基于 J2EE 设计面向服务体系结构框架

Design Service – oriented Architecture Frameworks with Based on J2EE

邓　武　杨鑫华　(大连交通大学 软件学院 116052)

摘要:面向服务的体系结构因其松散耦合与互操作性而成为许多企业应用。在本文中将利用 J2EE 来设计面向服务的体系结构(SOA)框架。通过采用 SOA 框架,企业可以最大程度地减少系统间的耦合,从而提高可重用性。

关键词:面向服务的体系结构 J2EE EJB Web 服务

1　SOA 和 Web 服务的简介

1.1　面向服务的体系结构

面向服务的体系结构(service – oriented architecture,SOA)是一种用于构建分布式系统的方法,也是一个组件模型,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以一种统一和通用的方式进行交互。它定义了以下的设计原则:

(1) 模块化将功能划分成更小的,可重用的模;

(2) 松耦合客户端和服务器端,相互之间不必紧密依赖;

(3) 封装性围绕着功能模块化,控件内部对于接口进行了很好的定义和封装。

客户和系统分析员都关注 SOA,因为 SOA 能够提供的两大优点:灵活性和敏捷性。以前的体系结构,很难对不同的功能和系统进行集成,很难对变化的商务需求和竞争需求做出及时的反应。面向服务的体系结构(SOA),具有以下优点:

(1) 能够更快地集成第三方软件;

(2) 采用商务流程管理工具,能够更容易的对复合系统进行配置;

(3) 通过平台管理和控制,能够更安全的升级某个单独地服务;

(4) 按照服务来划分开发任务,更好的支持了分布式的协同开发。

下面通过图 1 来说明 SOA 中的不同角色。

从图 1 可以看出,SOA 结构中共有三种不同角色:

(1) Service broker:注册已发布的 Service provider,对其进行分类,并提供搜索服务;

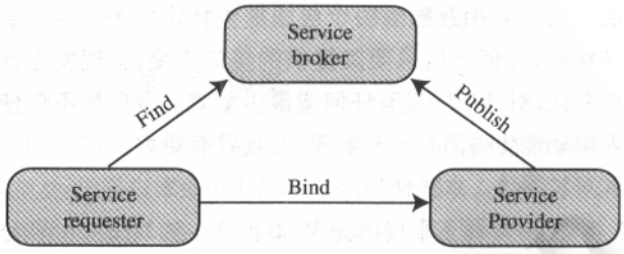

图 1　面向服务的体系结构(SOA)

(2) Service requester:利用 Service broker 查找所需服务,然后使用该 SOA 中的组件必须具有上述一种或多种角色;

(3) Service provider:发布自己的服务,并且对使用自身服务的请求进行响应。

在这些角色之间使用了三种操,分别为:

(1) Find 操作:使 Service requester 可以通过 Service broker 查找特定种类的服务。

(2) Publish 操作:使 Service provider 可以向 Service broker 注册自己的功能及访问接口。

(3) Bind 操作:使 Service requester 能够真正使用 Service provider。

为了支持结构中的三种操作(Publish、Find 和 Bind),SOA 需要对服务进行一定的描述,服务描述应具有几个重要特点:首先,它要声明 Service provider 语义特征。Service broker 使用语义特征将 Service Provider 进行分类,以帮助查找具体服务。Service requester 根据语义特征匹配那些满足要求的 Service provider。

其次,服务描述应该声明接口特征,以访问特定的服务。最后,服务描述还应声明各种非功能特征,比如安全要求、事务要求、使用 Service provider 的费用等等。

Web 服务是建立在开放标准和独立平台的协议基础之上。Web 服务通过 HTTP 使用 SOAP(一种基于 XML 协议),以便在服务提供者和消费者间进行通信。服务通过 WSDL(Web Service Definition Language)定义的接口来公开,WSDL 的语义用 XML 定义。UDDI 是一种语言无关的协议,用于和注册中心进行交互以及查找服务。这些特点都使得 Web 服务成为开发 SOA 应用程序的最佳选择。

1.2 基于 J2EE 开发 SOA/Web 服务

J2EE(Java 2 Platform, Enterprise Edition)是美国 Sun 公司推出的一种全新概念模型,它采用分层结构,层与层之间相互独立每个层面提供特定服务。在分层结构中,任何一层只要其提供的接口不变,实现发生什么变化,对其他层没有任何或最少影响。分层后不仅使各层功能变得简单且易实现,而且具有更好的可维护性和可扩展性。本文使用的是 J2EE 1.4 版本进行设计。

使用 J2EE 1.4 目的是 Web 服务,在 J2EE 1.4 平台下:

(1) 允许 J2EE 应用组件暴露为基于 HTTP/SOAP 的 Web 服务;

(2) 和原有的 Web 服务进行整合;

(3) Web 服务的关键技术:JAX – RPC;

(4) J2EE 1.4 下的 Web 服务框架:Web services for J2EE。

J2EE 1.4 平台升级的新增加技术大部分和 Web 服务相关。在 J2EE 1.4 平台下,开发、部署、发现 Web 服务变得非常方便。尽管 J2EE 1.4 平台对 Web 服务方便提供了升级,但是 Web 服务仅是 J2EE 平台中一种使用服务的通道,所以不需要改变 J2EE 的构架,并且原有的 J2EE 组件可以非常容易地暴露为 Web 服务和 J2EE 平台的优点仍然对 Web 服务适用。

在 J2EE 1.4 下,Web 服务客户可以通过两种方式访问 J2EE 应用程序。客户访问用 JAX – RPC API 创建的 Web 服务;在幕后 JAX – RPC 使用 Servlet 来实现 Web 服务。Web 服务客户也可以通过 Bean 的服务端点接口访问无状态 Session Bean;Web 服务客户不能访问其他类型的 Entity beans。第二种选择是公开无状态 EJB 组件作为 Web 服务,它有很多优势:

(1) 对服务的安全访问。Entity beans 允许在部署描述符中声明不同方法级别的安全特性。使用 EJB 组件作为 Web 服务端点,把这种方法级别的安全性也带给了 Web 服务客户。

(2) 并发支持。作为无状态 Session bean 实现的 EJB 服务端点不必担心多线程访问,因为 EJB 容器必须串行化对无状态 Session bean 任何特定实例的请求。

(3) 利用现有的业务逻辑和流程。在许多企业中,现有的业务逻辑可能已经使用 EJB 组件编写,通过 Web 服务公开它可能是实现从外界访问这些服务的最佳选择。

(4) 事务问题。EJB 服务端点在部署描述符规定的事务中运行。容器处理事务,因此 bean 开发人员不需要编写事务处理代码。

(5) 可伸缩性。几乎所有 EJB 容器都提供了对无状态 Session bean 群集的支持。因此当负载增加时,可向群集中增加机器,Web 服务请求可以定向到这些不同的服务器。通过把 Web 服务模型化为 EJB 端点,可以使服务具有可伸缩性,并增强了可靠性。

(6) 池与资源管理。EJB 容器提供了无状态 Session bean 池。这改进了资源利用和内存管理。通过把 Web 服务模型化为 EJB 端点,使 Web 服务能够有效地响应多个客户请求。

2 设计 SOA/Web 服务框架

假如有一家企业,它的各种系统需要彼此交互。而且其中一些应用程序还需要对外界公开,以便不同的业务合作伙伴与它们进行交互,还需要为各种应用程序设计基于 Web 的解决方案。所以需要设计一种松散耦合的基于服务的系统。这些方面的考虑将转向 Web 服务或 SOA 框架,通过无状态 EJB 组件把各种服务和业务流程公开为 Web 服务。图 2 说明了企业内部应用的 SOA 框架。

这是一种典型的 MVC 框架。下面列出了 SOA 框架中进行交互的各种组件。

(1) 客户(Client)。用户通过 Web 浏览器与不同的应用程序交互,浏览器作为应用程序的客户。

(2) 应用程序控制器(Application Controller)。是主控制器 servlet。它负责初始化、委派请求和响应请求处理程序。

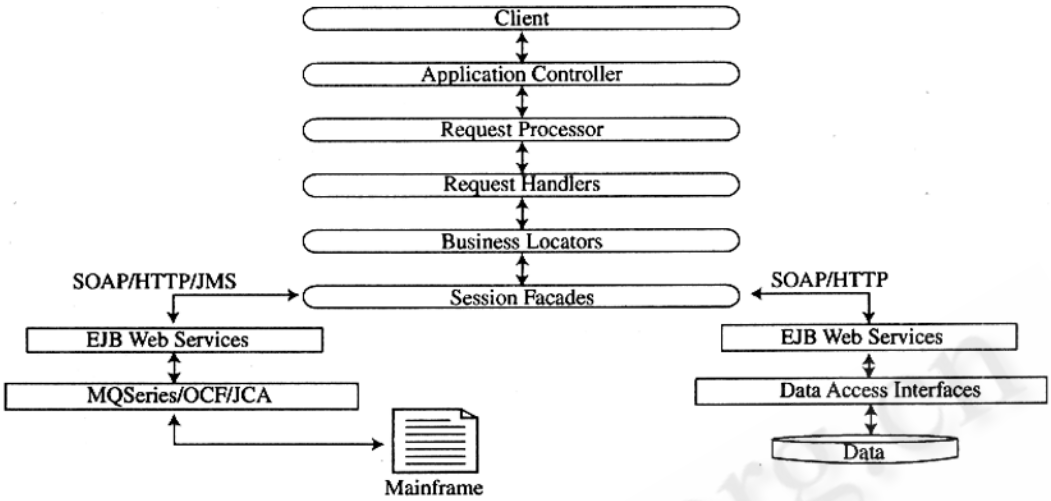

图 2　企业内部应用的面向服务体系结构

（3）请求处理程序（Request Processor）。这是一个 Java 类，通过调用相应的请求执行程序完成要求的处理，对请求进行预处理。

（4）请求执行程序（Request Handlers）。请求执行程序完成具体请求的活动，比如与服务交互，向不同的企业信息系统（enterprise information systems，EIS）增加或检索信息。请求执行程序依靠业务定位程序发现相应的服务，然后通过这些服务访问需要的 EIS 信息。

（5）业务定位程序（Business Locators）。这些程序负责隐藏查找服务的复杂性，并提供缓存逻辑。业务定位程序可以采用多种形式，比如 Web 服务定位程序、EJB 组件定位程序或者 JMS 定位程序。

（6）会话 Facades（Session Facades）。通过聚合来自多个系统或服务的方法，简化复杂对象的视图。Session Facades 是 EJB Web 服务方法的包装器。

（7）EJB Web 服务（EJB Web Services）。根据 EJB 1.4 规范，Web 服务端点可以模型化为无状态的会话 bean。

（8）数据访问接口（Data Access Interfaces）。使用不同的技术（如 EJB－CMP、JDO、DAO）和不同的持久性技术访问 EIS，所使用的访问技术取决于接口需求以及获取、插入或更新的数据量。

（9）MQSeries/JCA/CCF。现有的基于主机的服务可以公开为 Web 服务，从而向外界展示它们。Web 服务客户使用基于 HTTP 的 SOAP 协议与 EJB Web 服务交互。EJB 方法通过 JMS 协议向 MQSeries 队列发送请求。主机端的 MQSeries 服务器触发相应的基于 COBOL 的程序，后者为与后台系统进行交互提供必要的逻辑。然后这些程序把响应返回到队列中，应用程序逻辑检索这些响应并返回给 EJB 方法。SOAP 消息可以通过不同协议进行传输，比如 HTTP、HTTPS 和 JMS。

3　向外界公开服务

如果打算向外部用户公开服务，需要某种安全约束来保证只有授权的用户才能访问服务。一种方法是提供另外的 Web 服务层，过滤掉禁用的 Web 服务请求，并提供登录和安全约束。这种过滤方式还应提供一种工具，向每一客户只公开授权给该用户的服务子集。图 3 说明了企业外部应用的面向服务体系结构。

以下是这种体系结构的基本功能单元：

（1）外部客户（External Clients）。包括基于 Web 的客户、移动客户或者使用. NET 环境或其他编程语言编写的客户。所有这些客户都为不同的服务发送请求。只要遵循 WS－I Profiles 就不会出现互操作性的问题。

（2）企业防火墙（Corporate Firewall）。根据其安全策略，该企业在 Intranet 和 Internet 之间架起了防火墙，对收到的分组信息进行限制。

（3）Web 服务网关（Web Services Gateway）。该例使用 WebSphere Application Server 5.0 中的 Web Services Gateway（WSG）作为公开外部服务的网关。它是一种中间件产品，在调用 Web 服务时提供了 Internet 和 Intranet 之间的中间框架，使用 WSA，开发人员和企业可以安全地对外公开 Web 服务，防火墙之外的客户也能调用这些服务。

（4）EJB 服务（EJB Services）。EJB 服务没有任何变化。

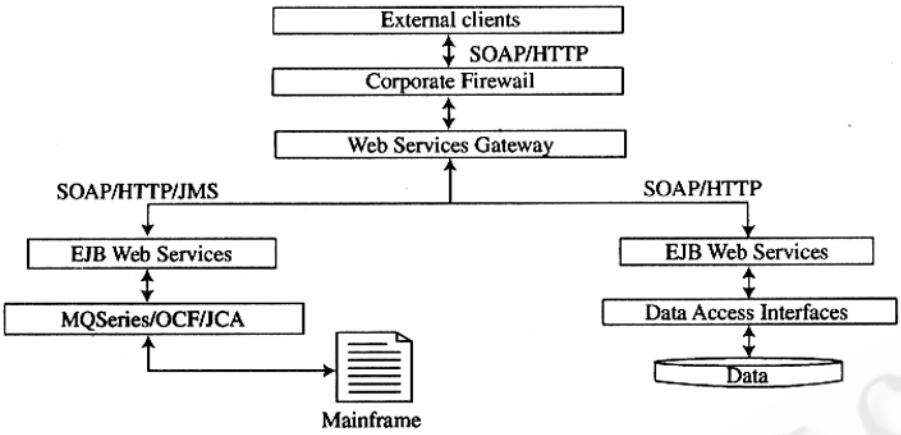

图 3　向外界公开的面向服务体系结构

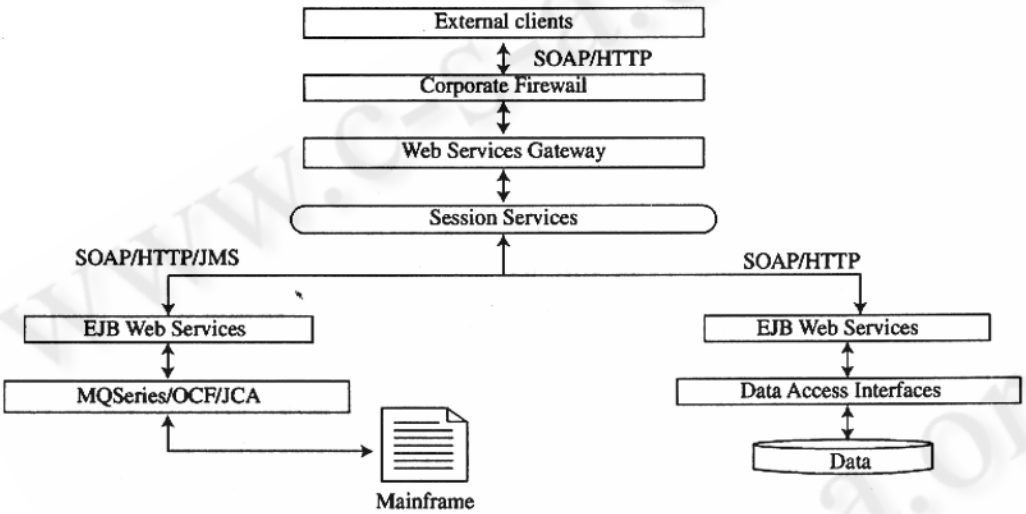

图 4　提供大粒度服务的面向服务体系结构

4　使用 EJB 组件实现大粒度的服务

前面所述的是向客户公开小粒度的 Web 服务。假设客户要进行在线资金转移，这种情况下提供单一的、大粒度的接口显然更加合理，让用户提供所有必要的信息，包括传输的金额、发出和接收的银行信息等等。此外，这类情形中验证必须在执行任何业务逻辑之前完成。在设计 Web 服务方法时必须考虑到这些问题，还要记住除了网络调用之外所有解析与规划 XML 请求和响应的开销。

考虑到这些因素，可以把 Session Facades 模型化为 EJB Web 服务端点。Session Facades 可以在把请求委派给相应的 Web 服务方法之前首先验证请求。这样就可以向 Web 服务客户提供大粒度的服务。图 4 说明了企业外部应用的面向服务体系结构的下一次迭代过程。

这里，主要的实现和图四中所示的相同。唯一的区别在于已经公开了 Session Facades 作为 Web 服务端点。EJB Web 服务可以模型化为本地接口而不是远程接口。使用 Session facades 和方法级安全性，可以限制要执行的服务。使用 WSG 也能为 Web 服务客户施加安全措施。根据需要，可以在大粒度和小粒度服务间的某种结合，通过调整 Web 服务网关中间件来向外部客户公开两种服务。

5　结论

采用 WSDL 文件形式的 Web 服务接口可以发布到商业注册中心，从而使客户能够动态查找这些接口。如果交易伙伴已经知道这些服务，也不一定要进入商业注册中心，但是全球服务需要公共注册中心，以便客户能够查找可用的服务。

参考文献

1　马琳、杨旭、郑谦等译，J2EE 宝典[M]，北京 电子工业出版社，2002。
2　Ed Roman, Mastering enterprise JavaBeans and Java 2 Platform, Enterprise Edition, 2002.
3　SOAP version 1.2, W3C working draft [S], 2002.
4　WSDL4J project [CP/OL], http://www-124.ibm.Com/developerworks/projects/wsdl4j/