Sun公司核心技术丛书

EJB 2.0企业级应用程序开发

（美） Chuck Cavaness
Brian Keeton 著

智慧东方工作室 译

机械工业出版社
China Machine Press

EJB是用于服务器端组件体系结构的一个规范。本书主要内容包括：构建EJB类及其接口的机制，EJB设计和性能的策略，构建Web层，EJB的高级概念等。附录包括了EJB 2.0的摘要及其新特征。

本书内容翔实、深入浅出，提供了详细的讨论和实例，对于懂Java语言的EJB初学者是一本有益的指导书。本书所附光盘包括：WebLogic Server 6.1的试用版、WebGain's VisualCafe 4.5的试用版、WebGain's TopLink 3.5.1的试用版，以及本书中使用到的实例。

图书在版编目（CIP）数据

EJB 2.0企业级应用程序开发/（美）卡瓦内斯（Cavaness, C.），（美）奇顿（Keeton, B.）著；智慧东方工作室译. -北京：机械工业出版社，2002.3

（Sun公司核心技术丛书）

书名原文：Special Edition Using Enterprise JavaBeans 2.0

ISBN 7-111-09910-9

Ⅰ.E… Ⅱ.①卡… ②奇… ③智… Ⅲ.Java语言-程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字（2002）第011052号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：宋燕红 张鸿斌

北京昌平奔腾印刷厂印刷 · 新华书店北京发行所发行

2002年3月第1版第1次印刷

787mm×1092mm 1/16 · 31印张

印数：0 001-4 000册

定价：65.00元(附光盘)

前　言

欢迎使用Enterprise JavaBean(EJB) 2.0！作为Java 2 Enterprise Edition（J2EE）的一部分，EJB体系结构已经成为被接受的标准，可用于分布式的、关键任务的商业应用程序的开发。EJB规范将J2EE应用程序服务器转变成一个基础，用于构建安全的、事务的、可伸缩的以及可移植的应用程序。它的成功范例正在不断增多。如果你在开发大型商业系统并且从未使用过EJB体系结构，现在就请加入这个行列吧。如果你已经使用过EJB，EJB2.0规范提供给你更多的方法来提高生产率和组件可移植性。

什么是EJB？EJB是用于服务器端组件体系结构的一个规范。不要与常规的JavaBean混淆。EJB是行业强化组件，它封装了可重用的业务逻辑以及对外部资源的访问，比如一个企业的关系型数据库。EJB的首要目的是使得开发者专注于业务逻辑，而不必担心他们的应用程序所需的事务、安全性以及持久性等底层细节。这些要求是按这样的一种方法处理的，该方法使你能创建跨应用程序服务器的可移植组件——这样也就达到了该体系结构的另一个目的。无论怎样，一个组件的价值通常是以它的重用性来衡量的。EJB采用一个说明方法来部署应用程序，它支持组件的扩展自定义，而又不需要修改代码。

EJB规范首先由Sun Microsystems这一家公司引入，但通过Java公用进程（Java Community Process，JCP），它的当前形式现在已经成熟了。得益于领先的应用程序服务器和软件厂商的参与，EJB 2.0在体系结构上又继续取得成功。

本书面向的读者

本书主要面向的是懂Java语言但对EJB陌生的编程人员。EJB是一个复杂的主题，但是本书主要介绍基础技术，为你深入研究EJB提供了一个起点。当你有了基础时，在进入更高级的主题前我们将引入更详细的讨论和实例，使你能掌握作为一个EJB开发者所需的技能。

对于在EJB方面没有经验的编程人员，这本书应该很适合。如果你已经知道了EJB，就可能认识到EJB 2.0规范在体系结构的几个关键领域引入了重大变化（如果想了解一个概要，请查看附录B）。对于一些熟悉EJB的读者，本书将教授如何利用新的内容。它也将为你提供几个已被证实可行的EJB设计实践，它们正在行业范围内逐步成型。如果你已经使用过EJB 1.1并正考虑转向EJB 2.0规范，那么本书为你指出了一些在改变体系结构之前必须考虑的重要问题。如果你已经决定现在继续使用EJB 1.1，只希望它能赶上2.0规范的速度，本书也能提供指导。

本书是如何组织的

本书共分五部分，介绍EJB的概念以及设计和实现自己企业bean的进程。继续阅读本书将看到本书还组织了相关的主题来建立你的EJB编程知识。

第一部分讲授了构建EJB类及其接口的机制。该部分开始是一个基于组件开发的概述，并用

详细的实例介绍了每个企业bean的类型，它告诉你在开始构建和部署你自己的EJB时需要知道的内容。为了确保你理解EJB直接依靠的J2EE其他技术，这些章节分别提供Java命名和目录接口（JNDI）和Java消息服务（JMS）更深的内容。这一部分还定义了在EJB应用程序中事务的作用，并介绍了异常处理和安全性管理。

第二部分超出了构建EJB的基础，介绍了一组设计和性能的策略，可应用于你的企业级开发计划。尽管EJB是一种相对较新的技术，但已经出现了一些标准的实践，作为一个设计者，必须意识到这一点。这部分还包括了在对你的应用程序进行负载测试时可以采用的几种方法的一个讨论。

第三部分超越了你在一本EJB书中所期望看到的内容。这里包括的章节有几种模式，你可以将它们应用在构建基于Servlet和JSP的表示层，它可以与使用EJB的应用层进行交互。

第四部分使你理解经验丰富的EJB开发者所关注的问题。这里你将得到一个用EJB 2.0引入的新要求的有关信息。它使得EJB能够与在另一个厂商的容器里运行的CORBA对象或EJB进行通信。另外，对于你的应用程序，还介绍了一些用于构建一个基础服务层的推荐实践。对于在第二部分中的构建性能的有关章节，你还将学习有关EJB组件和服务的集群内容。如果你是这种类型的人——当别人告诉你不能做某事的时候你总想知道为什么，那么你就会对第四部分的最后一章特别感兴趣。该章特别涉及了你在EJB容器内不能假定怎么做的事及其原因。

第五部分提供了一些快速参考材料，包括了EJB 2.0的摘要以及自EJB 1.1的变化的一个说明。

在本书中使用的约定

本书在格式上和印刷上使用以下特殊段落，更便于读者阅读。

注意 当你在本书中看到一个注意，它指明了附加的信息，它能帮助你更好地理解一个主题或避免与此主题相近的问题出现。

提示 介绍了经验丰富的开发者应用的技巧。可简化一个任务或产生一个更好的设计。提示的目的是帮助你应用标准的实践，来完成健壮的（robust）、可维护的应用程序。

警告 提醒你注意危险的过程（例如，潜在地危及一个系统安全的操作）。

参见 提供相关内容的介绍。

本书的许多章包括了一个“疑难解答”小节，关于某个主题你可能会遇到一些常见问题，该小节提供了相应的解决方案。

目　录

第三部分 构建Web层

第四部分 高级概念

第二部分 设计与性能

第五部分 附 录

第一部分 开发EJB

第1章 企业级应用程序导论

1.1 EJB体系结构

EJB 2.0规范将Enterprise JavaBean（EJB）定义为一个基于组件的分布式计算体系结构。如果你已经熟悉诸如基于组件和分布式计算的术语，这个定义会很有帮助，但是如果你是开发EJB或企业级应用程序的新手，就不太容易理解。把EJB定义一个企业级应用程序可能更好理解。在我们定义之前，先看一个EJB体系结构的示意图。这样，当我们进行讨论的时候，你的头脑中将有一个形象的认识。图1-1从一个高级别上示例了EJB体系结构。

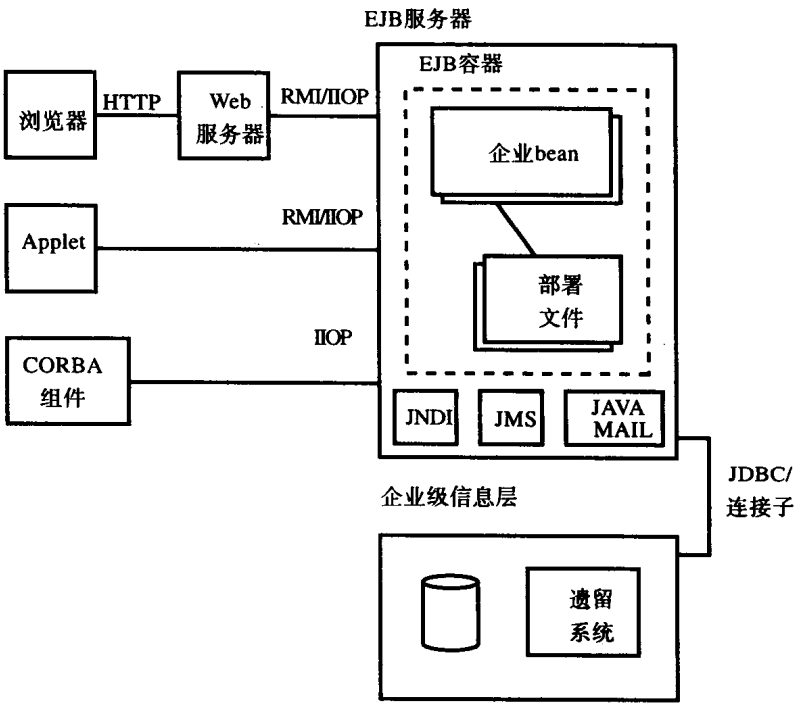

图1-1 EJB体系结构的高级视图

如果问100个软件开发者或结构设计师“什么是企业级应用程序”，可能会得到101种不同的回答。这并不是因为软件开发者的概念不清楚，而是企业级应用程序的定义确实有些模糊。每个开发组织或商业组织可能都有他们自己的企业应用程序定义。但是，如果我们能得到一个单一的大多数软件开发者都赞同的定义，可能更好一些。

企业级开发并不是EJB或相关的Java出现后才开始的。实际上，它已经存在了很多年，并且在大型机开发流行时就是一个经常使用到的术语。然而，对于你们当中的一些人来说它可能是一个新概念，因为他们涉及的应用程序仅执行一个相对较小的商业任务，并且通常被限制在一个单一的地址空间内。

由于Internet的飞速发展和信息技术（IT）部门的分散化使得各公司从20世纪90年代早期就使用了CORBA，尽管这样，越来越多的跨越网络边界的应用程序仍在被开发并且涵盖了更多的商业日常功能。同时，这些由企业现有其他组件或应用程序产生的应用程序正被传播得越来越远。对于“企业级应用程序”，我们指的是一个组织已经构建、购买并要求为另一个组织提供服务的所有应用程序或服务。这些服务可能处理的是一个组织在常规基础上必须管理的存货管理、标明价位或其他的一些事务。企业应用程序不再是过去那种被四面墙包围着的数据中心了。现在，组件和服务分布于所有组织。

开发组织必须处理这些新技术完全可能带来的挑战和复杂性：去正确驾驭分布性 。因为服务是在网络上传播的，这些新技术挑战在设计和开发时必须被考虑到。一个组织的数据或商业进程固定在单一的物理位置上早已不现实了。大多数公司现在必须全球化，并且不能再假定客户仅在相同的地理位置内。因此，应用程序就必须具有灵活性和可扩展性以满足无论何时何地遍及全球的客户和其他合伙人的要求。有了这些，如果运气好的话，新客户和合伙人每天都将增加，而这在应用程序的可伸缩性和性能上会产生极大的影响。

正如你开始看到的那样，企业级开发者必须处理许多在小应用程序中并不会出现的复杂技术问题。企业应用程序必须支持地理上分散的多个网站，处理客户和合伙人全天每时每刻访问一个应用程序、支持多种语言和多用户同时访问，并且考虑了这种分隔很远的连接所出现的复杂问题。其他的问题，比如用企业中现有应用程序接口，也是很常见的并且必须被支持。企业内的物理硬件和应用程序完全异构这也可能是事实。例如，有些企业应用程序需要在UNIX操作系统上运行，而另外一些则只在Windows平台上运行。这种不同设置所带来的制约只会增加企业开发者工作的复杂性。

回到由规范提供的EJB定义上来，考虑企业应用程序开发的另一个定义，我们试着提出自己的EJB定义。我们需要一个没有混淆的、可以直观地掌握理解并能够与他人交流的定义。所以这里试着提出一个定义，应该稍加考虑就能理解：“Enterprise JavaBean是Java开发者编写的并安装在一个应用程序服务器中的Java组件，它提供了命名、安全性、事务性以及其他企业级服务。这些被安装的组件能按一种分布式形式通过网络来应用。”

尽管上面的定义包含了一些对你来说可能是新的术语，但它应该使你更近一步地理解EJB体系结构如何帮助你来构建企业级应用程序。

注意 *在定义中提到的应用程序服务器一般是由第三方构建的，并安装在您的环境中。*

当使用EJB构建企业级应用程序时，有一些其他特征需要理解。这包括诸如可伸缩性、多用户、负载平衡、容错度等许多方面。问题是需要学习的内容太多，这就要求你按正确的步骤并在适当的时候有一个更宽松的时间来掌握所有这些概念。本书的主要目的是：逐步介绍这些概念，在能够理解时才引入它们。学习Enterprise JavaBean有时看起来可能过于庞大，因为有太多相关的概念和技术。当学完这本书之后，你将非常有希望获取来自使用EJB及其相关技术的最重要的

收益之一，即为你提供大量的基础结构。

1.2 基于组件的分布式计算

什么是基于组件的分布式计算？为什么它是如此的重要？这是一个很好的问题，在深入讨论之前必须回答它。首先要回答的问题是“什么是组件以及组件为企业级应用程序开发者提供了什么样的价值？”

1.2.1 软件组件的快速复习

当我们谈论组件时，我们指的是什么呢？我们可以说EJB是一个组件，也可以说Java类是一个组件。试图提出有关组件的一个单一定义，使得软件界的每个人都同意可能不太容易。近年来，全世界的软件开发者和结构设计师都试着描述什么是组件。在1996年，面向组件编程（ECOOP）的欧洲工作组提出了以下定义：

“软件组件是一个具有契约式特定接口和显式语境从属的组成单元。软件组件能被独立地部署并从属于第三方的组成。”

你能注意到在定义中没提及Java或EJB。这是因为组件的概念比Java的诞生要早。尽管Java支持组件的思想，但却不是它发明的。事实上在上面的组件定义中可以拿出三个很重要的特性：

- 特定接口。
- 显式从属。
- 部署能力。

组件通常为它的客户机提供一个或多个商业服务。客户机可能是一个GUI接口（基于WEB的或其他的），或者在多数情况下是另一个组件。组件能提供的服务，简单的可以像为一个客户返回电子邮件地址，复杂的可以像为运输到Berchtesgaden的定单进行运费计算。无论组件提供什么样的服务，它都是通过公共的特定接口来提供它们。这意味着与组件联系的客户机不能看到组件的内部，客户机对组件发出请求后仅得到一个结果。有时这也涉及到“封装”。

组件通常有操作（方法）、属性（状态）以及它产生的其他事件类型（可能是异步的通知）。一个好的组件，在如何维护它内部状态的细节上对客户机是隐藏的。这有助于组件与客户机去耦合。例如，如果组件如何计算运费的逻辑被改变了，客户机是不会关心的，只要它仍提供相同的接口并算出正确的数量就行。数量是由运输组件计算，还是由运输组件通过网络联系到一个遗留系统来得到数量，客户机都不会关心。

你可以把组件提供给客户机的公共接口看作是客户机和组件之间的一个合同。组件告诉客户机：“我将计算运输定单的数量，如果你给我提供正确的定单ID，我就能查看运输货物的重量。”组件显露给客户机的一组接口构成了客户机和组件之间的合同。有时这也涉及到“组件接口”。

来自定义的下一个很重要的概念是：一个组件可以从属于另一个组件，从而完成它的商业服务。这些从属性应该是显式的，并记录在文档中。在我们的实例中，Shipping组件从属于主机应用程序，它调用主机系统来计算装运定单总额。它就具有一个从属性。如果主机不能提供总额，Shipping组件只能通知客户，它这次不能完成该项定单，这也意味着商业收入的损失。在组件之间具有这些从属性，没有什么不合适或错误，只要它们能理解对方。在企业应用程序中让组件从

属于其他组件是很正常的。这些组件只要可理解并具有结合性即可。通过理解从属性，当公共接口中的一个需要改变或删除时，你能快速地确定哪些其他的组件将被影响到。

来自定义的最后一个概念是部署。这有点儿模糊，因为部署的定义没有给定，所以可能它对不同的人意味着不同的东西。Java类必须被部署。它们必须在正确的包中和系统的类路径中。CORBA类也要部署，但它是按一个完全不同的形式部署的。但是，一个组件的服务对客户机成为可用之前，它有一些类型的部署必须执行。对于一些技术或体系结构，谁负责部署，是有区别的。原始开发者通常是负责部署一个标准Java类的人，尽管有时可能负责的是一个较大的部署中的一部分。正像你在第3章的3.3节中看到的一样，EJB为应用程序所需的组件提供者、部署者以及其他必须角色定义了角色和责任。

参见　有关EJB角色及其责任的详情，请参见第3章的3.3节。

图1-2显示了一个组件的实例，它支持我们描述的所有三个特性。

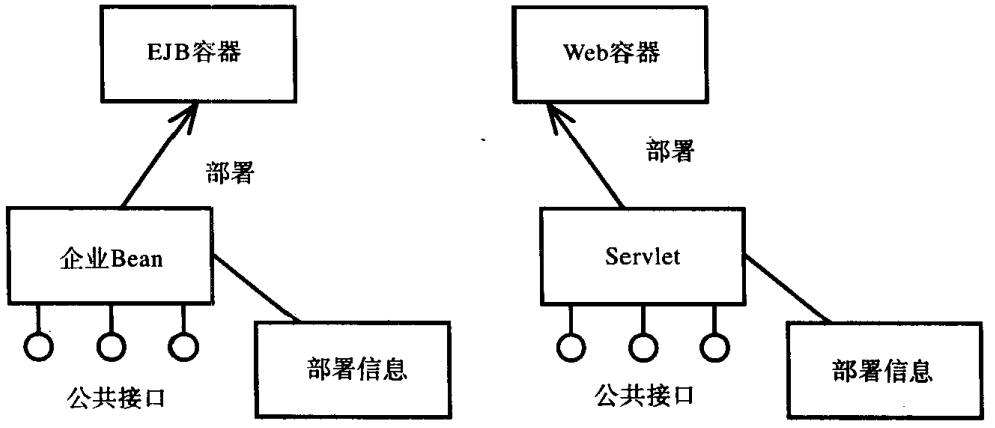

图1-2　一个具有公共接口和从属性的组件，并且能被部署

1.2.2　将组件集合到体系结构中

你创建单个组件后，下一步是将它们装配到一个更大的组件组中。这个更大的组就是所谓的组件体系结构。“组件体系结构”通常包括一组构建应用程序的组件和服务，并能利用一个或更多的framework。framework是其他组件的一个库，这些组件可以在多个应用程序上重用，并且通过提供经证明和测试的服务和功能，可以保存部署时间。你可能已经听说过“不必多此一举”这句话，这也是framework的目的。许多公司在商业上提供framework，但是通常它们是由构建应用程序的组织所构建的。

有许多不同样式和类别的体系结构。可能有系统体系结构、应用程序体系结构、网络体系结构、数据库体系结构以及列表体系结构。

如果在Internet上搜索“什么是体系结构？”，可以看到结果涉及的范围很广，从空间防御合同信息到一百篇介绍什么是体系结构的不同作者的论文。就我们的目的而言，“体系结构”就是一组相关的组件和framework，有助于描述组件之间存在的从属，以及在应用程序的生命周期期间，对于事件它们应该如何反应。在某种意义上，这是EJB规范要说明的内容。

1.3 N层体系结构

体系结构的另一种类别是体系结构所具有的不同的层数。“层”是软件与可能的硬件、组件、服务的一个组合。该组合可以是逻辑的和物理的。使用层的目的是使软件组件和服务在多个电脑之间分布，以此来获取可伸缩性和安全性。例如，正像你在第19章中看到的一样，Web服务器组件和服务有时分布在不同的层中，而不是在应用程序服务中。这就为应用程序增加了安全性，因为更多的应用程序可以更深地定位在一个受保护的网络中。你将会听到三个最常用的体系结构：

- 两层。
- 三层。
- 普遍存在的N层。

注意 这里引用的N层是普遍存在的，因为这个术语用于描述多种不同类型的企业级应用程序的体系结构。

N是指体系结构具有多少层，从1到某个数字（N）。在大多数情况下，N通常是3、4或者更多。通常，开发者用术语“N层”来指示一个三层体系结构。我们快速地浏览一些常见的体系结构，以及它们目前是如何使用的。

1.3.1 两层组件体系结构

如果技术性书籍没有提及两层客户机/服务器体系结构，那么它就不值得一看。常言道“忘记过去错误的人注定要再次犯错”，两层客户机/服务器应用程序编程模型在20世纪90年代早期就很大。实际上，对于某些类型的应用程序，它仍是一个非常流行的体系结构。问题是当许多用户在同一时间使用该系统时，它不进行衡量。我敢肯定读者中的许多人这时候立刻大叫，因为你发现了一种方法，可以让它进行衡量，但是一般地说，当并发用户的数量开始攀升时，它就不能衡量得很好。一般地，这是由于数据库不能处理大量的客户机连接。

两层体系结构也有其他问题。为了发布一个新版本的软件，所有客户机必须用新客户机软件更新，它包含了所有商业逻辑、数据库逻辑以及其他内容。这也意味着非常复杂的代码，比如安全性和数据库交互，它是在每个客户机上完成的。这就增加了客户机应用程序的开销，也增加了网络通信。用Java小程序，发布进程会变得很容易，但是小程序仍然包含一些逻辑，它们能更好的定位在其他地方。可以用一些安全性限制，将它们放置在小程序上，但该项技术仍不能适用于所有的应用程序。

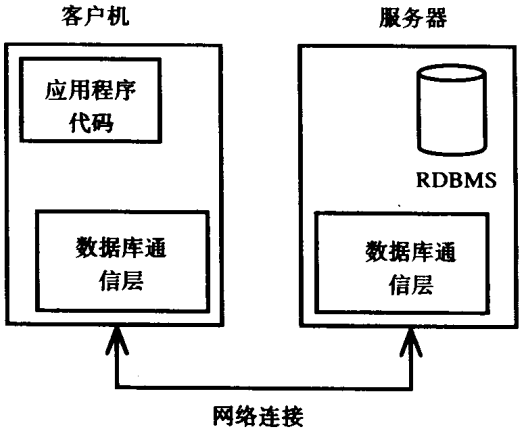

图1-3 一个常见的客户机/服务器体系结构

该体系结构通常在一个负面的形式中作为“胖客户机”被引用，因为它在层的内部虚拟地包含了所有组件。第二个层通常是某些类型的数据库，最常用的是关系型数据库。图1-3示例了一个常见的两层客户机/服务器体系结构。

在80年代后期和90年代早期，许多开发组织开始用三层模型替代两层模型。开发者和结构设计师基本上所做的是，在前面两个层之间添加一个新层，一些软件组件被安装或部署在这个层中，而不是在客户层。大部分被移到这个新层上的组件和服务是那些负责应用程序的商业逻辑的组件和服务。

1.3.2　N层组件体系结构

这个新中间层能安装在网络中的某处，并且它能被许多不同的客户机共享，而不是属于某个客户机。这也很有帮助，因为这样客户机应用程序的大小和复杂性都被减小。这是很好的，因为与常见的中间层服务器相比，客户机通常在处理能力上要更弱些，不能完全处理应用程序。

有了所有这些益处，当然应该对N层体系结构也会有负面的影响。答案是肯定的，是有一些负面影响。负面影响与应用程序的分布式特性相关联。当中间层被创建时，就添加了复杂性，因为应用程序现在需要处理许多事，比如安全性、并发访问、多线程问题、客户机如何定位中间层以及客户机/服务器模型不必处理的其他事情。当然，二层模型也必须处理诸如安全性这样的事，但是因为每个客户机是自主式的，所以这些事就更容易处理。必须为多层体系结构处理的最明显的复杂性之一是第一层与中间层之间必要的网络计算。图1-4示例了一个常见的多层体系结构。

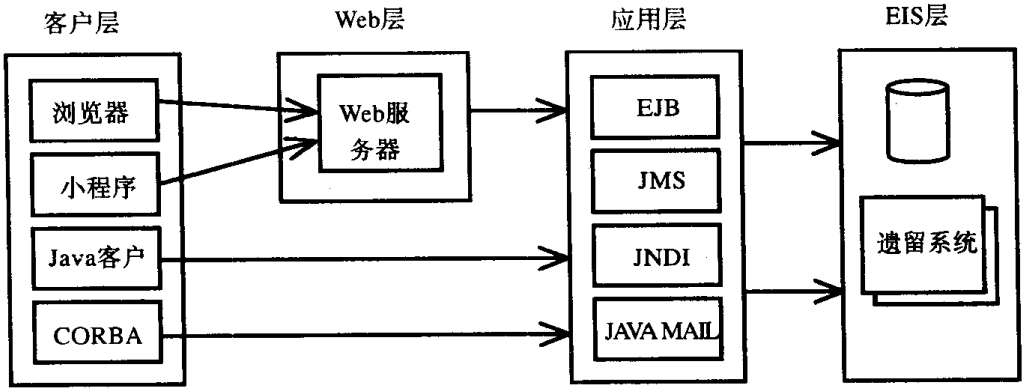

图1-4　一个多层体系结构的实例

你必须问一下自己，是否使用一个分布式组件体系结构是真的那么重要吗？通过学习这些技术，你真的从投资上获得了回报了吗？简而言之，答案是完全肯定的。我们将一个系统分成几部分，每一部分具有单一的或很小的一组责任，本书余下内容展示了每一部分所能侧重的工作，以及它是如何设计完成的。这也使开发者或厂商能够最优化一个组件，使它完全地实现最好的一面。其他合理性也是很有益的，比如具有瘦客户机，能够换出UI而不必担心商业逻辑出口。更不用说，由于具有一个物理上的分布式应用程序，你能更好地利用群集技术。我们将在第22章中讨论群集概念。

通过分隔层，许多组织能够购买第三方的复杂软件组件，然后将其安装到他们的体系结构中，这样他们就可以将更多的时间用于解决核心商业问题。

1.4　为什么使用EJB

到现在为止，你应该理解了组件的重要性和价值，但是我们没有规定一个情况，仅使用EJB，