EJB

企业应用与开发实例

班书昊　编著

北京科海电子出版社

EJB 企业应用与开发实例

班书昊　编著

北京科海电子出版社
http://www.KHP.com.cn

内 容 提 要

本书详细介绍了J2EE体系结构的重要组成部分——EJB及其相关技术。全书共分14章，分别阐述EJB的概念、方法和部署。内容涵盖了J2EE和EJB体系结构、EJB的简单开发、会话Bean（分有状态和无状态）、实体Bean、消息驱动Bean、应用程序的运行环境——容器、EJB的总体考察、资源管理与EJB服务、EJB应用程序、CORBA与EJB集成技术、J2EE应用程序、EJB生成Web服务和无线EJB服务。针对书中的知识点提供了大量的实例，并详细地讲解了EJB开发的方法和技巧，尤其是“实例目标”、“学习要点”和“实例分析”起到了提纲挈领、画龙点睛的作用。

本书适用于所有对EJB技术感兴趣的读者，特别是希望从事企业应用程序的开发人员和部署人员。

盘　　名：EJB企业应用与开发实例
作　　者：班书昊
责任编辑：陈 轶
排　　版：卞雨桂
光盘制作：马首鳌
咨询电话：（010）82896445-8406

出　　品：北京科海电子出版社
印 刷 者：北京科普瑞印刷有限责任公司
开　　本：787×1092 1/16　　印张：28.75　字数：700千字
版　　次：2003年3月第1版　　2003年3月第1次印刷
印　　数：0001～4000
盘　　号：ISBN 7-900107-43-6
定　　价：45.00元（1CD/配套手册）

前　言

什么是EJB?

EJB就是在Java 2平台企业版（J2EE）体系结构中的Enterprise JavaBeans部分，是生成业务逻辑应用程序的主要分布式组件模型。从程序员的角度来说，EJB就是一组Java类和一个XML文件相结合的产物。其中，Java类符合定义规则并能提供特定回调方法。在开发过程中涉及到的对象有客户、容器、主接口、组件接口和Bean实例。其中，容器负责管理系统级问题；Bean实例是业务逻辑方法的实现方法。

为什么学习EJB?

EJB实现了业务逻辑和系统级服务相分开的特性，同时提供了开发大型、安全、可扩展和具有事务性应用程序的重要技术。目前，J2EE和EJB可以说是最成熟、最健全的企业开发模型。

本书的读者对象

本书的阅读对象是所有对EJB技术感兴趣的读者，特别是希望能够从事企业应用程序的开发和部署的开发人员。

本书假定读者已经具备了Java语言的基础知识。如果您还没有具备，则可以通过学习任何有关Java语言的图书掌握以下知识即可：

- Java语言的基本数据类型及其运算。
- Java语言基本控制语句。
- Java类和对象的概念。

本书的特点

本书的目标就是让读者尽可能快的掌握运用EJB技术开发企业应用程序，因此知识点的安排是从易到难，即从简单开发到资源、事务和安全管理。每个知识点都有相应的实例，每个实例都有实例目标、学习要点（编程思想）、可编译的代码和实例分析。代码中有大量的注释，目的是为了使读者能够更好地学习该实例。本书的实例代码完全可以全部编译（个别代码是为了学习需要，文中已经标为样本代码），并且都放进了配套光盘中，光盘中也提供了运用EJB技术开发企业应用程序所需要的工具。

光盘说明

光盘中包括书中实例的全部源代码和许多资源文件。

一般情况下，将本光盘放入光驱中后，就会自动运行并打开光盘内容索引网页文件。如果光盘没有自动运行，可以通过双击光盘根目录下的Start.exe文件或Index.htm文件来启动索引网页文件。然后读者就可以通过超链接来打开光盘中的实例源代码或共享软件。

由于作者水平所限，书中可能存在错误和不足之处，敬请读者批评指正，E-mail地址：shuhao_stanben@yahoo.com.cn。

作者

2003年3月

目　录

第 1 章　J2EE 和 EJB 体系

本章重点

在本章中我们将要介绍 J2EE（Java 2 Platform, Enterprise Edition, Java 2 平台企业版）以及 J2EE 中的 EJB。1.1 节简要概述 Java 家族中的 J2EE 和 EJB，并且提到了分布式计算系统的变革。本章的重点是 1.2 节，该节包括 J2EE 体系结构，J2EE 的企业应用和 J2EE 中的各种角色。

1.1　Java体系简介

Java语言在其诞生的短短几年里，就在计算机行业中得到广泛应用，并且日益发挥着重要的作用。事实上，整个Java计算体系是由客户机和服务器（client/server）两大部分组成的，服务器在整个Java计算体系中承担着重要的责任。

Java由于其良好的跨平台性而成为服务器端的理想语言。为了利用Java实现服务器端的计算，Sun公司推出了一个完整的开发平台J2EE，其目的是为基于Java的服务器端配置提供一个多用户企业级的安全平台，而J2EE的基石就是Enterprise JavaBeans（EJB）。EJB是建立基于Java的服务器端组件的标准，它定义了如何编写服务器端组件，提供了组件与管理组件的应用服务器之间的标准约定。EJB是一种组件架构，使得开发人员能够快速开发出可扩展的企业级应用程序。

1.1.1　分布式计算系统

1. 分布式计算系统的变革

在个人计算机出现之前，计算系统通常是由连接大型机的哑终端组成的集中式系统。个人计算机的出现并没有改变计算模型，只是将处理功能转移到用户桌面上。而且，我们把这种在单个处理空间中运行的应用程序称为单一体，对应于单一体的系统称为单层系统。单层系统的缺点是，程序中任何部分的改变都导致要重新编译全部代码。

20世纪80年代后期，出现了新的计算模型——客户机/服务器（client/server）——两层系统。由于PC机至少把一些处理功能转移到了用户桌面，因此，PC机既可以作为大型机的哑终端，也可以用来运行小型的应用程序，还可以输入数据和进行最基本的验证，然后再将结果直接传送到大型机上运行的应用程序中。

在两层系统中，业务逻辑的任何改变仍然要求对应用程序的大部分代码进行改变。而

且，由于数据库和业务逻辑嵌入到服务器上的应用程序中，如果应用程序从一个数据库移植到另一个数据库，就要修改服务器代码。

为了简化代码的修改，应该把代码进行模块化，因此需要三层模型，使业务逻辑和数据库服务器分开，这样对业务逻辑与数据库服务器的任何改变都不会影响到对方。于是应用程序分为了如下三层：

- 表示层（Presentation）——提供用户界面。
- 业务逻辑层（Business logic）——实现业务逻辑。
- 数据层（Data）——负责业务层中所有数据的持久存储。

三层模型的引入使业务逻辑具有更好的伸缩性。把业务逻辑放在中间层，使之成为整个应用程序中处理量最大的模块，这就是所谓的分布式处理。

20世纪90年代初期，Internet的普及，使各大公司的应用程序都支持Web技术，许多公司人员可以在出差时或在其他地方运行相同的中央应用程序。于是，三层体系结构进一步模块化和分布化，这就是n层体系结构。n层体系结构中分布式技术主要在于使用专用的Web处理空间来处理业务逻辑，客户机只是哑Web浏览器。

2. 分布式应用程序

三层体系结构使分布式应用程序的开发变得更加复杂。其实，开发分布式应用程序的目标就是让每个应用程序模块不知道而且不必知道本地或者远程与之交互的其他模块。

任何较大规模的应用程序都要求在简单应用程序逻辑之外增加其他服务。要利用这类服务，通常需要建立和配置不同的中间件解决方案。因此，应用程序的开发、管理和维护是复杂、费时和昂贵的。

服务器方要求建立各种应用程序共用的、具有即时解决方案的平台，使基础结构层与更具体的工作相分离，J2EE就能达到这种要求。

1.1.2 J2EE简介

为了方便开发n层体系结构的应用程序，Sun公司开发了基于Java平台的新型企业体系结构，这种新的体系结构称为J2EE，J2EE采用了基本的Java概念，提供了独立的高层API。在企业环境中，J2EE还定义了新型的分布式应用程序体系结构，解决了n层应用程序开发中的许多问题。

1.1.3 EJB简介

EJB（Enterprise JavaBeans）是Sun公司在服务器平台上推出的Java技术家族的成员。从软件构件的角度看，EJB是Java技术中服务器端软件构件的技术规范。我们知道，在软件产业中，基于构件的技术是当前的热点，在面向对象技术发展的今天，构件作为可重用的软件组件，在软件系统的开发上，解决了重复开发的问题，提高了软件开发的效率。

从企业应用多层结构的角度看，EJB是业务逻辑层的构件技术，与JavaBeans不同，它提供了事务处理的能力。自三层结构出现，中间层（也就是业务逻辑层）就成了处理事务

的核心，由于从数据层分离，它基本上取代了存储进程。

从分布式计算的角度看，EJB既提供了分布式技术的基础，还提供了对象之间的通信手段。

从Internet技术应用的角度看，EJB和Servlet、JSP一起成为新一代应用服务器的技术标准。EJB中的Bean可以分为会话Bean和实体Bean，前者维护会话，后者处理事务。现在，Servlet负责与客户端通信，访问EJB，并把结果通过JSP产生页面传回客户端，这已成为开发的新潮流。

从发展角度看，EJB完全有可能成为面向对象数据库的新平台，构成企业计算的基础。

1.2 J2EE体系

1.2.1 J2EE体系结构

下面主要从J2EE的应用编程模型、J2EE平台以及J2EE服务器三个方面来介绍J2EE的体系结构。

J2EE体系结构提供中间层集成框架，用来满足没有太多费用而又需要高可用性、高可靠性以及可扩展性的应用需求。通过提供统一的开发平台，J2EE降低了开发多层应用程序的费用和复杂性，同时对现有应用程序的集成提供强有力的支持。它完全支持EJB，有良好的支持向导、打包和部署应用，还添加了目录支持，增强了安全机制，提高了性能。J2EE是一种利用Java语言的标准体系结构。在企业应用开发中利用这种体系结构，开发者不必担心运行业务应用程序所需的“管道工程”，从而可以集中精力重视业务逻辑的设计和应用程序的表示。J2EE巩固了Java 2标准版中的特征，在巧妙处理Java的性能和安全问题的同时，增强了可伸缩性。

1. J2EE 的应用编程模型

J2EE的应用编程模型提供一种体系模型，包含用于实施基于J2EE的多层应用的文档和实例套件。J2EE的应用编程模型是开发人员设计和优化组件的原则。J2EE应用编程模型要求开发者将自己的工作分成两类：业务逻辑和表示逻辑。由于不必为中间层进行编码，开发人员就能将更多的时间用在业务逻辑和表示逻辑上，对重视缩短项目周期的公司来说，这种模型深受欢迎。

2. J2EE 平台

J2EE平台是运行J2EE应用程序的标准环境，它由J2EE部署规范（一套所有J2EE平台产品都必须支持的标准）、IETF标准集和CORBA标准组成。最新的J2EE平台还添加了JavaBeans组件模型。开发人员可以利用JavaBeans组件模型来自定义Java类的实例，并可通过已定义的事件访问Java类。

3. J2EE 服务器的功能

J2EE服务器通过Java命名和目录接口（JNDI）、认证以及HTTP与EJB的兼容能力，提供命名和目录服务。J2EE服务器还利用了Java Servlet技术。Servlet（可以看作是运行在服务器上的一个小程序）是一种扩展Web服务器功能的技术，由于它是用Java编写的，因而能够访问整个Java API库，包括用于访问企业数据库的JDBC API。

1.2.2 J2EE企业应用

1. J2EE 架构概述

J2EE架构（J2EE Architecture）包括：

- 可重用应用组件（Reusable Application Component）；
- 设计用户界面和引擎（Designing the User Interface and Engine ）；
- 设计基于Web的应用（Designing Web-Based Application）；
- Servlet和JSP页面；
- MVC设计范式（Model，View，Controller Design Pattern）；

（1）可重用应用组件（Reusable Application Component）

J2EE组件（applet、客户程序、Enterprise Bean、JSP页面及Servlet）都被打包成模块，并以JAR（Java Archive）文件的形式交付。一个模块由相关的组件、相关的文件及描述如何配置组件的配置描述文件组成。使用模块使得利用某些组件来组装不同的J2EE应用程序成为可能。例如，一个J2EE应用程序的Web版可能有一个Enterprise Bean组件，还有一个JSP页面组件。该Enterprise Bean组件可以与一个应用程序组件结合，以生成该应用程序的非Web版本。这不需要进行额外的编码，只是一个装配和部署的问题。并且，可重用的组件使得将应用程序的开发和部署过程划分成由不同的角色来完成成为可能，这样，不同的人或者公司就能完成封装和部署过程的不同部分。

（2）设计用户界面和引擎（Designing the User Interface and Engine）

在为J2EE应用程序设计用户界面和后端引擎时，我们需要决定是否让该程序基于Web。在做出这个决定时，我们应考虑平台配置、下载速度、安全、网络流量和网络服务。还应考虑繁重的处理应当在哪里执行。例如，如果客户程序在一个蜂窝电话或者呼机中执行，服务器应当完成尽量多的计算和数据处理，而客户程序只显示结果就可以了。然而，在一个功能强大的台式机上运行的大型财务分析系统则应当在客户机上完成其复杂计算。客户程序和applet用户界面通常都是用Swing API创建的。Swing API提供了一整套GUI（Graphical User Interface，图形用户界面）组件（表格、树形结构、按钮等），这些组件可以用来实现更好的交互。Swing也支持HTML文本组件，这个组件可以用来显示来自服务器的响应。客户程序可以直接访问Enterprise Bean层或企业信息系统层。

（3）设计基于Web的应用程序（Designing Web-Based Application）

基于Web的应用程序是基于浏览器的，并且，如果它们运行在Internet上的话，可能被

全世界的人访问。当设计一个基于Web的应用程序时，不仅需要决定用什么来处理内容和应用逻辑（HTML、XML、JSP页面及Servlet），而且还应当考虑该应用程序如何国际化。一个国际化的基于Web的应用程序应允许用户选择一种语言，然后，根据该语言加载应用程序正文。对所支持的每种语言，应用程序正文都被存储在一个外部文件中，并与另外一个文件的关键词相对应。应用程序代码使用这些关键词以及选定的语言来加载正确的文本。一个基于Web的应用程序使用HTML来显示数据；使用XML来定义数据以便被另一个程序读取并处理；使用JSP页面或Servlet来管理用户与业务层或数据层之间的数据流。

可以在J2EE平台上实现的基于Web的应用程序有四种。从简单到复杂排列，它们是：基本HTML；基本JSP页面；Servlet的HTML；基于JavaBeans类的JSP页面。

（4）Servlet和JSP页面

Servlet是实现动态内容的一种简便方式。JSP页面提供了Servlet的所有优点，并且，当与一个JavaBeans类结合在一起时，提供了一种使内容和显示逻辑分开的简单方式。分开内容和显示逻辑的优点是，更新页面外观的人员不必懂得Java代码，而更新JavaBeans类的人员也不必是设计网页的行家。在选择使用Servlet或JSP页面时，要记住的是，Servlet是一个程序设计工具，它最适用于不需要频繁修改的低级应用功能，而JSP页面则通过以显示为中心的描述性方法将动态内容和显示逻辑结合在一起。

（5）MVC设计范式（Model, View, Controller Design Pattern）

在J2EE平台充分内置了灵活性的情况下，剩下的问题可能是如何组织应用程序以实现应用程序的升级和维护，以及如何让不懂程序代码的人员避开程序设计。答案就是使用模型、视图和控制器（MVC）。MVC是一个描述重现问题及其解决方案的设计范式，但每次问题重现时，解决方案并非完全相同。

MVC设计范式包括三种对象：模型、视图以及控制器。模型（model）提供应用业务逻辑（Enterprise Bean类），视图（view）则是其在屏幕上的显示（HTML页面、JSP页面、Swing GUI），控制器则是Servlet、JavaBeans或Session Bean类，它用于管理用户与视图之间的交互。我们可以将控制器想像成处在视图和模型之间，它对视图如何与模型交互进行管理。通过将控制器和模型代码保持在视图之外，那些不理解这些代码的人员就不能修改代码。将控制器和模型分开可以在不影响模型的情况下改变控制器，也可以在不影响控制器的情况下改变模型。

2. 基于 J2EE 的多层模型

电子商务和信息技术的快速发展给应用程序开发人员带来了新的压力。为了降低成本，并加快企业应用程序的设计和开发，J2EE平台提供了一种基于组件的方法，来设计、开发、装配及部署企业应用程序。J2EE平台提供了多层的分布式应用模型、组件重用、一致化的安全模型以及灵活的事务控制。有了J2EE，您不仅可以以更快的速度向市场推出创造性的客户解决方案，而且，这种解决方案不会被束缚在任何一个产品的API上。

- applet：它意味着应用逻辑根据其功能而被划分成应用组件，并且可以在同一个服务器或不同的服务器上安装。一个应用组件应安装在什么地方，取决于该应用组件属于J2EE环境中的哪一层。这些层是客户层、Web层、业务层及企业信息系统层

（EIS）。

- 客户层：客户层（Client Tier）的J2EE 应用可以是基于Web的，也可以是不基于Web的。在一个基于Web的J2EE应用中，用户的浏览器在客户层中运行。在一个不基于Web的J2EE应用程序中，一个独立客户程序applet，不是运行在一个HTML页面中而是运行在其他一些基于网络系统的设备（比如手持设备或汽车电话）中，并在不经过Web层的情况下访问Enterprise Bean。不基于Web的该客户层可能也包括一个JavaBeans类来管理用户输入，并将该输入发送到在企业层中运行的Enterprise Bean类中来处理。根据J2EE规范，JavaBeans类不被视为组件。为J2EE平台编写的JavaBeans类有实例变量和用于访问实例变量中数据的“get”和“set”方法。使用上述方法的JavaBeans类在设计和实现上通常都是简单的，但是它们必须符合JavaBeans规范中列出的命名和设计约定。
- Web层：J2EE的Web组件可以由JSP页面、基于Web的applet以及显示HTML页面的Servlet组成。就像客户层一样，Web层可能包括一个JavaBeans类来管理用户输入，并将输入发送到在业务层中运行的Enterprise Bean类中来处理。运行在客户层的Web组件依赖容器来支持诸如客户请求、响应和Enterprise Bean查询等功能。
- 业务层：作为解决或者满足某个特定业务领域（比如银行、零售业或金融业）需要的业务逻辑代码由运行在业务层的Enterprise Bean来执行。一个Enterprise Bean从客户程序处接收数据，对数据进行处理（如果需要），再将数据发送到企业信息系统层存储。一个Enterprise Bean还从存储中检索数据，并将数据送回客户程序。运行在业务层的Enterprise Bean，依赖于容器来为诸如事务、生命期、状态管理、多线程及资源存储池等提供复杂的系统级代码。业务层经常被称作EJB层。业务层和Web层一起构成了三层J2EE应用的中间层，而其他两层是客户层和企业信息系统层。
- 企业信息系统层：企业信息系统层运行企业信息系统软件。该层包括企业基础设施系统，例如企业资源计划（ERP）、大型机事务处理（Mainframe Transaction Processing）、数据库系统及其他遗留信息系统（Legacy Information System）。一个J2EE应用程序的组件是单独运行的，并且往往在不同的设备上运行，因此，需要一种方法能让客户层和Web层的代码查询并引用其他层的代码和资源。客户层和Web层代码使用JNDI来查询用户定义的对象（例如Enterprise Bean）、环境条目（例如一个数据库驱动器的位置）、企业信息系统层中用于查找资源的JDBC TM DataSource对象以及消息连接。
- 安全：J2EE安全模型允许配置一个Web或Enterprise Bean组件，使系统资源只能由授权的用户访问。例如，一个Web组件可以被配置成提示输入用户名和密码。一个Enterprise Bean组件可以被配置成只让特定团体中的成员调用某些方法。或者，一个Servlet组件可以被配置成让某个组织中的所有人都能访问某些方法，同时只让该组织中的某些享有特权的人访问另一些方法。同样是该Servlet组件，可以针对另外一个环境而被配置成让每个人都能访问其所有方法，或者仅让选定的少数人访问其所有方法。
- 事务管理（Transaction Management）：J2EE事务模型使得能够在部署时定义构成单一事务的方法之间的关系，以使一个事务中的所有方法被处理成一个单元。这是