中国农业大学学报 2001, 6(4): 85~88
Journal of China Agricultural University

分布式中间层 EJB 的研究与开发

高奇微① 梁言兵
(中国农业大学电气信息学院)

摘 要 通过对分布式中间层 EJB (Enterprise Java Beans)的研究, 以及对服务器端的业务处理组件(Session Beans)和数据组件(Entity Beans)的开发, 说明 EJB 技术可以为分布式的多层结构提供服务器端的组件模型。

关键词 中间层; 客户; 服务器; 组件模型; 支持平台
中图分类号 TP 393.09

Study and Implementation of Distributed Middleware

Gao Qiwei Liang Yanbing
(College of Electricity and Information, CAU)

Abstract By the study of Middleware EJB (Enterprise Jave Beans) and the development of session beans and entity beans of server port. It was explained that the EJB technology can be used to provide the Beans model for distributed architecture of multi-tier.

Key words middleware; client; server; beans model; container

长期以来, 基于客户机/服务器(Client/Server, C/S)结构的应用系统主要有客户端和服务器端两层。在两层 C/S 结构系统中, 客户端包含了大量的数据处理逻辑, 几乎所有的应用逻辑都在客户端实现, 导致客户端的应用程序越来越复杂, 给开发人员带来了大量的移植工作; 同时对如此"肥"且节点众多的客户机的维护更是一件庞杂的工作。而服务器端, 由于每个客户端都必须与数据库保持连接, 且与数据库进行频繁交互, 从而也限制了系统支持的用户数量。因此广大用户在享受 C/S 两层结构带来的便利时, 也承受着越来越大的成本投入和使用与管理上的麻烦。随着计算机硬件的大幅度降价, 服务器的价格也越来越低廉, 因此两层 C/S 结构正逐渐退出舞台, 取而代之的是一种分布式多层 C/S 结构。

1 多层 C/S 体系结构

1) 人机交互层。在多层 C/S 结构中, 处于第 1 层的是客户端表示层, 与两层 C/S 结构中的"肥"客户端不同, 多层 C/S 结构的客户层仅仅是指图形用户界面, 用来实现人机交互和数据显示, 负责向 Web 层请求应用处理, 如信息的查询、更新等, 它不表示任何应用逻辑。其运行代码可以从位于中间层的 Web 服务器下载到本地的浏览器中执行, 几乎不需要任何管理工作, 只要 Web 服务器对用户进行身份验证, 就可以用超文本传输协议将结果回送给 Web 浏览器, 予以显示。

收稿日期: 2001-01-15
①高奇微, 北京清华东路 17 号 中国农业大学(东校区) 142 信箱, 100083

2) 中间层。可由 1 台或多台服务器组成。中间层可继续细分为W eb 层和应用服务层[1]。

W eb 层　由W eb 服务器及其控制下的W eb 服务器扩展构成, 是客户与应用服务层的接口和中转站。一方面接收浏览器的客户请求, 由W eb 扩展模块对请求的参数解释、重组为请求信息后, 传送给应用服务器, 将返回的处理结果送至浏览器; 另一方面, 客户可以预设感兴趣的服务内容, 由服务器主动将信息"推"给客户, 增强信息发布的及时性。

应用服务层　实现核心业务逻辑服务和对数据库的访问等工作, 发出请求信息, 根据服务要求与资源管理层(如数据库服务器) 交互, 实现资源的存取和返回应答等功能。可利用多进程、多线程、动态负载平衡、对象管理等特性, 提供高性能的数据访问和快速响应。该层具有良好的可扩充性, 可以随着应用的需要任意增加服务器的数目。

3) 资源管理层。由数据库系统和已有系统组成, 负责管理应用系统的信息资源。根据应用服务器的请求进行资源操作, 并将操作结果返回应用服务器。

笔者采用美国 SUN 公司 J2EE (Java 2 企业版) 集成框架下的多层C/S 结构(图 1), 对中间层 EJB 进行了研究。在J2EE 集成框架下开发的应用程序不依赖任何特定的操作系统、中间层和硬件, 只要J2EE 的程序设计合理, 只需开发 1 次就可部属到多种异构平台上。J2EE 还提供了更为广泛的负载平衡策略, 允许多台服务器集成部属, 这种部属可超过数千个服务器。J2EE 集成框架提供的服务简化了应用程序的开发, 其服务包括命名服务、部属服务、事务服务和安全服务等。

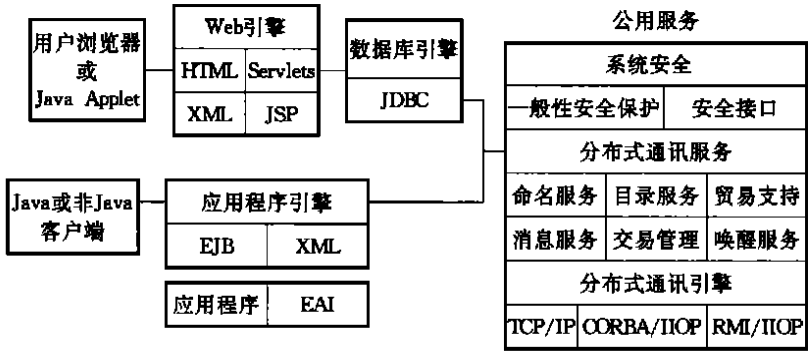

图 1　J2EE 多层C/S结构组成

多层C/S 结构实现了客户端、数据、数据处理逻辑的完全分离。与两层C/S 结构相比其通用性和代码可重用性更好, 并且具有更强的可维护性、更大的灵活性和更高的安全性。总之, 多层C/S 体系结构非常适合大数据量的商业事务系统, 特别是基于W eb 的应用中, 需要多层C/S 体系结构支持瘦客户机及浏览器的快速A pp let 下载。

2　中间层 EJB 的研究

2.1　组件模型

由于中间层要访问和管理各种各样的资源和数据, 建立这样的中间层往往需要耗费大量的时间和金钱, 且工作非常复杂。为了部分的简化这项复杂的工作, 应用程序的开发采用组件模型技术。组件模型是应用类型的软件单元, 它定义了组件的基本体系结构、组件界面结构、与其他组件及运行环境相互作用的机制等。采用组件模型技术可以简化开发过程, 达到软件重

用、高层开发、通过工具进行自动化开发等目的。组件模型思想在目前的软件开发界已得到广泛应用, Java Bean 的客户应用程序、EJB 的服务器端应用组件、Web 组件、Applet 和 COM + 等都是组件模型的例子。

2.2　服务器端的组件模型

由于 J2EE 的 Java Bean 提供了基于组件的开发机制, 过去几年 Java Bean 已经开发出许多客户端图形界面组件。一个标准的 Java Bean 是一个客户端组件, 一般是可视化的组件。根据 Java Bean 的属性方法和对事件的定义, 在设计或运行时可对客户端的 Java Bean 进行操作, 但 Java Bean 不能被其他客户机程序存取或操作, 而可以在多个应用系统中重用, Java Bean 不一定要用于 Client/Server 系统。而 J2EE 的 EJB (Enterprise Java Bean) 技术是在 Java Bean 组件模型的基础上进行了扩充的, 可以用于实现服务器端组件的开发。EJB 开发的组件没有用户界面, 完全位于服务器端, 可以由多个 Java Bean 组成, 是 Client/Server 系统的一部分; 可以和远程的客户端程序通信, 并提供一定的功能。EJB 技术和 Java Bean 的一个重要区别是 EJB 提供了网络功能, 因此, 在网络计算环境中组件开发的最好途径是由 EJB 提供服务器端的组件, 而由 Java Bean 提供客户机端的组件。

2.3　EJB 的支持平台

EJB 技术的支持平台营造了各种组件的运行环境, 主要控制组件和组件的生存期以及组件的实例化等, 负责提供较底层的服务, 如初始化, 程序释放, 内存清空以及负载均衡、状态监测等[2]。凡满足 EJB 支持平台条件所开发的组件可以在任何一个支持 EJB 的系统环境中运行, 还可以根据用户的需求将应用系统分解: 根据用户当前所需要的功能提供相关的组件; 随着用户新的需求随时下载新的组件; 用户没有用到其功能的组件可以驻留在服务器上。因此系统管理员可以很容易地重新配置系统的负载, 通过组合组件就可以得到一个分布式应用系统。这样, 应用系统的开发人员不需要理解底层的事务处理细节、状态管理、多线程、资源共享管理以及其他复杂的底层 API 细节。组件开发人员和应用开发人员也不需要实现系统中的一些复杂的逻辑结构, 因为 EJB 的支持平台已提供了对这些服务的自动管理和控制。开发人员可以集中精力开发应用系统的核心功能, 开发完之后可以部属在任何支持 EJB 的平台上, 即使该系统是由不同的开发商提供的, 也不需要重新编译或对代码进行修改。随着需求的改革, 应用系统可以不加修改地从一个服务器迁移到其他功能更强、更复杂的服务器上。还可以根据应用需求迅速地修改服务器端的组件, 并与组件在网络中的位置和应用无关。另外, EJB 将应用逻辑分成可再利用和可扩展的代码段, 开发人员只需在部署时定义 Beans 的事务属性, 而不必开发代码来管理易于出错的事务边界, 从而大大减少了开发服务器端应用系统的工作量, 提高开发人员的生产率。

3　服务器端组件模型的开发

本研究采用 EJB 技术来实现服务器端组件模型的开发, 编程采用 Java 2 程序语言, 实现过程如下。1) 组件模型的设计: 在抽象层描述系统中, 组件包括接口、属性及其关联等信息; 2) 组件模型的部属: 根据实际运行环境, 决定组件的分布和组件的实现细节; 3) 组件的具体化: 将逻辑转化为物理连接, 将组件间的连接以代码的形式表示出来; 4) 产生代码: 包括对象的初始化、对象实例间的链接、链接库指定、以及编译开关的设置; 5) 编译、链接、并产生最终代码。

3.1 业务处理组件的开发

业务处理组件(Session Bean)主要进行数据存储和数据交换。在客户会话期间,业务处理组件是对远程任务请求的响应,它通过方法的调用,掌握客户的信息。对于服务器来说,一个业务处理组件就代表了相应的单个客户。当客户终止与业务处理组件互操作的时候,会话终止,组件也不再拥有状态值,称之为有状态的业务处理组件;而一个无状态的业务处理组件并不掌握客户的信息或者状态。客户能调用组件来完成一些操作,但是组件只是在方法调用的时候才知道客户的参数变量,当方法调用完成以后,组件并不继续保持这些参数变量,因此所有客户无状态的业务处理组件的实例都是相同的,这样,无状态的业务处理组件就能够支持多个客户。系统能够定义一个无状态的业务处理组件,并将任何业务处理组件指定给任何客户。

业务处理组件的生命周期相对较短,只有当客户保持会话的时候,它才是激活的,一旦客户退出,它就不再与客户相联系了;因此业务处理组件被看成是瞬时的。另外,业务处理组件也能用于事务处理,它能够更新共享的数据,但它不直接描绘这些共享的数据。

3.2 数据组件的开发

数据组件(Entity Beans)主要和数据库打交道,对数据进行封装和链接,对数据库中的数据提供对象的视图。例如,一个数据组件能够模拟数据库表中一行相关的数据。数据组件是由主键(Primary key,唯一的对象标识符)标识的。通常,主键与标识数据库中的一块数据(例如一个表中的一行)的主键是相同的,主键使客户能够定位特定的数据块。多个客户能够共享同一个数据组件,多个客户也能够同时访问同一个数据组件。数据组件通过上下文来访问或更新下层的数据,这样,保证了数据的完整性。

数据组件能存活相对较长的时间,并且状态是持续的,只要数据库中的数据存在,数据组件就一直存活,即使 EJB 支持平台崩溃了,它也是存活的。数据组件的生命周期能够被支持平台或者组件自己管理,如果由支持平台控制数据组件的持续发布,则不需要编写代码;如果由组件自己管理,就必须写代码,包括访问数据库的程序代码。

4 结束语

EJB 技术定义了一组可重用的组件,利用这些组件,可以像搭积木一样的建立分布式应用程序。当把代码写好之后,这些组件就被组合到特定的文件中去。每个文件可有 1 个或多个服务器端组件,再加上一些配置参数,就可以将这些服务器端组件配置到 EJB 支持平台的运行环境中,客户通过这些组件的接口,定位到某个组件上,由此产生组件的一个实例。这样,客户就能够调用组件的应用方法和远程接口。EJB 技术提供了所有典型的中间层服务,如事务管理、安全管理、远程客户连接、生存周期管理和数据库连接缓冲等,它规范了跨平台的、分布式的和面向对象的组件模型,加快了服务器端组件模型的开发。

参 考 文 献

1 企业计算中的 JAVA. http://go.4.163.com/blueski/javaprg/javatech.htm, 2000

2 EJB-JAVA 服务器组件模型 http://sun.java.com, 2000