论文分类号 TP391 单 位 代 码 10183
密 级 公开 研 究 生 学 号 2004542028

吉 林 大 学

硕 士 学 位 论 文

开源 ESB－ServiceMix 的研究与应用

The research and application of open source ESB - ServiceMix

作者姓名：刘磊

专 业：软件工程

导师姓名

及职 称：秦贵和 教授

学 位 类 别： 软件工程硕士

论文起止年月： 2005 年 5 月 至 2006 年 5 月

吉林大学硕士学位论文原创性声明

本人郑重声明：所呈交的硕士学位论文，是本人在指导教师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：刘磊

日期：2006 年 4 月 30 日

作者姓名	刘磊	论文分类号	TP391
保密级别	公开	研究生学号	2004542028
学位类别	软件工程硕士	授予学位单位	吉 林 大 学
专业名称	软件工程	培养单位（院、所、中心）	软件学院
研究方向	中间件应用技术	学习时间	2004 年 9 月至 2006 年 7 月
论文中文题目	开源 ESB－ServiceMix 的研究与应用		
论文英文题目	The research and application of open source ESB - ServiceMix		
关键词(3-8个)	ServiceMix，JBI，ESB，系统集成		

导师情况	姓 名	秦贵和	职称	教授，博士生导师
	学历学位	博士	工作单位	吉林大学计算机学院

论文提交日期	2006 年 5 月 1 日	答辩日期	2006 年 6 月 10 日
是否基金资助项目	否	基金类别及编号	

如已经出版，请填写以下内容

出版地（城市名、省名）		出版者（机构）名称	
出版日期		出版者地址（包括邮编）	

内容提要

企业服务总线（ESB）的出现，在中间件和系统集成市场上掀起了一个不小的浪潮，各个厂商都在提出自己的计划，推出自己的相关产品。JBI 规范的推出为这个正在兴起的市场提供了一个可以共同遵循的开发标准。

本文对 ESB 和 JBI 规范，以及其涉及到的面向服务架构（SOA）和 web 服务（web service）进行介绍，并通过一个基于 ServiceMix（一个基于 Spring、完全遵循 JBI 规范的 ESB 产品）实施系统集成的案例来详细介绍如果利用 ESB 实施 SOA，并基于此进行进一步的分析。

本文分五个章节，

第一章绪论简单介绍了目前在研究如何实施 SOA 的领域中出现的一种产品 ESB 和一个 java 规范 JBI，以及本文研究的意义和主要内容。

第二章对 JBI 规范和 SOA、ESB 以及这三者及其之间的联系进行比较详细的介绍。

第三章对开源 ESB 产品－ServiceMix 的原理和特点进行探讨，并介绍了 ServiceMix 中的轻量级 JBI 组件和标准 JBI 组件。

第四章对一个基于 ServiceMix 的系统集成案例进行详细的分析，按照从需求到设计、实现逐步说明了如何利用 ServiceMix 实施系统集成。

第五章对全文进行了总结，提出一些笔者的观点，并对该领域进行了展望。

关键词：ServiceMix，JBI，ESB，系统集成

目 录

第1章 绪论

1.1 引言

在 2002 年年底，Gartner 的一份报告预测 03 年在中间件市场长将出现一种新型的产品——企业服务总线（Enterprise Service Bus,ESB），并将逐渐成为企业实施 SOA 的基础构件之后，IONA、Sonic、Infravio 等公司就相继推出自己的 ESB 产品，同时 IBM、BEA 等行业巨头也加入到竞争的行列。开源组织 Apache 和 ObjectWeb，和一些中小企业也参与其中，或竞争或合作，在 SOA 领域掀起了一个 ESB 的浪潮。SUN 公司则期望利用在制定 java 规范中的领导者地位，通过领导指定 JBI 规范以参与到这个浪潮中。

随着社会的进步和对软件需求的提高，软件系统变得越来越复杂、越来越庞大，在软件开发的活动中，如何提高开发效率，降低变更和维护的复杂性一直是开发人员面临的重要问题。从机器语言到汇编，再到高级语言，编写一个同样的软件需要的时间越来越少，写的代码也越来越短；从面向过程到面向对象，程序越来越容易被理解。而设计模式和软件架构的理论也不断发展，是软件的模块与模块之间边界越来越清晰，耦合度越来越的，模块或软件本身的可重用性越来越强。

SOA（Service Oriented Architecture）即面向服务架构，最近几年来越来越热并不断发展。各大厂商在这方面也不断加大投入，推出自己的解决方案和相关产品。ESB（Enterprise Service Bus）企业服务总线就是最近两年一种遵循 SOA 框架的，解决软件系统集成问题的方案。IBM、BEA、Sonic、INOA 等公司都推出了自己的 ESB 产品和解决方案。但是在 java 社区（jcp）里（以上厂商都是它的成员），对 ESB 还没有一个标准、规范。于是 2003 年 4 月由 SUN、IBM、BEA 等公司发起，开始制定 JBI（Java Business Integration）规范，并于 2005 年 8 月通过了该规范的 FR（Final Release）版本。该规范的制定为系统集成领域的产品开发商提供了一个共同遵循的标准。

1.2 课题研究的背景和意义

在国外，在 SOA 领域提出 ESB 的概念已经提出了 3 年多，学术界，开源组织，软件企业都在这方面投入力量去研究开发，并且在 2005 年 8 月 java 社区发布了 JBI 规范的情况下，国内在这方面除了可以在网上查到一些介绍性的消息外，相关资料、学术成果等还比较鲜见，在这方面的研究和投入还不够。

研究 ESB 和 JBI 的意义在于其将成为实施 SOA 的主流构建，Gartner 预测在 2008 年将有全球将有近 7 成企业在构建应用系统的时候导入 SOA(服务导向架构)，ESB 将在实施 SOA 的过程扮演基础构件的角色。而我国经济处于高速方展的阶段，也是信息化建设的重要阶段，要想在全球新一轮的信息化浪潮中，占据一个有利的位置，就必须研究先进的理论，跟踪领先的技术。我国国内对 SOA 产品的市场需求将不断扩大。因此研究 SOA 和 ESB 不仅有学术上的价值，更是积极的现实意义。

1.3 本文的主要内容

笔者在 2005 年 7 月到上海汇软信息系统有限公司实习，参与到基于 ServiceMix 的应用集成项目的开发中，参加了项目的前期调研，系统设计，编码三个阶段。

主要工作内容包括：

1. 对 ESB 产品的调查
2. 对 ServiceMix 的研究
3. 对应用系统结构的设计
4. JBI 组件的开发
5. web 组件的开发

本文将先沿着 SOA－ESB－JBI 这个线路，探究一下 SOA、ESB、以及 JBI 规范，通过对遵循 JBI 规范的开源 ESB—ServiceMix 的调研，然后详细介绍一个基于 ServiceMix 的应用集成项目是如何实施的。

第2章 JBI 规范与 SOA、ESB

2.1 SOA

SOA（service-oriented architecture，面向服务的体系结构）是一个组件模型，它将应用程序的不同功能单元（称为服务）通过这些服务之间定义良好的接口和契约联系起来。接口是采用中立的方式进行定义的，它应该独立于实现服务的硬件平台、操作系统和编程语言。这使得构建在各种这样的系统中的服务可以以一种统一和通用的方式进行交互。

这种具有中立的接口定义（没有强制绑定到特定的实现上）的特征称为服务之间的松耦合。松耦合系统的好处有两点，一点是它的灵活性，另一点是，当组成整个应用程序的每个服务的内部结构和实现逐渐地发生改变时，它能够继续存在。而另一方面，紧耦合意味着应用程序的不同组件之间的接口与其功能和结构是紧密相连的，因而当需要对部分或整个应用程序进行某种形式的更改时，它们就显得非常脆弱。对松耦合的系统的需要来源于业务应用程序需要根据业务的需要变得更加灵活，以适应不断变化的环境，比如经常改变的政策、业务级别、业务重点、合作伙伴关系、行业地位以及其他与业务有关的因素，这些因素甚至会影响业务的性质。能够灵活地适应环境变化的业务为按需（On demand）业务，在按需业务中，一旦需要，就可以对完成或执行任务的方式进行必要的更改[23]。

虽然面向服务的体系结构不是一个新鲜事物，但它却是更传统的面向对象的模型的替代模型，面向对象的模型是紧耦合的，已经存在二十多年了。虽然基于 SOA 的系统并不排除使用面向对象的设计来构建单个服务，但是其整体设计却是面向服务的。由于它考虑到了系统内的对象，所以虽然 SOA 是基于对象的，但是作为一个整体，它却不是面向对象的。不同之处在于接口本身。SOA 系统原型的一个典型例子是通用对象请求代理体系结构（Common Object Request Broker Architecture，

CORBA），它已经出现很长时间了，其定义的概念与 SOA 相似。然而，现在的 SOA 已经有所不同了，因为它依赖于一些更新的进展，这些进展是以可扩展标记语言（eXtensible Markup Language，XML）为基础的。通过使用基于 XML 的语言（称为 Web 服务描述语言（Web Services Definition Language，WSDL））来描述接口，服务已经转到更动态且更灵活的接口系统中，非以前 CORBA 中的接口描述语言（Interface Definition Language，IDL）可比了。

SOA 本身是应该如何将软件组织在一起的抽象概念。它依赖于用 XML 和 Web 服务实现并以软件的形式存在的更加具体的观念和技术。此外，它还需要安全性、策略管理、可靠消息传递以及会计系统的支持，从而有效地工作。您还可以通过分布式事务处理和分布式软件状态管理来进一步地改善它。

SOA 服务和 Web 服务之间的区别在于设计。SOA 概念并没有确切地定义服务具体如何交互，而仅仅定义了服务如何相互理解以及如何交互。其中的区别也就是定义如何执行流程的战略与如何执行流程的战术之间的区别。而另一方面，Web 服务在需要交互的服务之间如何传递消息有具体的指导原则；从战术上实现 SOA 模型是通过 HTTP 传递的 SOAP 消息中最常见的 SOA 模型。因而，从本质上讲，Web 是实现 SOA 的具体方式之一。

尽管普遍认为 Web 服务是实现 SOA 的最好方式，也是服务(s)的主流，但是 SOA 并不局限于 Web 服务。其他使用 WSDL 直接实现服务接口并且通过 XML 消息进行通信的协议也可以包括在 SOA 之中。正如在别处指出的，CORBA 和 IBM 的 MQ 系统通过使用能够处理 WSDL 的新特征也可以参与到 SOA 中来。如果两个服务需要交换数据，那么它们还会需要使用相同的消息传递协议，但是数据接口允许相同的信息交换。

既为了建立所有这些信息的适当控制，又为了应用安全性、策略、可靠性以及会计方面的要求，在 SOA 体系结构的框架中加入了一个新的软件对象。这个对象就是企业服务总线（Enterprise Service Bus，ESB），它使用许多可能的消息传递协议来负责适当的控制、流甚至还可能是服务之间所有消息的传输。虽然 ESB 并不是绝对必需的，但它却是在

SOA 中正确管理您的业务流程至关重要的组件。ESB 本身可以是单个引擎，甚至还可以是由许多同级和下级 ESB 组成的分布式系统，这些 ESB 一起工作，以保持 SOA 系统的运行。在概念上，它是从早期比如消息队列和分布式事务计算这些计算机科学概念所建立的存储转发机制发展而来的。

2.2 ESB

ESB 作为一个中间件， 支持分布、异构环境中的服务、消息，以及基于事件的交互，并且具有适当的服务级别和可管理性。但是目前对于 ESB 还是没有一个严格的定义或者标准来限定。

被普遍认同的 ESB 定义是：

• ESB 是一种逻辑体系结构组件，它提供与 SOA 的原则保持一致的集成基础架构。

• ESB 可以作为分布式的异构基础架构进行实现。

• ESB 提供了管理服务基础架构的方法和在分布式异构环境中进行操作的功能。

最低的 ESB 功能：

• 通信

提供位置透明性的路由和寻址服务，控制服务寻址和命名的管理功能，至少一种形式的消息传递范型（例如，请求/响应、发布/订阅等等），支持至少一种可以广泛使用的传输协议

• 集成

支持服务提供的多种集成方式，比如 Java 2 连接器、Web 服务、异步通信、适配器等等[9]

• 服务交互

一个开放且与实现无关的服务消息传递与接口模型，它应该将应用程序代码从路由服务和传输协议中分离出来，并允许替代服务的实现。ESB 产品的出现也加速了 EAI（Enterprise Application Integration）向遵

循 SOA 的方向发展。下面的图是 IBM 公司的红宝书《Patterns:Implementing an SOA Using an Enterprise Service Bus》中给出的一个 ESB 的概念图。

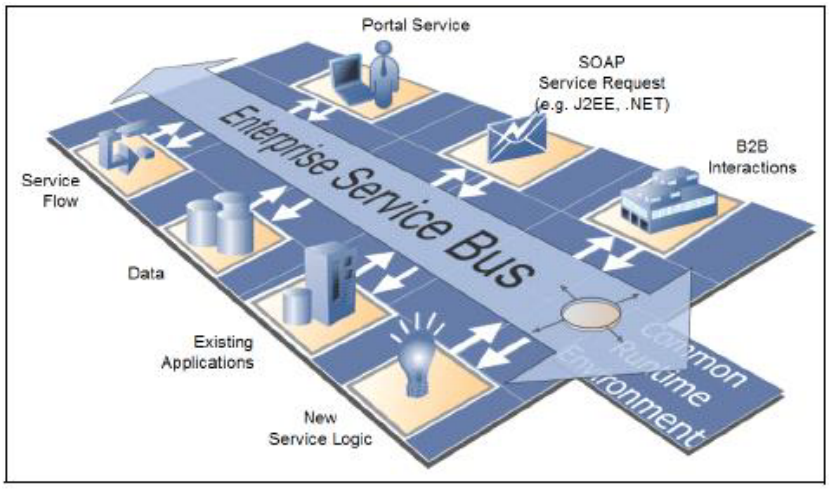

图 1　IBM 企业服务总线[17]

Figure 1　IBM Enterprise Service Bus

ESB 现在成为 SOA 领域中成为一个热点，它在具体实施中处于基础构件的层次，各个被集成的系统之间的交互、通信都要通过 ESB 进行。由于被集成的各个系统与外交互的协议可能不同，而在 ESB 内部传递着的信息是规范的、统一的，所以被集成的系统与 ESB 交互时就存在消息格式、协议转化的任务。但由于各个厂商在开发的 ESB 产品往往提供不了将任何一个系统集成 ESB 的能力，而且现实中做到这一点几乎不可能。但是在 JBI 规范出台之后，ESB 的开发商，适配器、消息转换工具的开发商都可以将使自己产品遵循 JBI 规范，这样使用 ESB 的解决方式将更有生命力，集成系统的规范性、灵活性、松耦合性更强。

2.3 JBI

JBI 规范描述了一个可插拔式的结构，整个结构分为 JBI 环境（JBI Environment）和 JBI 组件（JBI Component），JBI 组件。JBI 环境提供安

装、卸载、管理 JBI 组件的功能，并将由某个 JBI 组件送出的信息传送到目的组件。规范中将 JBI 组件分为两种类型，服务引擎和绑定组件。

服务引擎（Service Engine，SE）：是为其他 JBI 组件提供业务逻辑处理等服务，同时也可能使用其他服务引擎提供的服务的 JBI 组件。

绑定组件（Binding Component，BC）：是在 JBI 环境与外部系统之间起到一个连接器，协议转换作用的 JBI 组件。

服务引擎和绑定组件是在整个结构中担当的角色不同，在规范定义的 API 中并没有明确的两个类或两个接口来区分。相反，不管是服务引擎还是绑定组件都要实现 component 这个接口。

下面是规范中 JBI 的抽象图

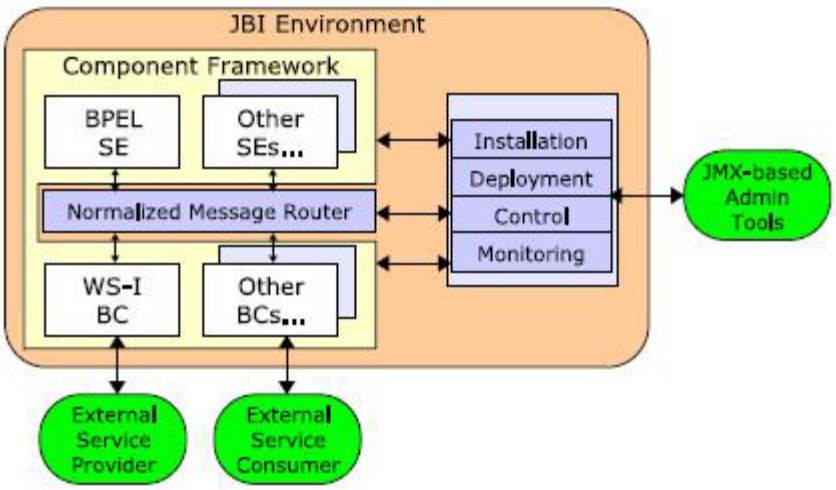

图 2　JBI 的架构图 [1]

Figure 2　The Architecture of JBI

外部系统通过绑定组件（BC）与 JBI 环境交互，而 JBI 环境内部除了包含绑定组件和服务引擎意外，还包含了五个模块：

规范消息路由器（Normalized Message Router）：所有规范消息（Normalized Message）都要通过规范消息路由器在 JBI 组件之间传送。

安装器（Installation）：负责安装 JBI 组件

部署器（Deployment）：负责部署服务集合（service assembly）

控制器（Control）：可以通过控制器对 JBI 组件进行启动，停止，重启动等操作

监听器（Monitoring）：监听 JBI 的信息与状态

基于 WSDL（Web Service Description Language）的消息模型

JBI 组件提供或使用服务的模型采用 WSDL1.1 和 WSDL2.0 标准，在术语上使用上保持与 WSDL2.0 标准一致。WSDL 中分为抽象描述和具体定义服务两部分。抽象描述部分只定义消息的数据类型和抽象服务包含的操作和使用的消息，不与具体的编码方式和通信协议相关。在具体定义服务部分要将抽象描述的服务与具体的通信协议和编码方式绑定。

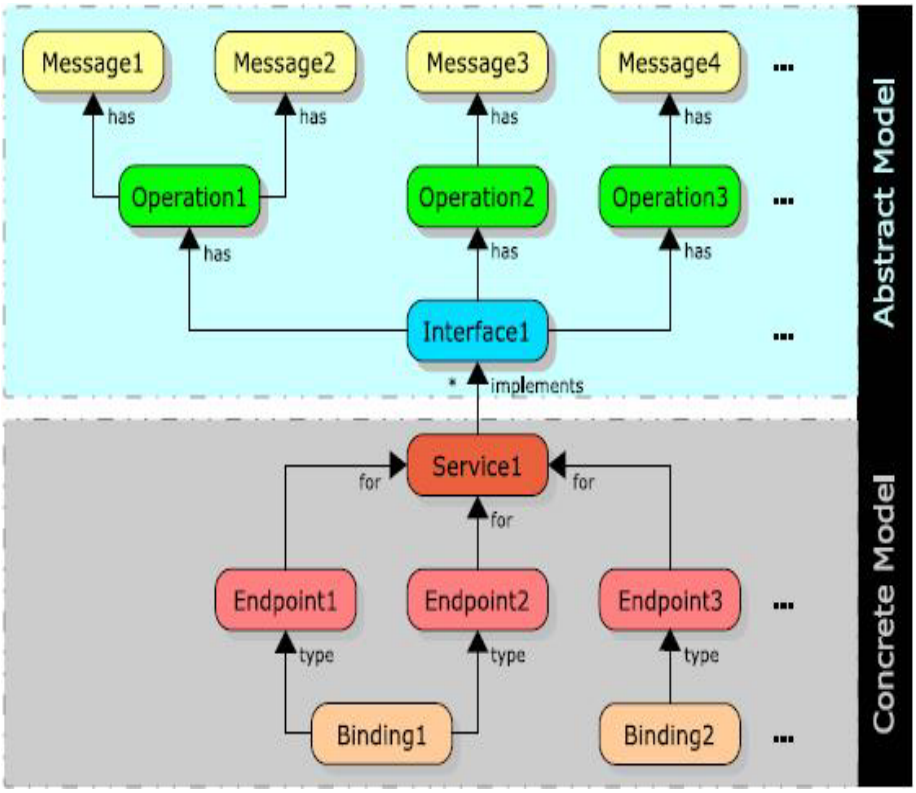

图 3　WSDL2.0 中抽象与实现模型[1]

Figure 3　The Abstract and Concrete Model of WSDL2.0

JBI 组件之间的交互使用 WSDL 的具体定义部分来描述。组件与组件之间交互的规范消息（Normalized Message）由两部分组成：消息的上下文数据（context data）和描述组件提供服务的信息。下图是 JBI 组件如何进行交互的示意图

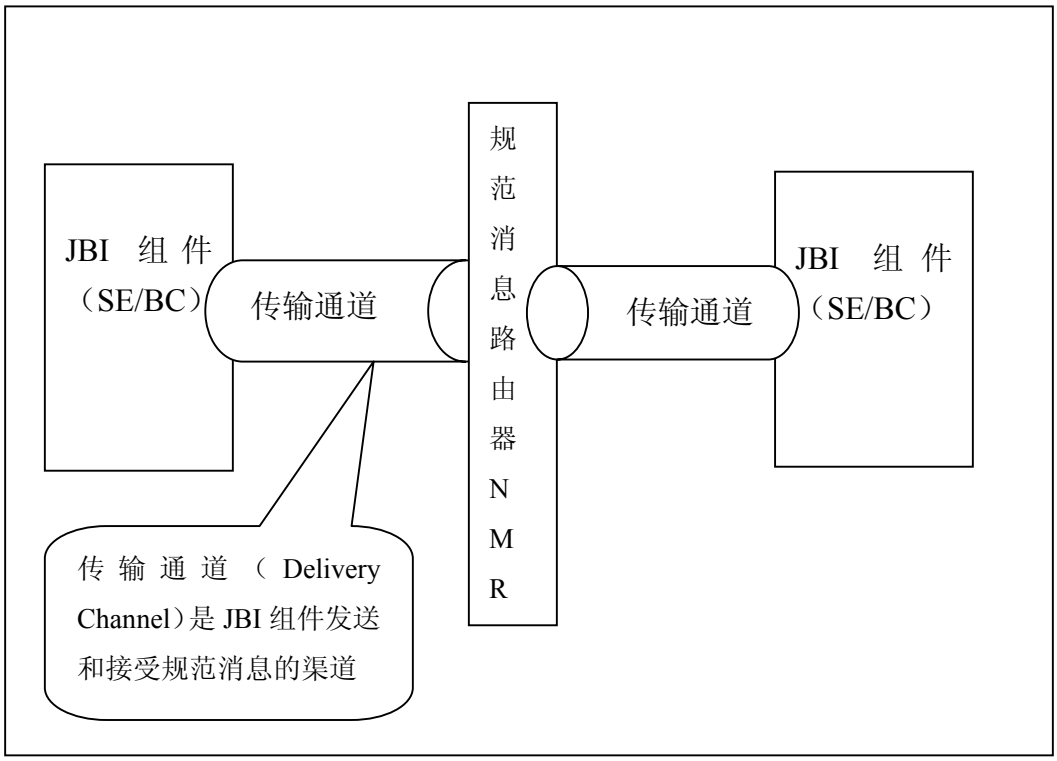

图 4　JBI 组件之间的通信

Figure 4　The Communication between JBI components

而在传输通道并不是直接传送的规范消息，而是规范消息交换包（Normalized Message Exchange），规范消息交换包除了封装规范消息以外，还包括规范消息的发送者信息和目标服务信息等其他信息。规范消息交换包（Normalized Message Exchange）有四种类型，分别代表了四种交互方式，下表是交互方式与规范消息交换包类型的对应表：

表 1 规范信息交换的类型

Table 1 The Types of Normalized Message Exchange

交互方式	规范消息交换包类型
单向类型（One－Way）	In－Only
可靠单向类型（Reliable One－Way）	Robust In－Only
请求响应类型（Request－Response）	In－Out
请求可选响应类型（Request Optional－Response）	In Optional－Out

当一个 JBI 组件安装到 JBI 环境中时，就会有由它的上下文负责生成一个与之唯一对应的传送通道（Delivery Channel）。当 JBI 组件要发送一个规范消息的时候，这个 JBI 组件首先要获得与它对应的传送通道实例，并由传送通道生成一个规范消息交换包的实例，由该规范消息交换包的实例再生成一个规范消息实例，在这个 JBI 组件将生成的规范消息赋值之后，将该规范消息加载到规范消息交换包中，然后通过传送通道将规范消息交换包发送给规范消息路由器（NMR）。JBI 组件接受规范消息时，是当目标服务由该 JBI 组件提供的时候，规范消息路由器将规范消息交换包通过传送通道发送给该 JBI 组件，这个 JBI 组件要先从规范消息交换包中取出规范消息，然后在对规范消息进行后续的处理。

在 JBI 规范简单的介绍之后，本文通过一个具体项目案例来介绍如果利用遵循 JBI 规范的开源 ESB 产品 ServiceMix 来进行系统集成的应用。

第3章 开源 ESB—ServiceMix

3.1 简介

ServiceMix 是一个实现了 JBI（JSR208）规范的，使用 Apache 许可证的一个开源 ESB 和 SOA 工具包。它是一个轻量级和可嵌入式的产品，综合了 Spring 的支持，可以运行在客户端或服务器端，可以作为一个独立的 ESB 提供基础服务的支持，也可以作为一个服务引擎组件集成到其他的 ESB 产品中。

同时 ServiceMix 可以被应用在 J2SE 或 J2EE 的应用服务器中，现在 ServiceMix 被 Geronimo（Apache 下开源 J2EE 应用服务器，已通过 J2EE1.4 认证）提交作为 Apache 的一个孵化器项目。ServiceMix 将与 Geronimo 可以完全集成，支持用户将 JBI 组件直接部署到 Geronimo 中。

目前 ServiceMix 除了实现 JBI 规范定义的 ESB 核心部分（规范消息路由器、管理组件、规范消息模型、JBI 组件框架）之外，还提供了一些基于第三方工具的 JBI 组件和一个开发轻量级 JBI 组件的框架。基于此框架，开发一个 JBI 组件将比 SUN 公司发布的，一个开发 JBI 组件的例子所使用的方法简便的多。

下面是 ServiceMix 的概念图

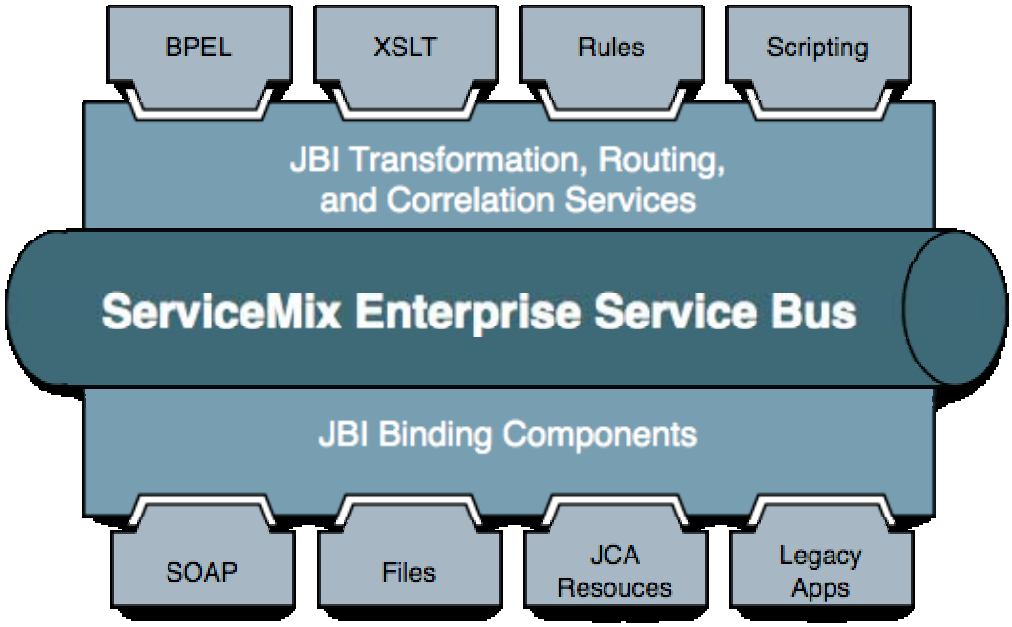

图 5 开源 ESB－ServiceMix[3]

Figure 5 Open Source ESB－ServiceMix

通过在 ServiceMix 上部署不同的绑定组件，可以支持具有不同接口和使用不同通信协议（JMS，SOAP 等）的系统与 ESB 进行通信。

在 ServiceMix 中通过 JBIContainer 实现了 JBI 规范中的 JBIEnvironment，JBIContainer 中包含了部署（Deployment）、安装（Installation）、注册（Registry）的服务，和规范消息路由器（Broker）。他们都是可由 JMX 工具管理的 MBean。当 ServiceMix 启动的时候，首先是根据配置文件去启动 JBIContainer 并配置它的属性，然后由 JBIContainer 启动其所包含的服务。JBIContainer 的注册服务中包含了对 JBI 组件和服务端点（endpoint）注册。一个 JBI 组件可以有多个服务端点，但至少会有一个服务端点，默认的服务端点为与 JBI 组件服务名相同。

3.2 ServiceMix 中轻量级组件与标准组件

在 ServiceMix 上运行的 JBI 组件可以分为轻量级组件和标准组件，它们由不同的方式开发，通过不同的方式部署。

轻量级组件（POJO）：利用 ServiceMix 的提供的开发轻量级 JBI 组件的框架，可以开发出单类的组件，容易被没有接触过 JBI 规范的开发者掌握。它通过配置文件部署到 ServiceMix 上。开发一个轻量级的组件只需要继承 ServiceMix 提供的支持类 ComponentSupport，下图是 ComponentSupport 类继承其它支持类和实现 JBI 规范中接口的类图。

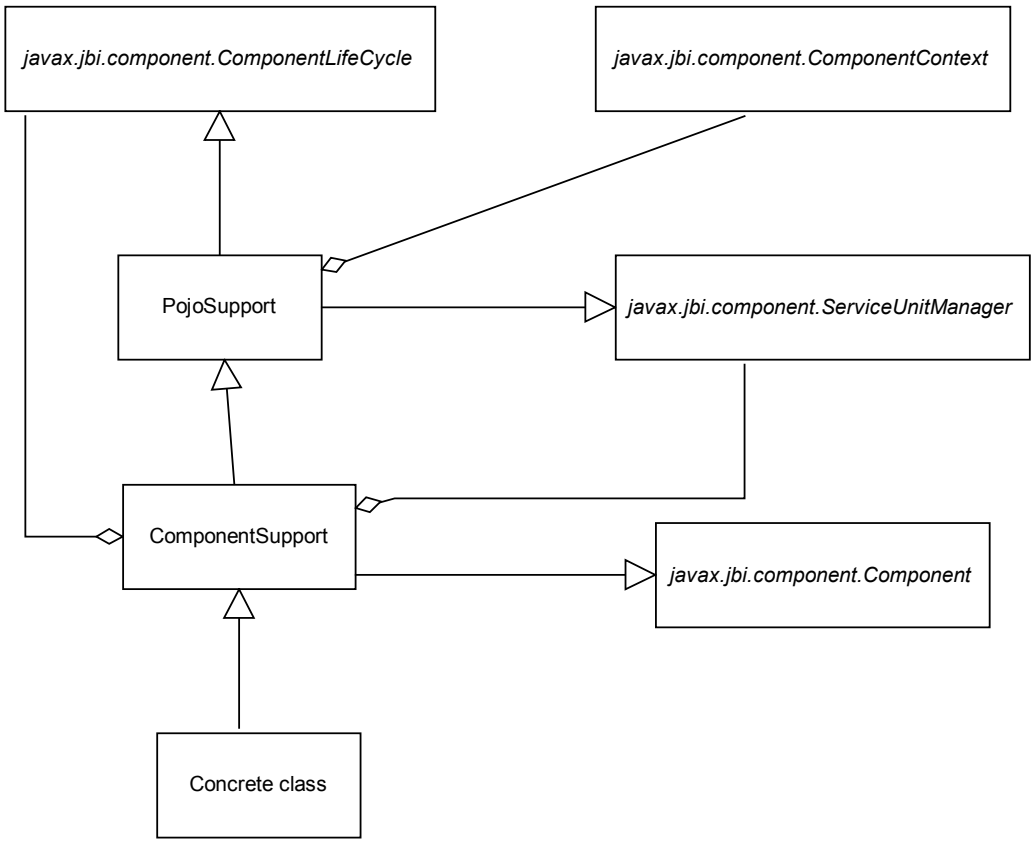

图 6 ComponentSupport 及其祖先类的类图

Figure 6　The class diagram of ComponentSupport and its ancestor

标注组件（jar 包）：开发标准的 JBI 组件相对开发轻量级组件要复杂，同时要求对 JBI 规范熟悉。一个标准的 JBI 组件以 jar 包的形式通过 ServiceMix 的自动安装机制进行部署（将 jar 包 copy 到 ServiceMix 的 install 文件夹下）。标准组件一般都包含多个类，每个类要实现 JBI 规范中的一个或多个接口。

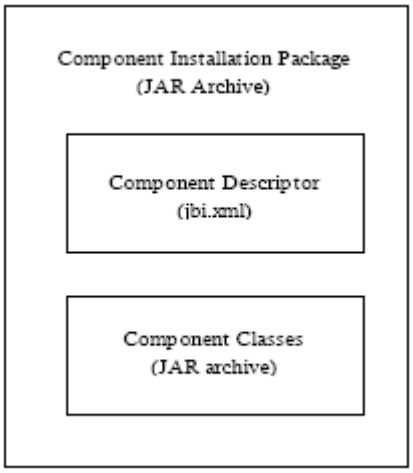

图 7 标准 JBI 组件安装包的组成[31]

Figure 7 The Composing of Standard JBI Component Installation Package

在 SUN 公司开发标准 JBI 组件的一个例子中，实现的类图为

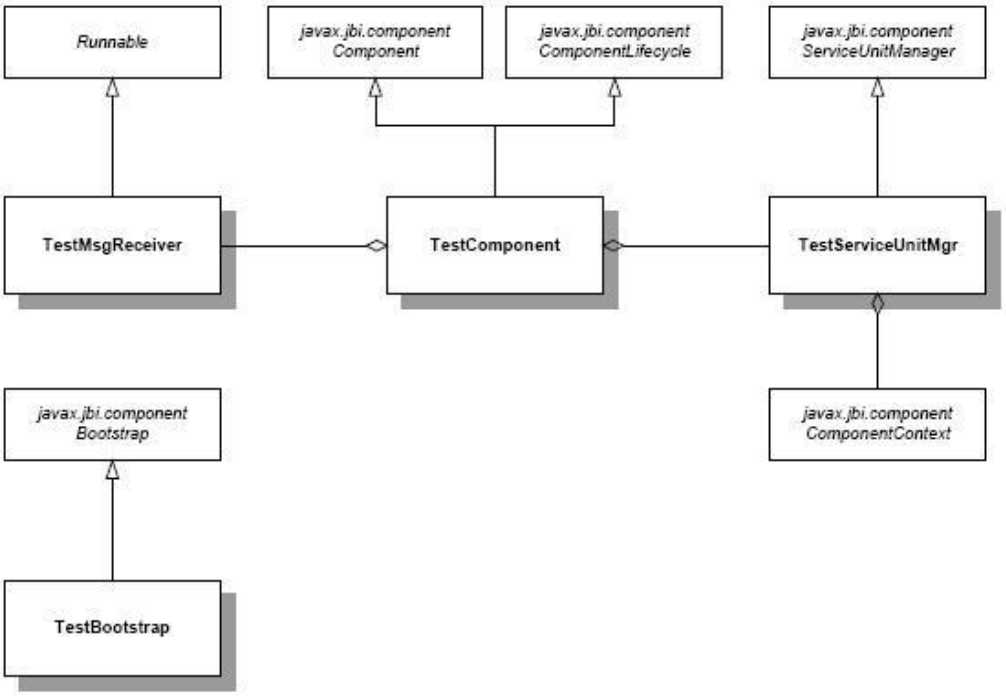

图 8 标准 JBI 组件的类图[31]

Figure 8 The class diagram of standard JBI component

轻量级组件开发简单，易于掌握，但依赖与 ServiceMix，可移植性差；标准组件开发复杂，但可移植性强，除了可以部署在 ServiceMix 中，还可以部署在任何符合 JBI 规范的 ESB 产品中。

第4章 基于ServiceMix的系统集成应用

4.1 项目背景

日本某上市软件公司决定在 SOA 领域加大投入力度，对 ESB 产品尤其是开源产品进行调查，并选定一个相对成熟的开源产品，在其基础上试验进行系统集成，以积累公司在这方面的力量。被集成的目标是两个在企业内部日常使用的系统，一个是所有员工使用的“个人事务管理系统”，员工在该系统上安排个人的工作安排、进行考勤和记录每日的工作内容和成果，该系统的记录是公司对员工进行考核的一个依据。另一个系统是由公司管理层成员、项目组长和办公室人员使用的“会议室预定系统”，方便安排日常会议和临时会议。但经过一段时间的使用，发现管理人员和项目组长有时需要临时安排一些会议，在安排会议时，他们要首先到个人事务管理系统查看自己已有的安排避免冲突，同时又要到会议室预定系统查找在自己的空闲时间内是否还有空闲的会议室，由于会议室比较多，同时还要在两个系统之间查询，比较麻烦。公司希望能构造一个新的系统，使管理人员和项目组长可以同时查到自己的空闲时间段和各会议室安排的情况，并一目了然地看到自己应该哪段时间、哪个会议室安排会议。同时新系统要能够提供查询股票价格的功能，以方便管理人员及时了解本公司股票和其他公司股票的价格。

由于个人事务管理系统和会议室预定系统不是由同一各开发团队开发，开发使用的语言因而不同。个人事务管理系统是用 php 开发的，而会议室预定系统是由 asp 开发。同时公司要求开发新系统不能影响目前两个系统的独立性和正常运行。因此从一开始，项目组就决定采用 ESB 方式进行集成，以保证系统的灵活性和可扩展性。

经过两个月的对各个 ESB 产品调查，开源 ESB 产品 ServiceMix 在成熟度、可靠性等方面都比其他 ESB 产品表现出更适合公司使用的特点。最终项目组决定在 ServiceMix 基础上构造新系统，对上述两个系统进行集成。

由于对两个已有系统不能进行任何变更，在一段调查和试验之后，项目组决定开发基于 HttpUnit（对 Web 系统进行黑盒测试的工具）的绑定组件（BC），作为与已有两个系统进行交互的代理。

对于要提供一个查询股票价格的功能，由于已经存在第三方开发的根据上市公司的股票代码查询股票价格的 web 服务，同时为了保证新系统的灵活性，因此项目组决定开发一个基于 Axis（Apache 组织的 web 服务引擎及开发和调用工具）的 web 服务客户端或代理。

4.2 需求分析

新系统的功能性需求：

- 访问个人事务管理系统并获取用户在某时间段内的事务安排信息
- 访问会议室预定系统并获取某时间段内会议室的安排信息
- 访问远程的查询股票价格的 web 服务并获得某公司的股票价格

非功能性需求：

- 将查询到的个人事务信息和会议室安排信息在同一个表单内显示
- 新系统仍为 B/S 结构的 web 应用程序

用例图：

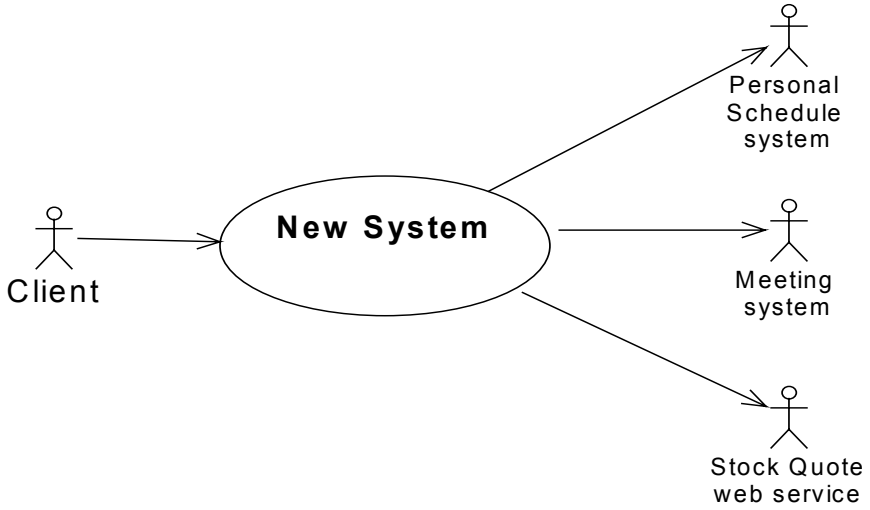

图 9 用例图 1

Figure 9 Use Case diagram 1

对用例 New System 进一步分解，系统应该包括登录，查询某时间段内个人的事务安排和各会议室的安排情况，查询某公司股票价格三个用例，分解之后的用例图为

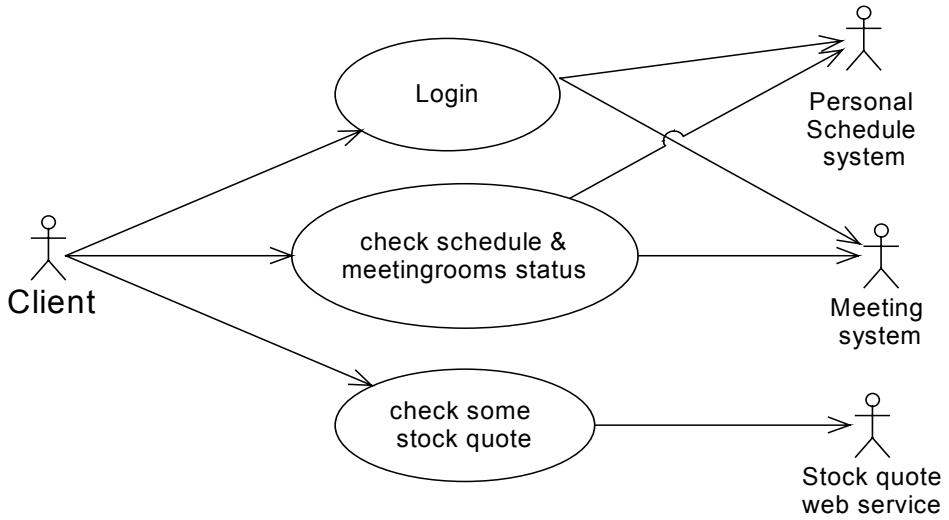

图 10 用例图 2

Figure 10 Use Case diagram 2

用例文档：
UC1：登录
用例描述：用户登录系统
参与者：用户
前置条件：用户访问系统
后置条件：用户通过验证
基本路径：
1. 用户输入帐号和密码并提交
2. 系统验证帐号和密码
3. 系统显示登录后的主界面
扩展点：
2a. 用户提交的帐号或密码错误
2a1. 系统显示登录失败和错误信息
补充说明：
只有当用户同时具有个人事务管理系统和会议室预定系统的使用权限时才能登录成功。
UC2：查询个人事务和会议室安排信息
用例描述：用户根据起止日期来查询所选择的会议室的安排状况以及个人事务的信息
参与者：用户
前置条件：用户登录系统
后置条件：用户获得个人事务和会议室安排状况信息
基本路径：
1. 用户选择起止日期和会议室（可多选）并提交
2. 系统检查起止日期内的所选会议室状况和该用户的个人事务信息
3. 系统显示检查结果
UC3：查询股票价格
用例描述：用户根据股票代码查询股票价格
参与者：用户
前置条件：用户登录系统

后置条件：用户获得某股票价格

基本路径：

1. 用户输入股票代码并提交
2. 系统根据股票代码查询股票价格
3. 系统显示查询结果

补充说明：

若用户提交的股票代码不正确或已被摘牌则返回结果为－1。

特别要求：

新系统查询出某时间段的个人事务和各会议室安排状况的结果要在同一个表单内显示。

个人事务管理系统中显示某人在某时间段内（某几天内）每日的事务安排的表单样式为

表 2 个人事务安排时间表

Table 2 The table of personal schedule

Time / Date	8		9		10		11		……		18		19		20	
某月 1 日	…															
……				…												
某月 5 日											…					

会议室预定系统中显示某时间段内（某几天内）每日各个会议室的安排状况的表单样式为

表 3 会议室时间安排表

Table 3 The table of Meetingrooms status

Time / Meetingroom	8		9		10		11		……		18		19		20	
会议室甲	…															
……				…												
会议室辛											…					

客户要求新系统同时显示某几天内某人的个人事务和各会议室状况的

表单样式为

表 4　个人事务信息与会议室信息的综合表

Table 4 The integrated table of personal schedule and meeting rooms status

某月 1 日	…																
Time / Meeting room	8		9		10		11		……		18		19		20		
会议室甲	…																
……				…													
会议室辛											…						
……																	
某月 5 日												…					
Time / Meeting room	8		9		10		11		……		18		19		20		
会议室甲	…																
……				…													
会议室辛											…						

4.3 详细设计

经过一段时间的分析与调研，整个系统的结构渐渐清晰，其系统结构图和部署图如下：

系统结构图

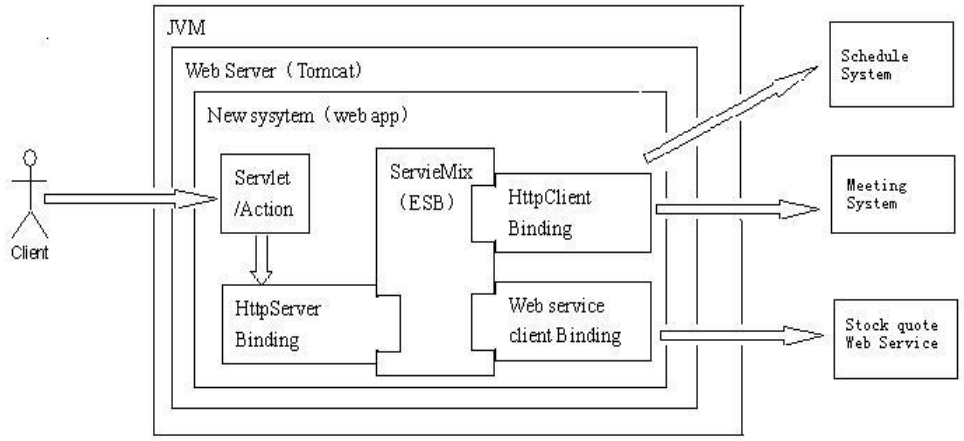

图 11　应用程序的体系结构

Figure 11　The architecture of application

与以往的 web 程序的不同之处在于，在引入了 ESB 的概念和结构。因此新系统的开发除了使用通常开发 web 程序的技术外，还涉及到开源 ESB 产品 ServiceMix、Web service 工具 Axis 和对 web 系统进行黑盒测试的工具 HttpUnit 的应用。

ServiceMix 是 LogicBlaze（JMS 工具 ActiveMQ 的开发者）、GlueCodee（Apache Geronimo 应用服务器的贡献者，现被 IBM 公司收购）、MortBay（web 服务器 Jetty 的开发者）等公司联合开发的开源 ESB 产品。现在 ServiceMix 的大部分代码作为基础，由 Apache Geronimo 项目组提议进去 Apache 孵化项目。

ServiceMix 是遵循 JBI 规范的，它可以作为一个 ESB 在 java 虚拟机上独立运行，其他的应用程序都可以通过绑定组件集成进来；也可以嵌入到 web 服务器、应用服务器或嵌入到应用程序中。ServiceMix 除了实现了 JBI 规范以外，还提供了开发轻量级 JBI 组件的框架，同时提供一些可供选用的 JBI 组件。它是基于 Spring 开发而成，因而具有控制反转（Ioc）的特点，所以 JBI 组件的开发人员只需要将精力放在该组件提供什么功能以及如何实现上。

开发工作的重点是在开发三个 JBI 绑定组件（BC）上，HttpServerBinding、

HttpClientBinding、WebServiceClientBinding。

HttpServerBinding：

它起着 web 程序（服务器端）与 ServiceMix（ESB）的连接器的作用。例如当用户登录时，提交的用户帐号和密码信息都保存在请求对象中，HttpServerBinding 要将请求对象中的信息保存到一个规范消息（Normalized Message）对象中，并将规范消息对象发送给 ServiceMix 的规范消息路由器（Normalized Message Router），然后由规范消息路由器发送到目标组件——HttpClientBinding，由 HttpClientBinding 访问个人事务管理系统或会议室预定系统验证用户的正确性并返回验证结果。

HttpClientBinding：

它是 ServiceMix 与外部的 web 系统进行交互的桥梁。它是利用了 HttpUnit 对 web 系统可以进行黑盒测试的功能，因此可以在不管目标系统的开发语言是什么，运行平台是什么，同时也不对目标系统进行任何修改，不影响目标系统的独立性的情况下，以从目标系统界面获取信息的方式利用目标系统的信息资源。

WebServiceClientBinding：

它是 ServiceMix 调用 web 服务的代理，可以配置目标服务地址，目标服务名称，及调用的操作名称调用 web 服务。

这里还有一个问题，在 ServiceMix 中一个 JBI 组件发送给规范消息路由器的规范消息如何被传送到正确的目标组件呢？ServiceMix 需要根据一个配置文件来配置 ESB 本身以及 JBI 组件。对 JBI 组件可以配置其服务名称，及这个组件对应的具体的类和属性的初始值，以及该组件发送规范消息的目标地址。也可以在程序中指定规范消息被传送的目标地址。一般情况使用配置文件的方式将更具有灵活性，这样当组件与组件之间的流程逻辑发生变更的时候，不需要更改程序，而只需要更改配置文件即可。对于轻量级的应用，也可以使用代码控制的路由方式。还可以利用 BPEL 引擎执行 BPEL 文件来编排服务调用流程,或者利用一些

基于规则的路由服务引擎组件来控制规范信息流。

下面是 ServiceMix 中一个自带的一个例子中，配置文件的部分内容

```
<sm:activationSpec componentName="fileSender"      service="foo:fileSender">
<sm:component>
<bean xmlns="http://xbean.org/schemas/spring/1.0"
                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>
           </bean>
      </sm:component>
</sm:activationSpec>
<!-- Look for files in the inbox directory -->
<sm:activationSpec componentName="filePoller"
        destinationService="foo:fileSender" service="foo:filePoller">
     <sm:component>
       <bean xmlns="http://xbean.org/schemas/spring/1.0"
               class="org.servicemix.components.file.FilePoller">
           <property name="workManager" ref="workManager" />
           <property name="file" value="inbox" />
           <property name="period" value="1000" />
      </bean>
```

```
    </sm:component>
</sm:activationSpec>
```

该配置文件（XML 文件）的 XML Schema 采用了 ServiceMix 开发人员对 spring 开发人员定义的 xml schema 的扩展。关于 Spring 及其定义的 XML Schema 如何使用等，可以到 Spring 的官方网站（www.springframework.org）上去查这方面的资料。在上面的 xml 文件片断中，定义了两个 JBI 组件 fileSender 和 filePoller。这两个组件提供的服务名称与这两个组件各自的名称相同。在组件 filePoller 的定义中指定了目标服务的名称，即 destinationService="foo:fileSender"，所以 filePoller 组件发送的规范消息都将由规范消息路由器转发给目标服务 fileSender,当如果同时有多个组件提供 fileSender 服务时，规范消息路由器将根据一定规则（目前 ServiceMix 采用根据服务名称获得的组件队列中选择第一个组件）来选择一个确定的 JBI 组件来提供相应服务。

这两个组件，filePoller 提供的服务是在每隔固定时间间隔（1000 毫秒）在当前目录的 inbox 文件夹中读取所有 xml 文件，并将文件装载入规范消息然后传送给规范消息路由器。fileSender 提供服务是在接受到规范消息之后，读取规范消息承载的文件并将文件写到当前目录下的 outbox 文件夹内。按照 JBI 规范的定义，这两个 JBI 组件为绑定组件。

如何开发 JBI 组件呢？在 SUN 公司的网站上有个开发向导（PDF 文件，java.sun.com/integration/reference/techart/DevelopingAServiceEngineComponent-1-2.pdf）和一些资料（http://java.sun.com/integration/pa1/docs/），但是这种开发方式相对复杂。利用 ServiceMix 的 JBI 组件开发框架，可以快速的开发一个轻量级的 JBI 组件。利用 ServiceMix 开发 JBI 组件需要了解其提供的一个关键类 ComponentSupport 和一个接口 MessageExchangeListener。如果开发人员要开发一个服务的“消费者”，只需要继承 ComponentSupport；如果要开发一个服务的“提供者”或者它既是“消费者”又是“提供者”，则需要继承 ComponentSupport 并实现接口 MessageExchangeListener。

ComponentSupport 类提供了对 Normalized Message Exchange 和 Normalized Message 等 JBI 规范中定义的功能都提供了完整的支持。实

现了接口 MessageExchangeListener，则可以为开发的 JBI 组件提供基于事件驱动和基于消息驱动来提供服务的功能。

◆ 开发 JBI 组件

客户通过新系统访问目标 web 系统的事件流为

1．客户发出请求
2．Action 接受到请求后，实例化一个帮助类 Hyp_SM_Util，并通过该帮助类获得 HttpServerBinding
3．HttpServerBinding 将 http 请求对象转化为规范消息并发送
4．规范消息路由器将规范消息发送给 HttpClientBinding
5．HttpClientBinding 访问目标 web 系统，并将结果转化为规范消息发送
6．规范消息路由器将规范消息发送给 HttpServerBinding
7．HttpServerBinding 返回结果至 Action
8．Action 显示结果

其时序图为

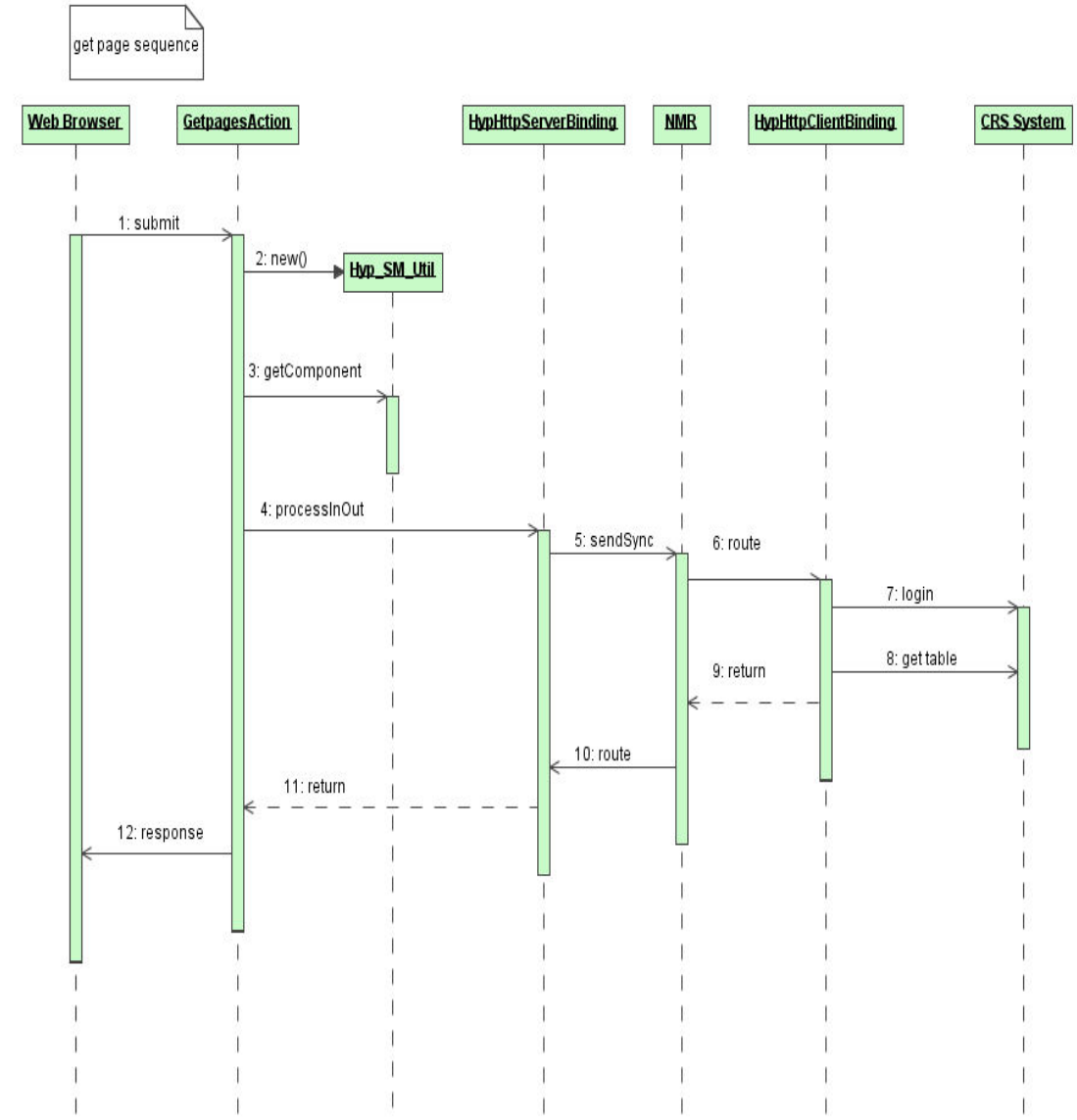

图 12 客户端访问目标系统的时序图

Figure 12 The sequence diagram of accessing the target web system

组件 HttpServerBinding（类名 HypHttpServerBinding）的类图为：

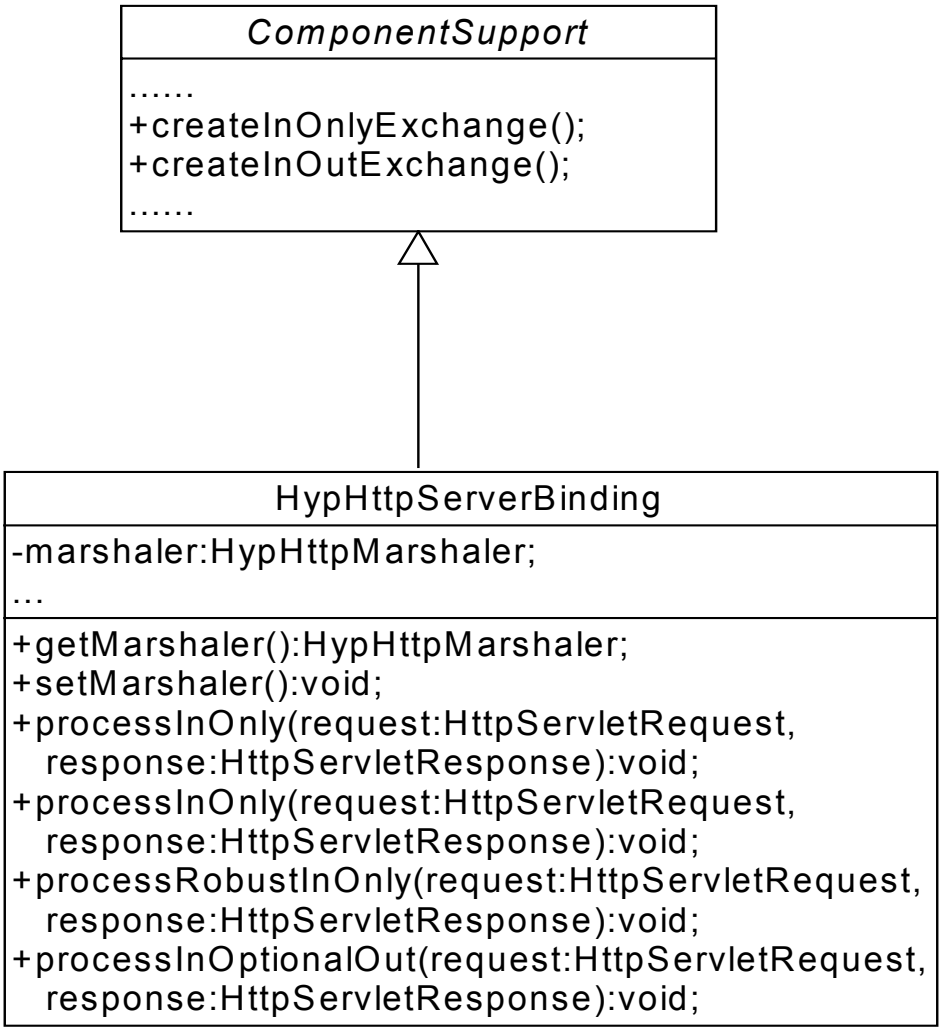

图 13　HttpServerBinding 的类图

Figure 13　The class diagram of HttpServerBinding

该组件的功能是将 http 的请求对象转换为规范消息对象，并将规范消息对象转换为 http 的响应对象。根据 JBI 规范对规范消息交换的不同类型的定义，该组件具有四种处理 http 请求和响应对象的方式。Action 可以根据不同的需要调用不同的方法。

组件 HttpClientBinding（类名 NewHttpClientBinding）的类图为：

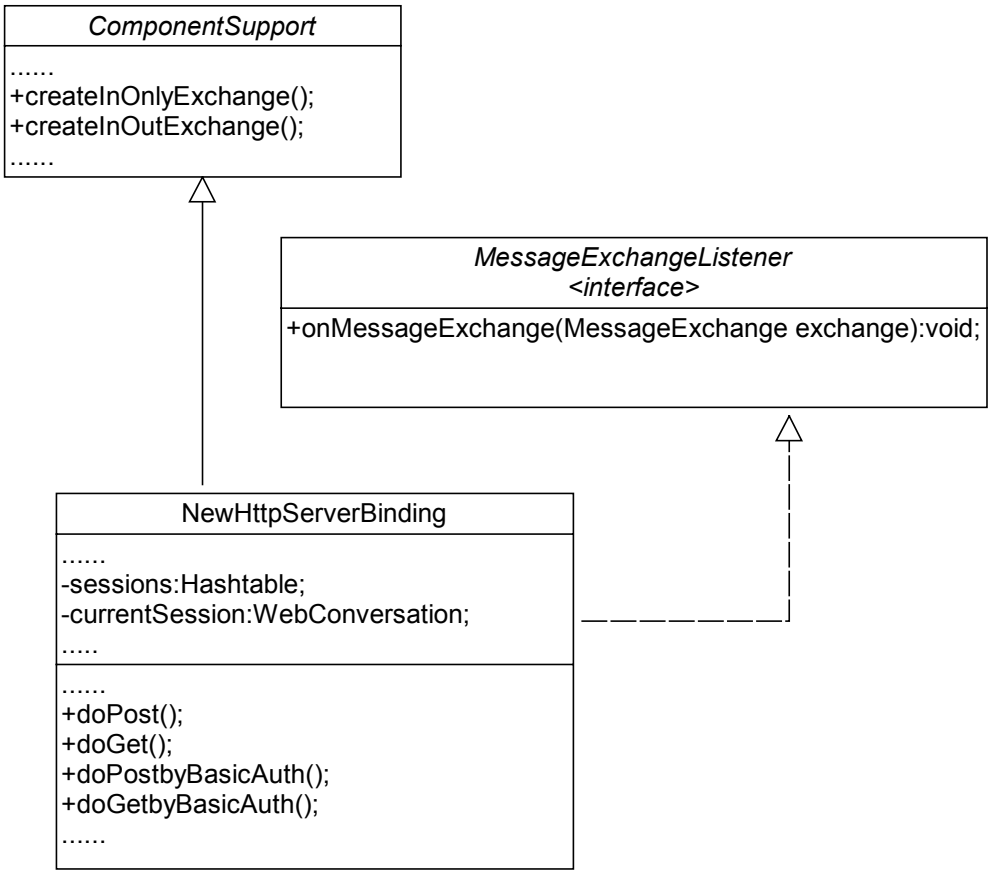

图 14　HttpClientBinding 的类图

Figure 14　The class diagram of HttpClientBinding

该组件的功能主要是与远程的 web 系统进行交互，并将获得的信息转换为规范消息。在这个组件要负责保持登录到新系统的用户与登录远程系统用户的一直性，同时还要保持会话的连续性。通过将登录到新系统的用户的会话（Session）ID 和用于保持和目标系统会话保持连续性的对象 WebConversation（HttpUnit 中的类）保存到一个哈希表中。当 HttpClientBinding 组件接受到一个请求（有规范消息路由到该组件），首先检查发出这个请求的用户的会话 ID 是否存在哈希表中，如果存在就读取对应的 WebConversation 对象，如果不存在，就新生成一个

WebConversation 对象，并将该用户的会话 ID 和 WebConversation 对象存储到哈希表中。

客户通过新系统访问远程 web 服务的事件流为

1．客户发出请求
2．Action 接受到请求后，实例化一个帮助类 Hyp_SM_Util，并通过该帮助类获得 HttpServerBinding
3．HttpServerBinding 将 http 请求对象转化为规范消息并发送
4．规范消息路由器将规范消息发送给 SAAJBinding
5．SAAJBinding 访问目标 web 服务，并将结果转化为规范消息发送
6．规范消息路由器将规范消息发送给 HttpServerBinding
7．HttpServerBinding 返回结果至 Action
8．Action 显示结果

其时序图为

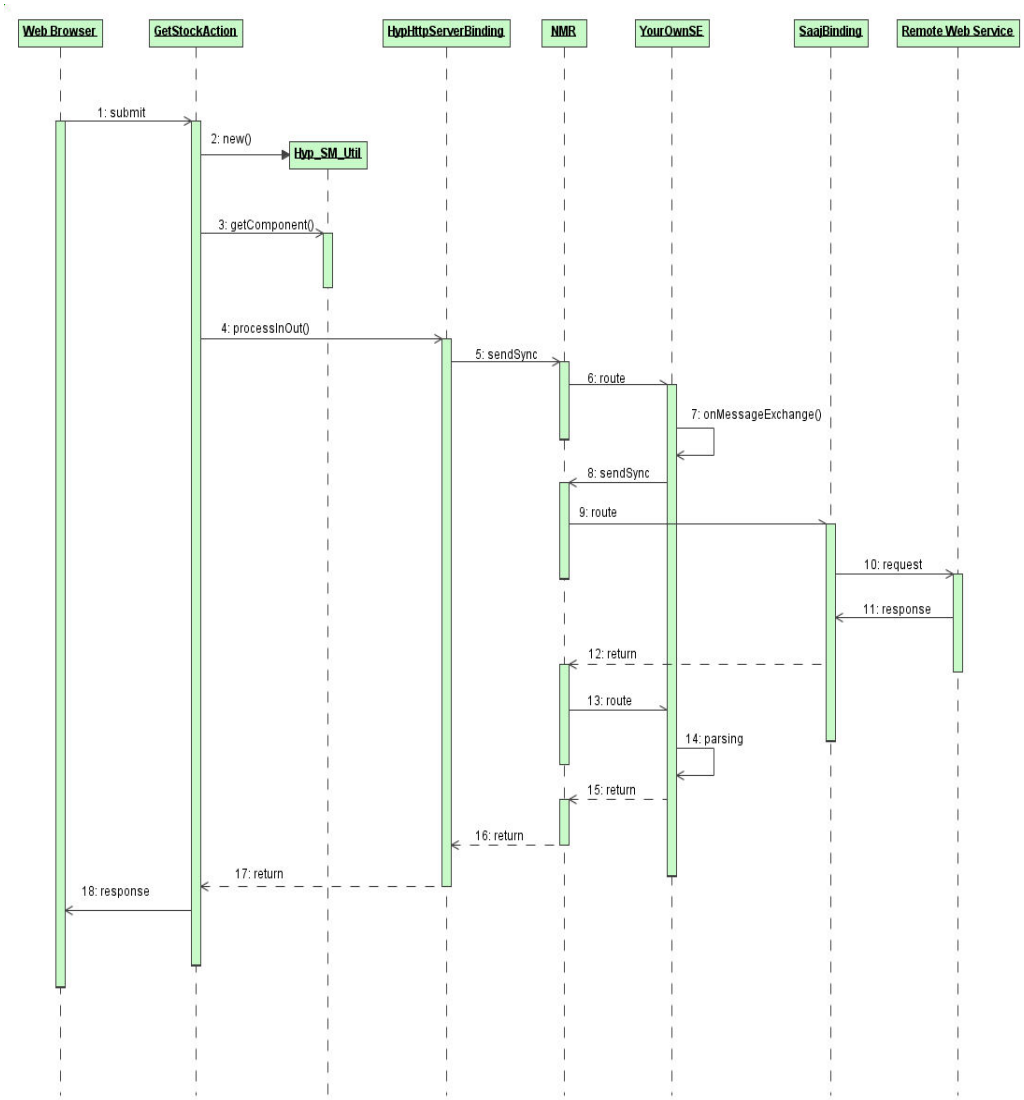

图 15 访问 web 服务的时序图

Figure 15 The sequence diagram of accessing web service

组件 WebServiceClientBinding（类名 SaajBinding）的类图为：

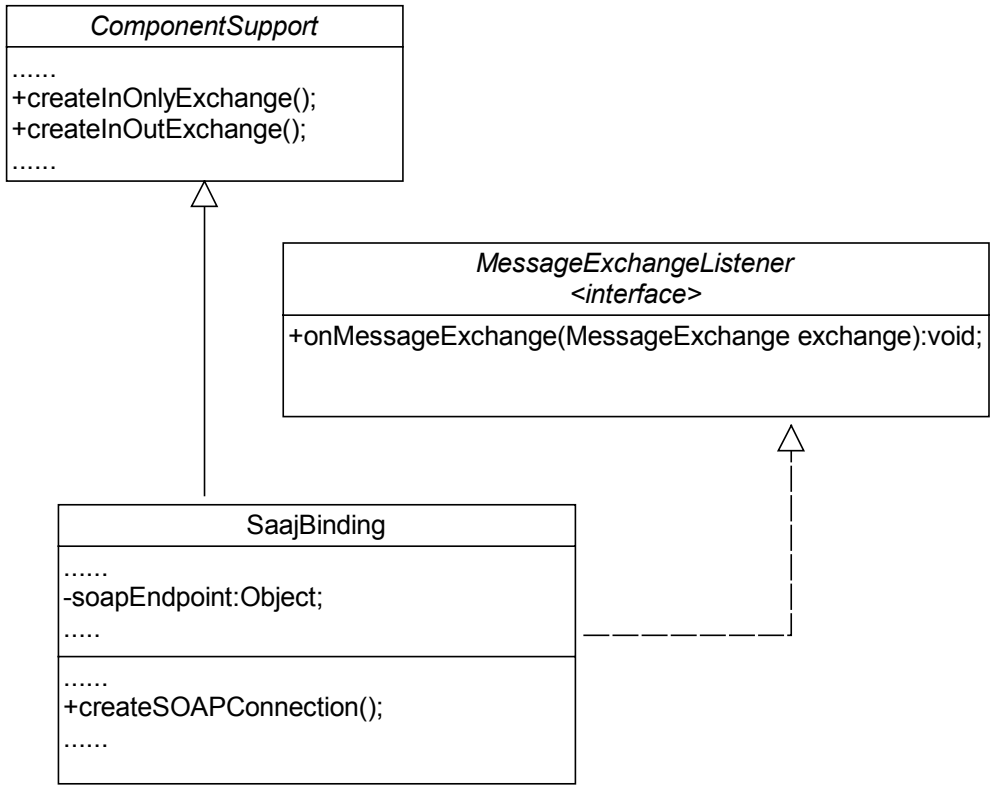

图 16 WebServiceClientBinding 的类图

Figure 16 The class diagram of WebServiceClientBinding

该组件的功能是访问远程的 web 服务并返回结果。

其实现可基于标准的 SAAJ 的 API，也可以利用其他的 web 服务客户端工具，如 Axis、ActiveSoap、xFire 等。在本案例中使用 ServiceMix 提供的基于 SAAJ 的轻量级绑定组件－SAAJBinding。

◆ 开发 web 组件

在实现了这三个 JBI 组件之后，所面临的工作主要是进行系统的前台的开发，这里有登录系统、查询个人与会议室安排状况、重新查询个人与会议室的安排状况、查询股票价格四个动作，因此在基于 struts 开发时需要有四个 action。同时为了应对可能发生的目标系统的迁移，和新系统可以灵活的访问可能需要进行集成的目标系统，通过配置文件

（xml 文件）的方式来指定目标系统的地址。

下面是这个配置文件的例子：

```
<?xml version="1.0" encoding="UTF-8"?>
<hypsft:Targets xmlns:hypsft="http://www.hypersoft.com.cn/TargetHostConfig"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.hypersoft.com.cn/TargetHostConfig
    TargetHostConfig.xsd ">
<hypsft:Host>
        <hypsft:HostName>Schedule</hypsft:HostName>
        <hypsft:URL>http://192.168.1.87</hypsft:URL>
</hypsft:Host>
<hypsft:Host>
        <hypsft:HostName>Meeting</hypsft:HostName>
        <hypsft:URL>http://192.168.1.87</hypsft:URL>
    </hypsft:Host>
</hypsft:Targets>
```

当目标系统发生迁移，或者需要将另一个新的 web 系统集成进来时只需要更改该配置文件。

当用户进行登录时，LoginAction 要对该配置文件进行解析，解析部分的实现为的实现为

```
String rootpath=request.getRealPath("")+"/WEB-INF/";
String filename="TargetHostConfig.xml";
FileReader freader=new FileReader(rootpath+filename);
SAXReader reader=new SAXReader();
Document doc=null;
doc = reader.read(freader);
Element root=doc.getRootElement();
Iterator HostList=root.elements("Host").iterator();
Element  ScheduleHost=(Element) HostList.next();
String Schedule=ScheduleHost.element("HostName").getText();
```

```
String scheduleURL=ScheduleHost.element("URL").getText();
Element   MeetingHost=(Element) HostList.next();
String Meeting=MeetingHost.element("HostName").getText();
String meetingURL=MeetingHost.element("URL").getText();
```

在查询动作 CheckMeetScheduleAction 中，需要将查询到的个人事务安排的表单和会议室安排的表单合成为一个表单。

个人事务安排的表单截图为

图 17 个人事务表单

Figure 17 The table of personal schedule

会议室安排状况的表单截图为

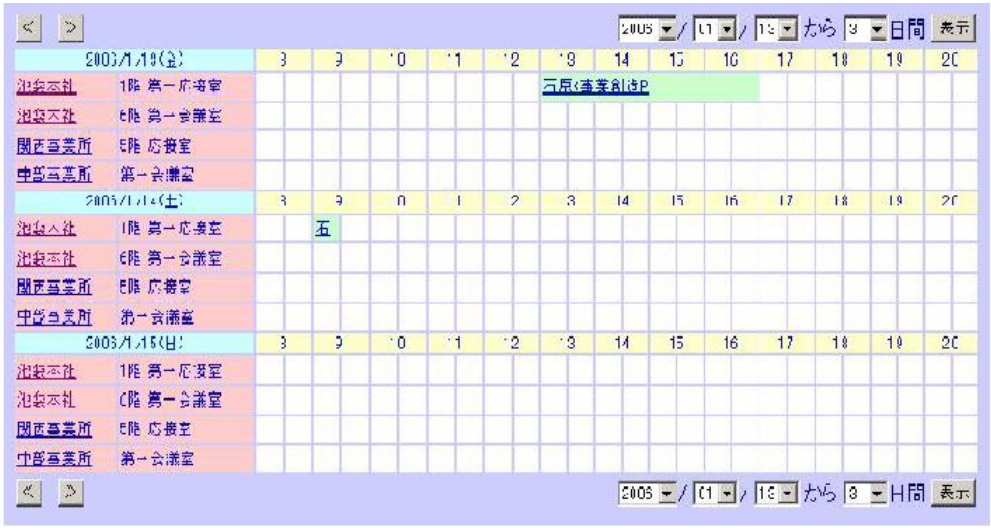

图 18　会议室状况表单

Figure 18 The table of meeting rooms status

新系统显示个人事务和会议室安排状况的表单截图为

图 19 个人事务与会议室状况的整合表单

Figure 19 The integrated table of personal schedule and meeting rooms status

CheckMeetScheduleAction 中通过获得两个表单对象（HttpUnit 中定义）之后，通过对两个表单对象的解析，获取表单对象中每个单元格的文本（text）后，重新组织再显示，该部分的实现为

```
jspout.append("<table align=\"center\" border=\"1\" bordercolor=\"#CCCCFF\" cellspacing=\"0\">");
                Integer sumdays=new Integer(msForm.getSpan());
                for(int i=0;i<sumdays.intValue();i++){
                    //output personal information
                    jspout.append("<tr>");
                    //output userid
                    jspout.append("<td colspan=\"4\">"+msForm.getUserid()+"</td>\n");
                    for(int k=0;k<scheduleTable.getColumnCount()-1;){
                        k=k+1;
                        TableCell cell=scheduleTable.getTableCell(i+1,k);
                        jspout.append("<td colspan=\""+cell.getColSpan()+"\"");
                        if(cell.getText()!=null && !cell.getText().equals(""))
                            //output personal schedule information
    jspout.append("bgcolor="+ColorConstants.GRAY+">"+cell.getText()+"</td>\n");
                        else
                            jspout.append("> </td>\n");
                        if(cell.getColSpan()>0)
                            k=k+cell.getColSpan()-1;
                    }
                    jspout.append("</tr>");
                    //output meeting information
                    for(int j=0;j<msForm.getMeetrooms().length+1;j++){
                        int temp=msForm.getMeetrooms().length+1;
                        jspout.append("<tr>");
                        for(int k=0;k<meetTable.getColumnCount()-1;){
```

```
                        TableCell cell=meetTable.getTableCell(j+i*temp,k);
                        jspout.append("<td
colspan=\""+cell.getColSpan()*2+"\"");
                        if(cell.getText()!=null && !cell.getText().equals("")){
                            if((j+i*temp)%temp>=1 && k>2 ){
                                //output meeting room arranged plan

    jspout.append("bgcolor="+ColorConstants.LIGHTYELLOW+">"+cell.getText()+
"</td>\n");
                            }else{
                                //output meeting room name

    jspout.append("bgcolor="+ColorConstants.ORANGE+">"+cell.getText()+"</td>\
n");
                            }
                        }else{
                            jspout.append("> </td>\n");
                        }
                        if(cell.getColSpan()>0)
                           k=k+cell.getColSpan();
                        else k=k+1;
                    }
                    jspout.append("</tr>");
                }
            }
            jspout.append("</table>");
```

查询股票价格的 GetStockAction 在接受到用户的请求时，调用 HttpServerBinding 处理 http 的请求对象的方法，然后解析返回的结果（soap 文件），显示股票价格。

在 Struts 的 Action 中如何获得一个 JBI 组件的实例呢？由于 ServiceMix 是在基于 Spring 开发的，所以可以利用了 Spring 提供的机制获得 JBI 组件，在 ServiceMix 的配置文件中定义的 JBI 组件其实相当与 Spring 里定义的 bean。在 Action 里可以通过一个帮助类 Hyp_SM_Util 来获得需要的 JBI 组件实例。帮助类 Hyp_SM_Util 的实现为

```
public class Hyp_SM_Util {
    private ServletConfig config = null;
    private ApplicationContext applicationContext = null;
    public Hyp_SM_Util(Action action) {
        config = action.getServlet().getServletConfig();
        ServletContext servletContext = action.getServlet().getServletContext();
        applicationContext = WebApplicationContextUtils
                .getRequiredWebApplicationContext(servletContext);
    }
    public SpringJBIContainer getContainer() throws ServletException {
        String jbiName = config.getInitParameter("jbi");
        if (jbiName == null) {
            jbiName = "jbi";
        }
        SpringJBIContainer jbi = (SpringJBIContainer) applicationContext
                .getBean(jbiName);
        if (jbi == null) {
            throw new ServletException(
            "Could not find the JBIContainer in the Spring application context for
name: "                                    + jbiName);
        }
        return jbi;
    }
    public Component getComponent(String componentName) throws
ServletException {
```

```
        String endpointName = componentName;
        if (endpointName == null) {
            throw new ServletException("the parameter compoentName is missing");
        }
        SpringJBIContainer jbi = getContainer();
        Object value = jbi.getBean(endpointName);
        if (value == null) {
            throw new ServletException(
                    "Could not find bean in the SpringJBIContainer for id: "
                            + endpointName);
        }
        if (value instanceof Component) {
            return (Component) value;
        } else {
            throw new ServletException("This is not a Component: "
                    + value);
        }
    }
    public Component getComponent() throws ServletException {
        String endpointName = config.getInitParameter("endpoint");
        if (endpointName == null) {
            throw new ServletException(
                    "You must configure a servlet config parameter of endpointRef");
        }
        SpringJBIContainer jbi = getContainer();
        Object value = jbi.getBean(endpointName);
        if (value == null) {
            throw new ServletException(
                    "Could not find bean in the SpringJBIContainer for id: "
```

```
                                    + endpointName);
            }
            if (value instanceof Component) {
                return (Component) value;
            } else {
                throw new ServletException("This is not a Component: "
                        + value);
            }
        }
}
```

通过一个 Action 实例作为初始化参数来实例化一个 Hyp_SM_Util 对象，则 Hyp_SM_Util 对象可以获得 ServletContext 对象和 ServletConfig 对象，spring 可以通过这两个对象获得 web 应用程序中的任何一个组件，包括 JBI 组件。

4.4 项目的成果与不足

在这个项目开发过程中跟踪业界一些的新的动向和趋势，应用了目前属于先进的 SOA、ESB、JBI 方面相关的技术。当然这些技术目前并没有到一个稳定成熟的阶段，不管是理论、规范、相关产品的开发，还是实际应用的解决方案都在不断的发展和成熟的过程中。

这个项目一个比较独特的方面就是，没有从程序的接口（API）集成应用程序或者数据库中获得信息，而是从界面——web 系统的页面获取需要的信息，并将两个系统的信息集成进来。不过这只是信息的集成，只将目标系统作为信息的来源，并没有通过目标系统的界面对其数据源进行新增，修改和删除的操作。要做到这些必须解决在这种集成方式下如何处理“事务（transaction）”的问题。同时现在项目中一个不足之处是，开发人员需要对了解被集成系统的页面信息，这样导致一是开发时

工作量大，而是系统的可重用性差。形成这样的原因有两方面：

a. 缺少可视化的从页面提取信息的自动化工具

b. 对基于 web 界面进行集成的软件结构的设计还不够合理

这些都还需要进一步的研究和实践去解决。

第5章 总结

5.1 对基于 ESB 实施系统集成的认识

软件行业发展至今，积累了很多成果和无形的物质财富——运行在各种各样平台上的软件系统。它们中的许多系统，或是由于现实的需要，或是别的原因，一直为人们服务着。它们可能是由不同的技术，不同的设计模式，不同的编程语言开发而成，但它们有一个共同的特点，一般访问和操作这些系统的是人，它们不易被别的系统或程序使用其资源。当某个系统要使用其资源时，在两个系统之间要出现一个代理或者适配器。当当一个系统要使用多个系统的资源时，这个系统就要使用多个代理或适配器。而 ESB 的可以作为一个代理或适配器的“管理者”的角色，使系统的灵活性打打提高，也降低了以后系统维护的成本。

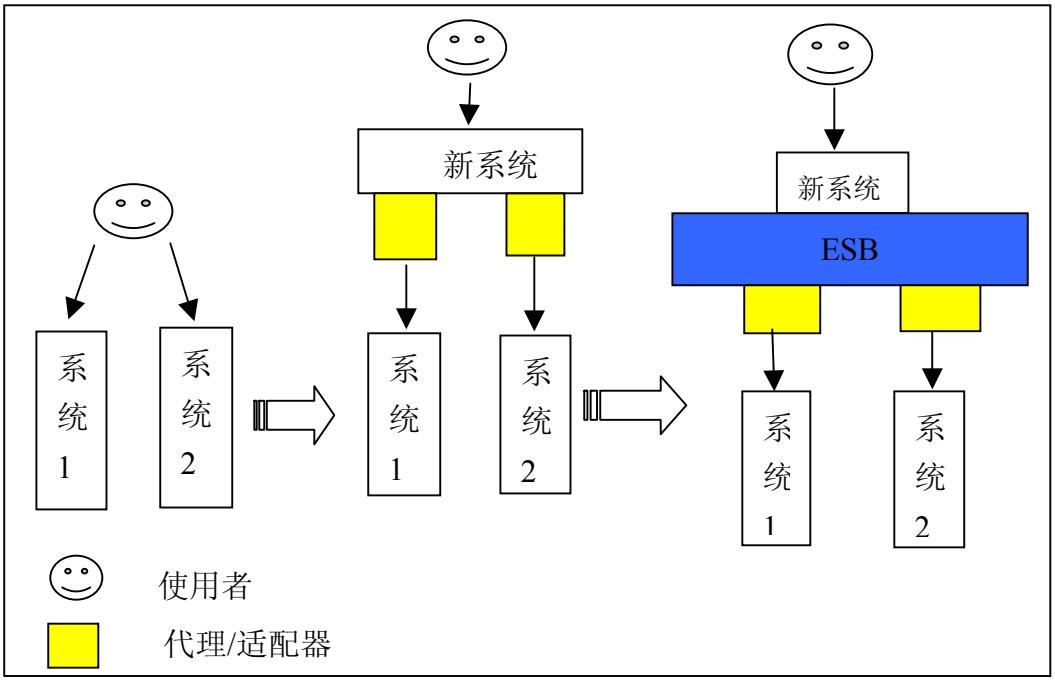

图 20 系统集成的演化

Figure 20 The evolution of integration

但是 ESB 也并不是万能的，并非通过 ESB 就可以对任何系统或模块进行访问和操作，因为在没有开放的标准或规范可以共同遵守的话，现实中只可能做到提供尽可能多的支持。而 JBI 规范就是为了实现给 ESB 的开发商，给代理或适配器的开发商，给开发 ESB 部件的开发商一个共同的标准，让它们之间的产品可以无缝连接。

5.2 对基于 web 界面集成的认识

◆ 优点

1） 缩短了开发时间：由于重用了被集成系统的功能与数据，减少了开发的工作量，开发的周期会因此缩短

2） 降低开发成本：开发周期的缩短，可以有效的节约开发成本

3） 保持被集成系统的独立性：这里的独立性不只是指被集成的结构不被更改保持独立，也指被集成系统仍然可以独立的运行，用户仍然可以单独使用该系统。

◆ 缺点与风险

1） 稳定性：当一个系统的结构越复杂，即该系统中包含的模块越多，那么该系统出现异常情况的概率就越大。要保证系统的高可靠性就要系统中的每一个模块都具有高可靠性。因此要求被集成的系统自身首先是一个稳定可靠的系统。

2） 与被集成系统界面耦合：由于通过界面与被集成系统交互，因此要求被集成系统的界面是稳定的、不作变更的。

3） 非规范化：由于界面的都为安客户特制，而不是要遵循一定的标准。虽然最终都以 html 标记语言组成的文本显示，但是在 JavaScript 等脚本语言中存在差异，有不同的“方言”存在。

基于 web 界面进行系统集成的方式，目前属于比较新的事物，为系统集成提供了一种可供选择的方案。但是由于表现层的特制性，和相对易变更的特点，基于 web 界面进行系统集成的方式可能只是对特定用户需求的一种解决方案，而不会为主流方式。企业将会更多的选择将现有

系统的功能进行 web 服务化，而系统集成时将更多采用通过 BPEL 引擎或 BPM 工具对 web 服务进行编排组合，达到应用集成的目的。

5.3 展望

目前由于水平和时间都有限，工程中存在着一些不足和待完成的工作：

假如把被集成的 web 系统看作一种数据源，则目前只是对这种数据源进行了查询操作，并未对数据源中的数据进行更改、删除、新增的操作。而进行这样的操作，要保证操作的原子性、数据的一致性等。同时这些操作依赖于被集成系统。

界面集成图形化编辑与代理自动生成工具：该工具可以访问 web 系统，并通过图形化的方式对被集成系统的界面进行裁减、组合而定制一个新界面。根据页面中被裁减部分以及目标系统的信息自动生成访问目标系统并获得被裁减部分内容的代理。

总之，如果可以降低基于界面集成过程中与被集成系统界面的耦合度，降低开发的复杂度和提高访问 web 系统代理的可重用性，则这种基于界面的集成方式可以有更广泛的应用。否则则只能作为一个可以采用而并不实用的方式。

参考文献

[1] JBI(Java Business Integration) Specification-JSR208，
http://jcp.org/en/jsr/detail?id=208

[2] Introduction to ESB，ServiceMix.org,
http://www.servicemix.org/Introduction+to+ESB

[3] Background to ServiceMix，ServiceMix.org,
http://www.servicemix.org/Background+to+ServiceMix

[4] How does routing work in ServiceMix，ServiceMix.org,
http://www.servicemix.org/How+does+routing+work+in+ServiceMix

[5] Deploying Lightweight Components Tutorial，ServiceMix.org,
http://docs.codehaus.org/display/SM/Deploying+Lightweight+
Components+Tutorial

[6] ServiceMix as an enterprise service bus，J. Jeffrey Hanson
http://www.javaworld.com/javaworld/jw-12-2005/jw-1212-esb.html

[7] ServiceMix 1.0: Apache 2.0 open source ESB based on JBI，
theserverside.com,
http://www.theserverside.com/news/thread.tss?thread_id=35996

[8] Service-Oriented Java Business Integration, Frank Sommers
http://www.artima.com/lejava/articles/jbi.html

[9] 理解面向服务的体系结构中企业服务总线场景和解决方案，
Rick Robinson, http://www-128.ibm.com/developerworks/cn/
webservices/ws-esbscen/index.html

[10] Axis 文档资料, http://www.apache.org/axis/

[11] HttpUnit 文档资料，http://httpunit.sourceforge.net/

[12] Spring 文档资料，http://www.springframewotk.org

[13] Java Business Integration (JBI) Forum,
http://forum.java.sun.com/forum.jspa?forumID=512

[14] Http 协议基础，中国协议分析网

http://www.cnpaf.net/Class/HTTP/0532918532641885.html

[15] WSDL2.0，w3c.org， http://www.w3.org/TR/wsdl20/

[16] WSDL Tutorial，w3schools.com ，
http://www.w3schools.com/wsdl/default.asp

[17] Patterns: Implementing an SOA using an Enterprise Service Bus，IBM red book，http://www.redbooks.ibm.com/abstracts/sg246346.html?Open

[18]Patterns: Integrating Enterprise Service Buses in a Service Oriented Architecture，IBM red book，
http://www.redbooks.ibm.com/abstracts/sg246773.html?Open

[19] SOAP Tutorial，w3schools.com ，
http://www.w3schools.com/soap/default.asp

[20] Spring Reference Document, springframework.org
http://static.springframework.org/spring/docs/1.2.x/reference/index.html

[21] Introduction to the Spring framework, Rod Johnson
http://www.theserverside.com/articles/article.tss?l=SpringFramework

[22] What Is Service-Oriented Architecture, Hao He
http://www.xml.com/pub/a/ws/2003/09/30/soa.html

[23] Service-oriented architecture (SOA) definition,
http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html

[24] SOA 实践 -- 使用 IoC 和 AOP 重构 SOA 应用, 易立、赵勇，IBM 中国软件开发实验室，
http://www-128.ibm.com/developerworks/cn/webservices/ws-refactoring/

[25] 通过 IBM WebSphere Business Integration Adapters 集成技术和业务解决方案, Ganapathi Adimurthy
http://www-128.ibm.com/developerworks/cn/websphere/library/techarticles/0403_adimurthy/0403_adimurthy.html

[26] 在 WebSphere Business Integration Message Broker 中部署一个复杂流,Andrew Lau 等，http://www-128.ibm.com/developerworks/cn/websphere/library/techarticles/0404_zawawy/0404_zawawy.html

[27] 基于 WebSphere Business Integration Message Broker 的消息转发和变

换，卜冠英， http://www-128.ibm.com/developerworks/cn/websphere/library/techarticles/0601_buguany/

[28] 与 Enterprise Information Integration 技术结合使用 Service Data Objects，Jacques Labrie 等, http://www-128.ibm.com/developerworks/cn/db2/library/techarticles/dm-0407saracco/

[29] 李健行，系统科学原理与现代管理思维，湖南师大出版社，1994

[30] 陈忠、盛毅华，现代系统科学学，上海科学技术文献出版社，2005

[31] Developing a Service Engine component, sun.com http://java.sun.com/integration/reference/techart/jbi/index.htm

摘要

目前无论是应用服务器的供应商，还是中间件产品的供应商，抑或是系统集成商都在关注着企业服务总线（Enterprise Service Bus, ESB），以及 ESB 的出现对实施面向服务架构（Service Oriented Architecture, SOA）系统的影响。同样开源组织也加到开发 ESB 产品的这个浪潮中。在 ObjectWeb 组织下的 Celtix 是在 IONA 公司的 ESB 产品 Artix 基础上进行开发，而且准备全面支持 JBI 规范；在 Apache 组织下 ServiceMix 是一个基于 Spring、完全遵循 JBI 规范的 ESB 产品。

本文沿着 SOA－ESB－JBI－ServiceMix 这条路线，从一个领域－SOA，到该领域的某类型产品－ESB，到针对着类型产品的开放标准－JBI 规范，再到具体一个 ESB 产品－ServiceMix，逐步进行深入的调研。最终通过一个基于 ServiceMix 实施系统集成的具体案例来作进一步的分析。

在软件系统正在变得越来越复杂，越来越庞大，而且要求分布异构的多个系统可以良好交互、协作而组成一个新的系统的情况下，面向服务架构应运而生。SOA 的出现不是对面向对象的颠覆，而是向更高层次的发展，就是面向对象是对面向过程的发展一样。在面向过程的软件中，程序＝数据结构＋算法；在面向对象的软件中，程序可以看作由各种对象组成，而对象＝数据结构＋算法。在面向服务架构的软件中，可以看成由各种服务和这些的服务的控制流组成。

企业服务总线的出现，为系统集成提供了一个新的解决方案的同时，也为实施 SOA 提供了新的思路。改变了在 SOA 发展的初期，服务消费者与服务提供者端到端的调用方式。BPEL 引擎或其他服务的编排工具可以作为一个模块集成到企业服务总线上，使服务消费者可以调用一组服务；还可以在企业服务总线上集成事务处理的模块，来处理包括长生命周期的各种事务。

目前 SOA 领域中各种技术、解决方式、产品和规范都不断涌现，正处于一个快速发展的时期。JBI 规范的发布目的是为 ESB 产品提供一

个开放标准。它包括规范 ESB 内部的消息模型，消息路由方式，ESB 提供的基础服务以及 ESB 与被集成模块之间的接口等。ESB 有与应用服务器集成应用的趋势，目前 ServiceMix 可以与 Geronimo、JBoss、JOnAS 应用服务器集成。而 Sun 公司开发的 Open－ESB 将集成到 Sun 公司下一代的应用服务器 GlassFish 上。而且 JBI 规范中也称将来 J2EE 规范有可能将引入 JBI 规范。

本文中的案例是一个基于开源 ESB 产品 ServiceMix 的系统集成项目。ServiceMix 是利用 Spring 技术对 JBI 规范的实现。进行集成时，对于不同类型的被集成系统需要不同类型的绑定组件（Binding Component）。例如对于与外部环境通过 JMS 消息通信的系统，被集成时就需要有可以将 JMS 消息和 JBI 规范中定义的规范消息（Normalized Message）相互转换的绑定组件。如果集成中需要增加新的功能或业务逻辑，则可以根据情况（模块的重用性、内聚度等）选择是否创建一个或多个服务引擎。

在本文的案例中，被集成的两个系统都是 web 系统，要求只能通过界面来访问这两个系统。而目前尚无这样的绑定组件可以选用，因此需要开发一个可以将通过 http 协议传输的 html 文本与规范消息之间进行转换。这个绑定组件因此需要具有类似浏览器的功能：基于 http 协议通信，对 html 文本进行解析，保持与服务器端的会话，同时在这个案例中，还要求用户与访问代理的一一对应。同时本案例中要求可以访问远程的 web 服务，因此还需要一个可以将 soap 消息与规范消息之间进行转换的绑定组件。ServiceMix 中提供有多种这种类型的绑定组件可以选用 SAAJ（SOAP with Attachments for Java）绑定组件，xFire 绑定组件、ActiveSOAP 绑定组件等。

本文中对可以部署在 ServiceMix 上的 JBI 组件也进行了分析。在 JBI 规范中将组件分为服务引擎（Service Engine）和绑定组件（Binding Component）两种。而根据开发方式和部署的方式不同，运行在 ServiceMix 上的 JBI 组件可以分为轻量级组件和标准 JBI 组件。轻量级组件开发简单，但依赖于 ServiceMix 本身；标准组件开发复杂，但可以被部署到任何一个遵循 JBI 的 ESB 上。

案例成功地解决了在系统前端如何同时与多个 web 系统进行交互，

并将 web 界面进行整合再显示，以及如何如何保持前端用户与目标系统会话。在该案例中的解决方案适合对于 web 界面不作变更，同时只允许从系统界面进行集成的场景。这种解决方案不影响被集成系统的独立运行，不需对被集成系统做任何修改。但同时由于界面的不规范的特点，访问界面的模块与界面的耦合性强，重用性差。目前该案例中的方式可以作为一种可行的解决方案，将来如果可以有图形化编辑工具进行多个页面的整合，和基于界面自动化生成访问代理的工具，也许可以使该种解决方案的应用场合更广泛一些。

Abstract

The ESB has come to various vendors' attention, including middleware, application server and the integration products providers all talk about it. The emergence of ESB, is a highlight in the middleware and integration field. Lots of vendors have released their products or plans about it. But there was not a open standard for various vendor until the JBI specification is released. The open source organizations also are developing their ESB product. Celtix is hosted ObjectWeb, which is base on IONA's Artix. ServiceMix is developed by LogicBlaze, Gluecode, MortBay and others. It's a incubator project under Apache.

This article is along with the thread of SOA-ESB-JBI, introduces SOA, ESB ,JBI and the open ESB-ServiceMix. Then analyzing a concrete case, which integrated two web system base on ServiceMix.

When the software system is changing more and more complex, more and more huge, moreover requires distributed and heterotypic systems can interoperation well. So the SOA is emerging. The SOA approach is not opposite to the object-oriented, but is to the higher level development,. In the process-oriented software, program = data + algorithm; In the object-oriented software, the program may regard as by each kind of object to be composed, but object = data + algorithm. In the SOA software, may regard as by each kind of service and the control flow of these services is composed.

There is a new solution for the system integration because the emergence of enterprise service bus. The ESB's emergence also provide a new approach for implemented SOA at the same time. The end-to-end invocation which is a common way in the initial period of SOA development is changed. The BPEL engine or other services orchestration

tools may be integrated into the enterprise service bus, enables the service consumer to be possible to consume a group of services; Also may integrate the module on the enterprise service bus which business processes, processes including long life cycle each kind of business.

At present in the SOA domain each kind of technology, the solution, the product and the standard all unceasingly emerge. The goal of JBI specification is providing an open standard for the ESB product. This specification describes a messages model, and the message routing, the foundation services which ESB provides, the interfaces of ESB, and so on. There is a tendency of ESB integrated with the application server. At present, ServiceMix can be integrated with Geronimo, JBoss, and the JOnAS application server. But Sun Corporation ‘s ESB product Open - ESB will integrate to Sun’s next generation application server GlassFish. And the J2EE specification will reference the JBI specification in the future.

In this article case, there are two systems which integrated both is the web system, and only can access these systems via theirs presentation level interfaces. At present, still did not have such binding module to be possible to select, therefore needed to such a binding which can communication with http and do transformation between html text and the Normalized Message. So this binding needs to have some common functions with a browser. Based on the http protocol communication, analysis to the html text, and maintenance the session between the client and the server, at the same time, but also requests the user and the visit proxy is one-one map in this case.

In this article, analysis the JBI component which can be deployed on ServiceMix. The JBI Component is divided into two different type, the service engine and binding component. While according to the different approach of development and the deployment, there are two different component type in the servicemix, the lightweight component and the standard JBI component. The lightweight component development is simple, but relies on ServiceMix itself; The standard component development is complex, but may deploy any ESB which support or be compliant JBI

specification.

The problems how to communication with multiple target systems and how to integrate the web page is solved successfully in the case. As well as solve how maintains the session between the client and the server. The solution of this case suits the scene that the target web system's presentation isn't changed, and the system only accept the accessing via its presentation level interface. This kind of solution does not affect by the integrated system independent running, and does not do any change on the integrated system. But the integration base on presentation will relies on target system's surface, the coupling is strong. In future if there might be graphic edition tools and automatic generation tools for editing and generating the proxy, this solution may be applied in many scenes, but it only is a feasible solution at present.

Keywords: ServiceMix, JBI, ESB, Integration

致谢

在论文的写作过程中，得到了秦贵和老师和上海汇软的钟友良的殷切关怀和细致认真的指导，没有两位老师，论文也很难顺利完成，在此要对二位老师特别感谢！二位老师同时在实习单位的同事们和在学校的同学们也给了我很多的帮助，他们给了我鼓励，也给我一些很好的意见。特别是我的同事马中游和胡建强，在实习的过程中，他们给了我很多帮助，对我的提问不厌其烦的去回答，对我面临的困难他们竭尽全力去解决，对此我也十分感谢！

在两年研究生的学习生活中，马东辉等老师提供了一个不可缺少的良好学习环境，对此也表示忠心感谢！还要谢谢我的父母和亲朋，他们的鼓励与帮助也是我进步一个动力。

导师与作者简介

导师简介

姓名	秦贵和	性别	男	出生日期	1962.10
民族	汉	学位	博士	出生地	山东高密
职称	教授，博士生导师				
E-Mail	qingh@jlu.edu.cn				

著作与成就

研究方向：实时与嵌入式系统。

先后完成国家攻关项目、基金项目、省部级项目及开发项目二十余项；发表论文50余篇，其中有10余篇次被SCI、EI和ISTP检索机构收录；出版著作两部；获得教育部科技进步二等奖一项，机械部科技进步三等奖一项；得到教育部“高校骨干教师资助项目”、 “高校优秀青年教师资助项目”、 韩国KFAS基金的资助。被评为吉林省有突出贡献的中青年专业技术人才。

作者简介

姓名	刘磊	性别	男	出生日期	1982.8
民族	汉	学位	硕士	出生地	河南孟州
E-Mail	guichi_jlu@yahoo.com.cn				

学习经历

2000.9~2004.7 吉林大学计算机科学与技术学院

2004.9~2006.7 吉林大学软件学院

科研情况及发表论文

科研项目

发表论文：