关的。ESB 满足了业务和管理层希望在现有组件基础上结合有意义的体系构架的需要，并能通过这种方式构建、部署、管理和连接跨多个不同 IT 环境的服务。

然而，令人感到奇怪的是，现在还有一些集成软件厂商提倡将现有系统和应用程序分散然后各个替换，用这种方式来过渡到新的 IT 构架。实际上，通过这种方式并不能给企业带来更多的收益，因为业务的价值本身没有增加，而且还会导致 IT 部门陷入来自高层的压力之中，与其这样还不如让现有的企业投资成果继续运行，至少还能保证达到业务的底线要求。

实际上，现在的一个企业拥有足够多的 IT 构架。对于企业来说，真正需要的是将现在拥有的东西组合成新的应用程序。一个 SOA 体系提供了用于设计如何更广泛的重用和共享现有资源的完美蓝图。整个行业需要一个共存的解决方案，而不应该采用分散再替换的方法。分散和替换的方法的花销很大，这里面包括了软件和劳力成本，而且也在某种程度上降低了现有的系统的价值。如果非要采用这种方式，那么也是在 IT 企业现有的软件系统非常庞大和臃肿(很可能来自上个世纪)并且无法和现有的构架、网络和语言部署良好共存的情况下来使用。

ESB 能够保证为现有的系统增加合适的价值(意为着合适的代码和花销)并让它们一起协同工作。ESB 并不会通过分散现有的中间件、网络结构、消息发送语言或者应用程序来安装新的产品。为了增加效率，ESB 能够在已经投入的应用程序之间良好的工作，并且不会干扰现有系统，也不会向外暴露过多的企业机密(例如现有的数据和功能)。从这些方面

可以看出，ESB 是 SOA 满足现在的企业需求最现实的一条道路。作为基于 Web 服务的 SOA 之上的构架体系，ESB 仅仅为现存的系统提供增值的功能和服务，而并不要求企业投入更多的成本。

ESB 能够帮助企业在现有应用程序的基础上创建新的应用程序，它本身并不是一个系统。过去，IT 企业接受了通过大量投资带来回报的做法，但是现在，ESB 作为企业现有的 IT 资源合理重用的解决方案，这个更现实、更有前途的技术正等待人们的认可。

（作者：Eric Newcomer 来源：TechTarget 中国）

使用 ESB 简化 SOA 复杂性

为什么在面向服务的景观图中又加入了一个移动的部分?难道对面向服务的应用的管理还不够复杂?引入企业服务总线（ESB）的原因和许多年前选择企业应用集成策略的原因是一样的。

在 SOA 实现的早期阶段，当目录仅仅由一个或者两个基于项目的服务组成的时候，ESB 看起来是英雄无用武之地。但是幸运的是，如果在企业中采用 ESB，那么服务的部署会被加速。任何一项策略都被要求能够提供随需应变的扩展性，可靠性和足够的性能等特点。从构架的角度，使用一个合理的原则避免服务混乱不失为一个好的想法。

随着企业 SOA 的逐步成熟，业务功能会从各种源头被挖掘和发现出来。这些服务的提供者可能是遗留的应用，第三方软件包或者主要解决方案提供的功能。虽然理想状态是所有这些服务都使用相同的技术，但是现实情况证明这是不可能的。很有可能 Web Service 标准仅仅是使用的技术中的一个而已。

高级的 SOA 功能，比如服务编排(archestration)，自动业务流程，事件驱动架构和复杂事件处理，都依赖于健壮的企业应用集成架构。

将这些注意事项牢记在心，判定带有 ESB 的完整功能的 integration broker 的标准就很清楚了。

可靠性和可用性

由于多层应用的出现，对分布式软件中所有移动部分的可见性是一项重大的挑战。由于消费者完全从应用面向对象开发中分开，服务的消费者必须能够发现服务并且预计服务

的可用性、质量和性能特征。ESB 能够处理服务注册，同时有助于兼容 SLA(服务等级协议，service level agreement)。为了监控服务的健康状态，就必须清楚该服务对其他服务的依赖，以及运行平台的状态。当问题被检测出来，就会发出警报和通告。许多 integration broker 的产品或者提供内建的监控功能，或者，更常见的情况，提供钩子(hook)给监控工具以便获得更全面的可见性。这些工具可以提供审计，日志和异常处理的通用服务。

扩展性

基于总线的架构提供了对服务的真实的部署拓扑结构的一种抽象。增加 Web 服务器、应用服务器、第三方软件、遗留应用和数据库实例——服务描述中功能的真正执行者——的处理能力，将不会影响到服务的消费者。实际上，工作可以进一步委托给联合的总结，这样 ESB 本身可以按照最有效的利用硬件和网络资源的方式来部署。

安全

已经实现的服务不应该承担管理什么用户在什么情况下能够访问服务的重担。用户的角色可能随着时间而变化。这种横切(cross-cutting)的功能应该处于 ESB 中的各种服务之外。

延伸性和敏捷性

随着业务适应于变化的时代和新的机会，业务流程同样会变化。对于一个成功的 SOA 实现来说，对灵活性的规划非常关键。转换应当在尽可能的靠近往来于商业信息模型的集成端点处完成。这种方法可以便于流程的灵活自动化，组合服务的编排(orchestration)和将流程相关的商业规则从服务的商业功能中抽取出来。

总之，功能完善的 integration broker 会提供核心的 ESB 功能，以及其他的一些异构的服务拓扑中必须的功能。这类技术能够为复杂的结构提供一个“简化的”外壳，否则这种简化需要在每个应用或者服务里面取实现。

(作者：Maja Tibbling 来源：TechTarget 中国)

超越 ESB：下一步 SOA 难题

不久以前有一些比较聪明的做法，那就是脱离企业服务总线(ESB)来配置 SOA。你可以将 ESB 加入到强化现有的一系列已经存在的应用程序中去，从头建立一些服务，然后再将他们串连起来，这样你就完成了 SOA。

事实上，最初的 SOA 活动，就是这么进行的。企业要处理相关的优先数量的服务，配置给他们相关的有限的方法。IT 部门只是进行“SOA 试验”，花一些时间弄明白哪些是需要的而哪些是不需要的。经过一些试验，在级别分割和申请使用上，SOA 就被采纳了。这些很少会被斟酌。

但是， SOA 也在不断的成熟和发展。企业从 SOA 中得出结论，认为 SOA 可以帮助企业带来新的动力和在现有的系统上创造新的价值。SOA 可以使服务的申请和复用变得简单，从而促进生产的发展，同时降低了成本。

但是，采用 SOA 虽然可以带来利润，但是企业需要摒弃以前的一些观点和看法，重新定义大多数早前的 SOA 配置，来适应在意识形态上的变化，而这种意识会影响企业和其流程。

如果缺乏适应 SOA 配置结构化的巨大转变的心理准备和认识，组织往往会陷入“SOA 恐慌” 的危机之中。不幸的是，这样的认识往往可望而不可及。早期的 SOA 试验在各操作领域和商业单位中建立了“独立的 SOA”系统的环境。

随着 SOA 深入企业内部，将面对由被复制在不同分区中的服务对应得唯一的可能性，这是展示 SOA 主要益处的一个方面，即服务的重复利用。其他的方面，用离散的方法来对

SOA 进行配置，需建立针对处理现存的服务和所支持的 IT 资源的无效管理的成熟环境，即处理那些在 IT 部门内不需要存储的服务。

帮助引导 SOA 进入下一个阶段，使之不仅成熟，而且有效，需要以下方式：以企业方面定义的调配内涵来管制你的 SOA 发展，强调应当强调的流程和任务，以及处理新的 SOA 动力学。

管制看起来很好，但是当你静思该如何做的时候，就会发现那真的很迷茫。一些人会说很简单，“将这些工作注册，然后就行了。”可是，“简单”这个词好像是相反的意思，每一个从事过 SOA 开发工作的人都会这么说，告诉你那个东西很简单，但是这个词很少用于描述一个流程。

这并不是说注册工作不重要。明白你现在在此企业中所处理的服务和怎样完成这些服务是第一步工作，可以来避免“SOA 恐慌”。但是那里有很多工作要去做，以明确你的管制工作。存储功能可以作为一种好的 SOA 管制来考虑。

然而，你的注册功能就像目录或电话簿一样，可以在其中找到服务的位置，而你的 SOA 存储功能是一个针对全企业的完全有效服务目录，那么代替对服务种类和服务位置的辨识，存储功能可以使企业对现存的有效服务进行适当的使用。筛选和存储对所有相关的服务的信息的存储，不仅仅是契约，还包括政策，实现的产物和依赖分析等。通过由组织内部定义的角色和职责对这些服务提供不同的显示，那么有效的存储就可以加大服务的重复使用率，而更好的控制调配和 SOA 策略中的各个不同部分。

什么可以使一个 SOA 存储唯一呢?即 SOA 存储包括 SOA 生命周期的所有方面，从设计和发展到发展和管制。在一个逐渐成熟的 SOA 中，这可以有效的为你提供所有需要的 Web 服务。

例如，某个存储应该保存服务的概念，基于应用的细节，包含服务使用的规则和更加详细的信息，如针对指定用户的每一个服务。简短的说，存储应该包含为整个 Web 服务所准备和重新调配的必要信息。通过手工获取这些功能，明确它们的作用，更简单的分析在 SOA 环境中的服务，检查出存储中与实际流程相悖的地方。

SOA 领域在成熟和变更，如果 SOA 成功地话，企业也要变更。ESB 足以运行的地方，SOA 就要做的更多。公司需要思考他们的 SOA 调配策略和帮助他们实现目标的技术。对于整个企业，正确的存储选择可以给客户带来加强已有的服务，调配和管理周期的方法。

手工完成这些功能，因审视被可以带来利润的 SOA 调配支持的商业灵活性和适应性使 SOA 变更时，企业最好可以重复利用已有的服务，理解现有的多种服务的关系，重新调配和准备服务。

（作者：Steph Bacon 来源：TechTarget 中国）

Forrester:视 ESB 为 SOA 的本质

尽管关于如何把必要的企业服务总线转化为面向服务的架构存在激烈的争论，但是就在最近 Forrester Research 公司在报告中说他们认为持续采用 SOA 能很好的体现 ESB 的思想，他们把它称为“SOA 的主要切入点”。

该报告的细节分析了两种独立 ESB 市场、处于领导地位的公司以及使 ESB 对有集成问题的公司价格上的吸引力。Forrester 公司总共调查了 116 家公司，报告显示 77%的大企业、51%的中型企业和 46%的小企业将主动在今年年底前实现 SOA。评估认为三分之一的“基础设施决策者”在未来 12 个月中会增加他们的 ESB 部署。

Forrester 公司分析师 Mike Gilpin 说:“尽管人们还不十分确定如何构建出一个完整的 SOA，但他们已经知道要解决集成问题，而 ESB 正好能帮助他们解决该问题。”

报告把客户市场分成两个独立的群体。一个被称为“保持简单”群体，另一个被成为“即刻整体部署”群体。

“保持简单”群体想要一种低廉的、基于标准的 Web 服务编排工具并在此之上构建健壮的 SOA。Gilpin 认为价格正在成为 ESB 的一大卖点，它使得公司用相对少的投资就能启动 Web 服务，于是他们就可依靠必要的技术去构建完整的 SOA。

Cape Clear Software、Sonic Software 以及 Fiorano Software 等公司就是采用了以上这种做法。

他说:“实际上，ESB 的成功为导致集成软件方面费用的降低。因此尽管公司迁移了更多的单元，但费用却降低了。”

而“即刻整体部署”群体则希望能一次性解决集成方面的所有问题，希望得到在服务证明周期中负责细粒度控制和服务过程管理的工具。

Gilpin 相信这两种不同的市场会导致对两种不同类型 ESB 产品的需求，即一种轻量级的产品以及一种全面的 SOA 集成与控制产品。报告显示，所有 ESB 倡导者以及企业应用集成的推崇者都有目光投向了全面产品市场，于是把轻量级产品的市场留给了 IBM 不久将要发布的 WebSphere ESB。

Forrester 公司高兴的发现经常被分析师团体称为遗留中间件的 EAI 厂商正在发生转变。他们在 Web 服务基础设施的服务仲裁以及控制/变更方面变得很有竞争力。根据报告显示，EAI 的推崇者对扩展市场贡献巨大，但只有 Oracle Corp 和 webMethods 公司有能力在价格上取得有竞争力的地位。

处于领导地位的公司包括 Cape Clear,、Sonic、Fiorano、Iona Technologies、Oracle、webMethods、Tibco Software 以及收购了 SeeBeyond 公司的 Sun Microsystems。今年夏天，BEA Systems 公司作出了两次领导者的举动，一次是发布 AquaLogic ESB，另一次则是推出 WebLogic Integration suite。IBM 由于最近才刚推出 ESB 产品所以没有参加调研。

Gilpin 并不认为像 BEA、Oracle 以及 IBM 这样的平台提供商能够引起市场两极分化，因为集成问题不应该与应用程序运行在 SOA 内部哪个地方紧密耦合在一起。

他确信与产品缺少结合会对市场造成挑战。

他说:“人们都认为 ESB 并没有被标准良好的定义。它还处于一种类似于 J2EE 之前的应用服务器市场的状态。”

他看好像 Apache Synapse 这样的项目，该项目被认为会为 Web 服务仲裁定义标准的方法，而那会是解决问题的潜在手段。

Gilpin 认为控制已经是 ESB 市场没有解决的最大问题。SOA 注册提供商早已注视着该领域，而他希望这两个领域能够通过融合或协作的方式走到一起，也可能是通过 XML 网络的方式。

他说:“SOA 太大，以致于已经不可能靠一个提供商来实现全部的内容，即使是 IBM。今后很可能会靠很多提供商的团队工作才能形成一种生态系统。”

(作者：Michael Meehan 来源：TechTarget 中国)

Sun 紧抓 ESB 和 JavaEE 实现 SOA 应用

本次访谈中 Sun 公司 SOA 产品负责人 Kevin Schmidt 把之前的 SeeBeyond 的技术整合到 JavaCAPS 的过程，Java 商业整合(JBI)的未来，持续的对 JavaEE5 的批评的反映以及新生的 Java EE6 的工作……

在这次的 JavaOne 大会上，TechTarget 对 Sun 公司 SOA 产品负责人 Kevin Schmidt 进行了采访。在第二部分的访谈中，他谈到了 Sun 在把 SeeBeyond 并购到 Java 组成应用程序平台组件之后在技术方面的不断整合，以及一些关注的重点。他还提到和那些开源软件组织许诺的与商业产品的平衡问题。他还讨论了在关于 Java 商业整合(JBI)的规范上 Sun 做到了什么和失去了什么，以及 Java EE 这个平台在面向服务的架构的世界中的前景的问题。

在第一部分，他谈过了有关 GlassFish，开放式的 ESB，Java SE，JavaFx 脚本以及其他和 SOA 和 Web 服务相关的 Java 的内容。

我们把目标放回 Sun 公司收购的 SeeBeyond，SeeBeyond 公司的代码里面有多大部分是属于开源组织的呢?

其实我们并没有把 SeeBeyond 公司的产品交给那些开源代码组织。 我们正在做的事情恰恰是我们所有的新的开发都是在开源代码组织里的。在开放的 ESB 中，我们正在将 SeeBeyond 从事的以及 Sun 公司已经在 JBI 和 JSR-208 上做的一些研发进行了整合。因此我们一直在从事开源的工作。而且那也是我们在整合技术中创新所在。我们并没有在这点上做出关于将所有的从 SeeBeyond 得到的代码交给开源组织的计划。在这一点上有两部

分。一些人在开源代码组织和自由获得的概念上发生了混淆。我们前进的方向是让他们可以自由获取，但是，很有可能的，我们并不将所有的得到的代码交给开源代码组织。

那么对于你们来说，要想代码能够完全获得的话，要经历多久?

这也是我们正在讨论的问题。很有可能今年晚些时候就能实现。不过确定的是，一定能在开放的 ESB 中获得。

如果 你在开放的 ESB 中能够很好的工作的话，那么是不是意味着你就会很好的使用 JavaCAPS 呢?

这也是我们在我们的战略中经过充分考虑的问题。我们的战略是，我们支持的和销售的商业产品都将是 JavaCAPS。我们在开放的 ESB 中开发的，都将是开源的，但是我们将把这些能够给我们的客户提供价值，而且可以作为 GA 产品进行发布的技术和组件合并到 JavaCAPS 里面去。

用批评的眼光去观察 JBI，你认为还有哪些是你还没有做到的呢?

大概有两件事情是我们现在还没有做到的。一件是到目前为止， 我们还没有将一件产品推向市场。我们可以指向 SeeBeyond 这次兼并，但是这并不是借口。我们需要得到一些门外的东西。一些和他们相关的东西，部分上受了我们没有尽快的将我们的产品推向市场的影响。我们还没有做到像我们宣传所要求的那么好。毫无疑问的是，有一些其他的软件厂商建立了他，他们在其中看见了商业价值。但是主要的软件厂商还没有支持他。我们将继续和那些厂商进行讨论，而且很有可能的在某种程度上获得他们的支持。这两点是我们应该做的更好的事情。

为什么 JBI 依然是重要的?

JBI 依然很重要的原因是因为他标准化了整合平台的管道。这是有非常重要的价值的。将其解释给用户并不是一件容易的事情。但是，拥有这种标准将给用户更多的灵活性以及更多的选择。通过 JBI，以及规范化的管道，它允许在有很多可供选择的软件厂商的系统中进行开发。

十年以前，如果你想要做集成的话，你必须购买单独的能帮你实现目标的产品，而且还要解决他们如何在一起工作的问题。集成软件厂商看到了这部分的问题，认为这部分问题是要通过他们提供一整套的集成软件包来解决的，但是，每个软件厂商提供的产品都来源于截然不同的形式。他们都在某些领域功能比较强大，因此，经常出现的情况是，集成工具包并不能完成你想要的全部东西，你还需要某方面拥有 erp 特长，又有很强的独立性的 best-of-breed 软件来补充你的集成软件包。这就是最后这种集成软件包是如何服务客户的，它将客户从软件厂商的限制当中解放了出来。

那么，是什么样的因素在驱动着 JBI2.0 呢?

JBI2.0 驱动着标准不断的前进，并且将我们在过去一些年里学到的有关可靠性和聚类的经验进行一些运用。而且，我们还相信，JBI 以及 SCA(服务组成架构，现在正在 OASIS 的标准讨论的过程中)都是值得称赞的。因此，我们想要保证的事情之一就是，我们所做的事情，一直是广受好评的。也许会有 JBI 可以利用的从 SCA 的元数据方面提取出来的东西。

我们进行了一些展望，一些工作是在 Java EE6 的基础上进行的。你可以告诉我们一些对这些事的评价以及 Java EE6 的时间表吗?

在关于 GlassFish3 的工作中是包含了一些考虑使用 JavaEE6 的东西。有一种你不需要拥有 EJB 容器的网络轮廓。他利用了一些更为先进的模型结构。至于时间表的问题，我想，他们将会在零八年末或者是零九年的某个时间会提到的。

那会出现这样的情况吗：那些在 Java EE6 上工作的人看上去好像对 Java EE5 提出了自己的反应，认为里面的由于成分过多而导致的膨胀进行了很多的批评?

肯定是有一部分这样的情况的。我们暂且不管这公平与否，现在的确有一种观点认为 Java EE 应用程序服务器实在是太沉重了。在某种程度上，事实的确如此。因此可以说，反馈情况的很大一部分是这样的。但我们的确要着眼于市场，观察人们究竟想要什么样的产品。在市场上的确有 Java EE 和 EJB 开发的一席之地，但是毫无疑问的，也有 Tomcat——不是一个完全的 Java EE 容器，也是一个 servlet 容器——广泛使用的现象存在。最清晰的概述是有一个有相当重要意义的采用。拥有一个支持不需要完整的 Java EE 栈的方法是大家现在正在关注的。

IBM 公司依然不支持 Java EE5 ，这是我们关心的一个问题吗?

有点是吧。我不知道具体的讨论或者是他们的计划。不过，我猜他们最后一定还是会支持它的。当然，现在 Oracle 公司和 BEA 公司已经支持他了。我认为，IBM 公司某种程度上是那种已开始并不支持，而市场需求最总会推向他们支持 Java EE 的企业。

说道 Web 服务栈方面，你也拥有一个，那就是 AXSIS，Jboss。我们先把微软放在一边，如果只有一个 Web 服务栈的话，生活不就能简单一些了吗?

其实我也是这么认为的。在这方面我们愿意和其他的厂商贡献和合作。在我们负责的 Tango 项目的部分当中，我们已经在开源组织里完成了。他也是 GlassFish 的一部分。而且已经是公开的了。其他的人正在使用它们，并把它们整合到他们的产品当中去。因此，可以说，我们拥有了这种合作的开始。我们已经和 BEA 公司和 Oracle 公司进行了合作。他们不会离开去做他们自己的事情去。他们看到了合作的价值。我也愿意看到这个价值。这将会是在 Web 服务平台之间拥有的意义重大的能够进行协同工作的能力。而我们和微软合作也是有原因的。就像你说的那样，我们并没有在那里拥有相同的栈，因为我们是在不同的技术上的，但是可以保证的是，在工作中，协同工作能力是关键的驱动因素。

你可以为人们描述一下 Mammoth 项目是什么吗?你可以提供一些具体的例子或者是具体的由其生成的东西吗?

在 Mammoth 项目中，我只涉及到了外围。这个项目的重点是放在我们软件基础设施产品，我们的认证产品，我们的商业集成产品上的。他们主要从事将服务进行整合，并提供能够让顾客使用我们产品和技术的服务。在 JavaOne 会议的发言中，你已经听到了治理随着越来越多的客户和企业建立自己的面向服务的架构而变得越来越重要。他们发现，随着你有越来越多的服务，你必须要有方法来管理他们，智力他们，确保他们的安全。这就是 Mammoth 项目早期阶段我们关键注意的问题。但是我们期待还有一些事情可以被确定下来，像什么创造一系列关于使用我们产品的最佳实践方法。我们对于这点是相当激动的。我们正在做这件事的一些简单的类型。这并不是我们和埃森哲工作的的事情。

如果我画一幅关于 JavaCAPS 的画的话，特别是在 SOA 方面你想让人们拥有的部分，在注册/仓库方面似乎是一片空白，你能谈一下这方面的情况吗?

Sun 公司拥有 Sun 服务仓库，但是，毫无疑问的是，有很多我们需要合作的或者是在上面操作的占有很大市场份额的其他公司的产品。我们将要保证我们能和这些产品合作，并提供相应的支持。

你们是不是正在寻找一个能够为 UDDI3.0 提供更好的鲁棒性注册/仓库支持?

是的，我们绝对是这样的。现在，我们并没有确定我们如何去做这件事，我们正在寻找最佳的做这件事的方法，而不论它是否是在我们拥有的产品上构建的或者是在一些 Web 服务组织的产品上构建的。在这件事上，我们正在运用一些在这方面能够提供一些有价值的输入的服务治理框架。

(作者：Rich Seeley 来源：TechTarget 中国)

独家专访：如何看待开源 ESB 和基于 REST 的 SOA？

开源 ESB（企业服务总线）的前景如何？是否有闭源 ESB 的应用空间或者能否采取某种开闭源混合的方式？在 TechTarget 采访 Paul Fremantle 的第一部分中，这位 WSO2 公司的共同创始人兼副总裁论述了这些问题，并介绍了 WSO2 公司 ESB 产品的核心----Apache Synapse 开源 ESB。在合作建立 WSO2 公司、开发基于 Web 服务标准的开源产品之前，Fremantle 曾担任 IBM 的高级技术人员，创建了 Web Services Gateway，而且带领团队开发并组装成 WebSphere 应用服务器的一部分。他作为团队的成员，也为 WebSphere Application Server 6.开发了综合服务总线技术。目前，他是 OASIS Web 服务可靠交换技术委员会（WS- RX）的主席之一，该委员会研究使用 SOAP 进行可靠信息交换的标准。他第一次参与开源的研究可追溯到最初 Apache 的 SOAP 项目。Fremantle 获得过牛津大学的数学和哲学硕士，以及计算技术的理科硕士。

新推出的 WSO2 Enterprise Service Bus 和你正着手进行的 Apache Synapse 项目有什么联系？

Apache Synapse 是核心运行的动力，而且如果我们想提高核心运行能力，需要执行的代码都在 Apache Synapse 中。我们并非是贬抑其他的核心代码，但根本上，核心运行是以 Apache 项目为基础。

那么 WSO2 公司的价值观是什么？

我们为 ESB 提供支持，无论其用于高质量的商业培训、支撑还是服务。我们有一个图形用户界面。这是一个完全基于网络的用户界面，允许用户对基础位置的 Synapse 进行配置、监控和管理。

这是一个基于 Ajax 的 Web 界面吗？

是的，它是一个基于 Ajax 的 Web 界面。它的作用之一是公开所有的 API 管理及服务，因此你可以将其与其他界面区别出来。

这么说来，这是一个将管理内置的 ESB？

正是，但它的源程序是完全开放的，包括管理控制程序也是。

还有没有其他的特点？

我们有两样法宝。第一，我们在常数存储器中编写信息，而不是套用存储库中的大型信息树或信息模型。不过在某些情况下，也必须在存储库中建立信息模型。信息不可能都被流化，但只要是可以流化的，我们就可以做到。第二，我们有一个完全无障碍的运输模式。因此，我们可以处理大量的通信连接，而不会用光所有线程或是受到阻碍。我们非常重视提高运行能力时运行的稳定性，同时也注重通过采用某种清洁、简单的模式提高其简易性。

一些厂商试图提供开源 ESB 和所谓的“闭源”ESB。你怎样看待？

当我在 IBM 工作的时候，我接触过开源和闭源共存的情况。我发现客户总是很难分辨两者之间哪一个更好，特别是在过去几年中，开源各项性能的品质不断上升。这是为什么我们没有推出企业版、标准版、免费版和共享版中任何一个版本的一个原因。我们只生产简单的开源产品，客户可以直接购买。我们倒认为这对客户来说更为简便。

将来会不会有闭源 ESBs 的应用空间？

我认为，有一些产品会非常适合用闭源 ESB。举例来说，有一些金融机构有超高的通信要求性能。如果每秒需要处理上百万条信息，就需要非常复杂的高度调试软件产品。而这个产品的市场可能是 30 到 100 个客户，也就是说它不是一个开放的市场。所以，如果我要生产这么一个产品的话，我不会使用开源。

但是另一方面，ESB 正成为解决问题的不二之选。即使是小公司也看到了 ESB 的好处。因此，我认为开源正在完全接手这类市场。所以，既然两三个高质量的开源产品能够很好的解决问题，而且有更低的获取成本，为什么要使用专利产品，限定于某一个特定的供应商呢？

在过去的一年里，应用于 SOA 的开源软件是否取得了进步？

是的。我认为我们的第一组软件很稳定。我们在 2007 年已经为 SOA 提供了一个更为坚实的平台。现在我们正在同更大的机构洽谈，比如说财富 500 强公司。他们表示正在认真考虑将开源用于 SOA。

除了 ESB，你觉得将开源应用于 SOA 方面，还有哪些新的项目或新技术特别能引起你的兴趣？

我们最近刚启动了基于 REST 的 SOA 注册项目。现在有开源的 UDDI 项目和基于 ebXML 的项目，但是仍然有很多人买很昂贵的专利产品。我发现相对于我们的那个项目，UDDI 和 ebXML 两者都是过于沉重和复杂的解决方法。因此我们回头去找最初的原理，等我们再回来看的时候，我们认识到，面对资料库/存储库时，Web 资源才是最根本、最重要的。这才是真正的管理资源。所以，这个想法引领我们采用了 REST 模式。这就是为什么我们要建设一个完全基于 REST 的资料库/存储库。

（作者：Rich Seeley 译者：Eric 来源：TechTarget 中国）

如何转变 EAI 到 ESB

 David Linthicum 是 ZapThink 执行合伙人。他主要从事 SOA 策略咨询，SOA 项目指导，风险投资顾问服务和 SaaS/Web 2.0 集成咨询。David 已发表八本著作，最新的是《Next Generation Application Integration》（下一代应用集成）。

问：大多数的 EAI 开发商抱怨他们现在需要做出改变以适应 SOA/ESB 模型。在你看来，有没有已经做出重大改变的出色的开发商呢?你认为经典的 EAI 软件需要做出哪些关键性改变才能转为 ESB 呢?

答：大多数传统的 EAI 开发商并不擅长 SOA。这是关于服务与面向信息的整合这两者孰优孰劣的问题。他们是完全不同的概念，并且最早的整合技术是为了信息整合而提出的。

ESB 只是采用了 Web 服务接口的队列系统，因此，大多数 EAI 软件添加这些组件之后便可以成为 ESB。

(作者：David Linthicum 来源：TechTarget 中国)

实施企业服务总线 ESB

Thomas Erl 是 SOA Systems 公司的创始人。该公司是一家企业解决方案提供商，专注于策略性 SOA 咨询和培训服务。Thomas 自 2004 年开始，一直是世界上最畅销的 SOA 作者。他所著的两本书《Service-Oriented Architecture: Concepts, Technology, and Design》和《Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services》受到广泛好评，主要探讨的是现实生活中的 SOA 集成和迁移问题，以及提供一种主流的 SOA 方式。Thomas 是 OASIS 组织的投票会员，在相关的研究领域非常活跃，例如 ServiceOrientation.org 和 XML 与 Web 服务集成框架(XWIF)。他出版了无数文章，包括 Web 服务期刊，WLDJ，LooselyCoupled.com 和应用开发趋势（Application Development Trends）。

问：我在考虑如何从 ESB 厂商那里获得好的解决方案从而大范围的应用 ESB。但是在我考虑的时候觉得，真的有必要在 ESB 平台上应用 Web 服务?这不正是将现有的应用系统的遗留问题完全曝露?就当前的厂商而言，已经有了很多其它相关的 Web 服务支持产品，那 ESB 是否又是真正所需的?那如果脱离了这个平台，单独的 Web 服务是否能够解决我所能遇到的所有问题?

答：考虑一个 ESB 作为一个中间件平台提供许多与中央运行时处理过程和服务管理相关的特性。由 ESB 主导的服务能够以 Web 服务的形式发布，并且只要服务合约是由客户定制的(理想情况下是标准化的)，你可以非常确信地认为你能够暴露服务逻辑而不需要将合约匹配到任何底层基础服务实现细节上去。这将给你充分的自由去将服务安置到另外一个 ESB 平台上或者其它地方上。而且这个可选择性并不会让 ESB 变得孤立。

许多 ESB 产品都拥有自己的一套功能和局限。仔细研究一下对这些产品的比较，从而帮助您决定哪种产品才能最好地满足您的需求。这能够使您更好地决定引入 ESB 到您的环境所带来的花费和影响是否是值得的。如果你发现你不需要使用 ESB 而且能够独立地部署服务，你当然选择不这么做。做决定的时候，最好保证你是从长期的发展角度考虑的——如果你计划把数量可观的将被组合到一起的服务发布到复杂的有运行时的需求的组合环境中，建立一个稳定集中的平台是一个不错的主意，这个平台将能够承担运行和管理所需的负载。同样，必须注意到不同的销售商在不同情况下所说的 ESB 是不一样的，甚至有些销售商提供的 ESB 只是没有标志的产品。

（作者：Thomas Erl 来源：TechTarget 中国）