宋智军 米伟哲 武桂香 编著　邱仲潘 审校

EJB 3.1 从入门到精通

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

EJB 3.1 从入门到精通

宋智军　米伟哲　武桂香　编著
邱仲潘　审校

電子工業出版社
Publishing House of Electronics Industry
北京 • BEIJING

内 容 简 介

本书坚持以语言为基础、应用为主导的编写原则，理论联系实际并通过大量的实例，循序渐进地为读者介绍了有关 EJB 3.1 开发所涉及的各类知识。全书共分为 11 章，首先从最基础的 EJB 开发环境的搭建开始，再通过介绍 EJB 基础、会话 Bean、消息驱动 Bean、JPA、对象关系映射、事务等，最后通过相应章节的知识点进行实例的讲解。

全书的基础知识介绍清晰，理论联系实际，具有很强的操作性。本书还提供了大量的通过测试可运行的完整实例代码，这些实例都有相应的设计步骤、代码详解、程序运行结果等，通过实例不但可以复习前面所学的内容，而且还增加了一定的创作技巧。

图书在版编目（CIP）数据

EJB 3.1 从入门到精通 / 宋智军等编著. —北京：电子工业出版社，2012.4
ISBN 978-7-121-16729-4

Ⅰ. ①E… Ⅱ. ①宋… Ⅲ. ①JAVA 语言－程序设计Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2012）第 064377 号

责任编辑：吴　源
印　　刷：
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编：100036
　　北京市海淀区翠微东里甲 2 号　邮编：100036
开　　本：787×1092　1/16　印张：15　字数：384 千字
印　　次：2012 年 4 月第 1 次印刷
定　　价：31.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

EJB（Enterprise Java Bean）是 Java EE 的一部分，定义了一个用于开发基于组件的企业多重应用程序的标准。为了帮助读者全面了解 EJB 的知识体系及最新的 EJB 3.1 特性，笔者精心编著了本书。目前，关于 EJB 3.1 组件技术介绍方面的资料比较少，主要参考 EJB 3.1 的新特性是根据官方所发布的文档。而 EJB 3.1 版本和之前版本有诸多不同，对 EJB 的扩展性和简易性进行了很大的提升。关于 EJB 3.1 的最新特性可查看第 1 章的有关内容。

本书依照循序渐进的学习规律，首先介绍 EJB 的发展历史、基本概念和开发环境的搭建；在读者掌握了这些基础知识之后，再结合大量的代码实例对各个知识点进行深入浅出的讲解，引领读者学习 EJB 的各个组件在项目开发中的实际应用。

本书由宋智军、米伟哲、武桂香编著，邱仲潘负责审校工作，同时感谢冯姝慧、邹文、邓欣欣、王帅等给予的帮助和支持。作者非常感谢电子工业出版社的编辑和出版，他们为本书的选题策划、编辑加工和出版发行付出了辛勤的劳动。由于编者水平有限，加之时间仓促，书中难免有疏漏和不足之处，恳请专家和广大读者指正。

本书结构严谨，内容翔实，可读性强，可作为广大读者快速掌握 EJB 3.1 的自学实用指导书，也可作为大专院校程序设计课程的指导教材。

目　录

第1章 概 述

1.1 EJB的发展历史

EJB（Enterprise Java Bean，企业Bean）是生成业务应用的主要分布式组件模型，是J2EE体系的核心部分。EJB使业务逻辑实现与系统级服务分开，从而使开发者能够轻松构建企业级分布式组件应用。从1998年EJB 1.0到现在的最高版本EJB 3.1，EJB得到了很好的普及和发展。首先让我们简单回顾一下EJB的发展历史。

（1）EJB 1.0

1998年3月在San Francisco召开的JavaOne98开发者大会上，Sun公司正式发布了EJB 1.0，这是EJB的第一个版本。该版本开始支持有状态和无状态的服务器对象（称为会话Bean），这也是最早的会话Bean，并且支持持久化域对象（称为实体Bean）。为了兼容性和远程访问，EJB 1.0提供了良好的分布式支持功能，它允许通过远程接口来远程调用EJB中的业务方法。正是由于它提供了一组远程访问的规范，导致远程也需要部署基础架构，加大了系统的开销，影响性能。

（2）EJB 1.1

EJB 1.1的发布对1.0版本中的一些特性做了补充说明和改进，其中主要引入了XML格式的部署文件，使XML配置文件以声明的方式来对原数据进行配置，而不是像1.0版本那样需要一个单独的class文件来存储。

（3）EJB 2.0

EJB 2.0通过引入本地接口的概念和措施，解决了以前版本中强制远程访问所带来的系统开销和性能下降的问题，它允许开发者自己决定是否要让EJB组件支持远程访问，如果EJB组件不需要支持远程访问，则让Bean实现类实现本地接口即可，这就可以避免远程访问所带来的系统开销和性能下降。EJB 2.0最大的贡献就是提出了消息驱动Bean，能够参与异步消息系统，从而解决了EJB应用的异步调用问题。

（4）EJB 2.1

EJB 2.1增加了Web Service的支持，并兼容SOAP协议，允许会话Bean暴露站点接口，从而更有利于异构系统的整合。不仅如此，EJB 2.1还加入了定时自动执行功能，可按照指定的时间或者时间间隔调用EJB的业务方法，这种定时功能非常方便地为系统提供任务高度的支持。除此之外，EJB 2.1还提供了扩展EJB QL的功能，并引入XML schema来替换定义ejb-jar.xml部署描述文件的DTD。

（5）EJB 3.0

EJB 3.0不是以前版本的修订和改良，而是与旧版本有截然不同的规范，其中最突出的特点是实体Bean模型被实体模型所取代，只保留了原有的Session Bean和消息驱动Bean。EJB 3.0的开发模式，极大地简化了EJB的开发模式，也优化了EJB技术本身，并降低了EJB的复杂

性。 EJB 3.0 提供服务器解决方案的完整套装，包括持久化、消息、轻量型计划、远程处理、Web 服务、依赖注入（Dependency Injection，DI）和拦截器。这就是说，我们不必花费很多时间寻找第三方工具并将其集成到应用程序中。此外，EJB 3.0 提供与其他 Java EE 技术以及持久化层技术的无缝集成，比如，JDBC、Java 事务 API（Java Transaction API，JTA）、Java 消息服务（Java Messaging Service，JMS）、Java 验证和授权服务（Java Authentication and Authorization Service，JAAS）、JavaServer Pages（JSP）、servlet、JavaServer Faces（JSF）和 Swing，等等。

（6）EJB 3.1

EJB 3.1 的主要目标是使 EJB 尽可能简单，其核心思想就是要简化 EJB 架构，同时引入一些急需的新特性，目前已经作为 JavaEE 6 规范的一部分被发布，去掉了 EJB 3.0 繁琐的编程模型，而带给 Java EE 6 一个更简单的编程环境。

1.2 EJB 3.1 的新特性

EJB 3.0 给 Java EE 5 带来了由重量级向轻量级的转变，而 EJB 3.1 是在 EJB 3.0 基础上将简单开发深入下去并增加了一些急需的特性，下面简单总结了 EJB 3.1 中的一些新特性。

● 引入单例

EJB 3.1 之前，Session Bean 被设计为单线程的模型，此模型要求同一时刻某一个 bean 的实例只能被一个线程使用，其中无状态会话 Bean 采用实例池，而有状态会话 Bean 采用激活钝化技术实现。在 EJB 3.1 中，通过@Singleton 标记该 Session Bean 为 Singleton，这样每个应用程序会确保只实例化一次，如下面的示例所示。

```
@Singleton
public class MySingleton {
    private Properties props;
    public String getProperty(String name)
    {
        ...
}
@PostConstruct
    public void initialize
    {
        //props = ...
}
}
```

● 提供了更强大的 Timer Service

EJB 从 2.1 版本就引入了定时的机制，但是一直到 3.0，定时机制使用起来都不是很方便，比如要注入定时服务到 bean 中。EJB 3.1 引入了申明式的定时服务，提供了编程和注释两种机制来进行定时服务，如下面的定义所示。

```
@Stateless
```

```
public class MyTime {
    @Schedules{
        {
            @Schedule(hour = "10")
        }
public void MyTime(Timer time)
        {
                ...
}
}
}
```

● EJB 接口是可选的

EJB 3.0 中要求 Bean 至少实现一个接口，而在 EJB 3.1 中的 Bean 可以不需要接口，只要在相应的实体模型中标注上@Stateless 或@Stateful 就可完成 Session Bean 的功能。我们可以看一个从 EJB 3.0 中改过来的实例。

```
@Stateless
public class HelloBean {
    public String getHello()
        {
                ...
}
}
```

● 异步服务

在 EJB 3.1 之前，在会话 Bean 上的任何函数调用都是同步的。而在 EJB 3.1 中支持异步函数调用的会话 Bean，通过使用@Asynchronous 注解的 Bean 函数实现异步调用。异步函数可以返回一个 java.util.concurrent API 的 Future 对象，当客户端想获取调用的状态时这个非常有用，通过检索函数返回值，可以检查一个异常或者取消调用。比如，在客户确认订单后，消息驱动 Beans OrderBillingMDS 异步地给客户开出账单，并且当支付完成后，更新订单信息，请看下面的示例。

```
@Stateless
public class OrderBillingServiceBean implements OrderBillingService {
    ...
    @Asynchronous
    public void billOrder(Order order) {
        try {                    // Attempt to charge the order.
            bill(order);
            // Send email notification of billing success.
            notifyBillingSuccess(order);
            order.setStatus(OrderStatus.COMPLETE);
        } catch (BillingException be) {
```

```
            // Send email notification of billing failure.
            notifyBillingFailure(be, order);
            order.setStatus(OrderStatus.BILLING_FAILED);
        } finally {
            update(order);
        }
    }
    ...
}
```

由于使用了@Asynchronous 标注，当客户端调用 OrderBillingService.billOrder()方法时，调用会立即返回，而不会一直阻塞，直到 billOrder 方法执行完毕后才正常退出该方法。EJB 容器会去确认哪些方法需要同步执行。请注意异步方法的返回值是 void 型的。不过，EJB 3.1 还支持 java.util.concurrent.Future<V>作为返回类型，其中 V 表示异步调用后，返回结果的类型，Future<V>接口允许我们做以下几类事。

- 取消异步调用
- 检查调用是否运行结束
- 检查是否有异常
- 得到异步调用后的结果
- 简化打包机制

EJB 3.0 可选的 XML 部署描述符已经大大简化了 JavaEE 应用程序的打包和部署。尽管如此，JavaEE 打包仍然必须分开为 web 和 EJB 模块打包成 jar 文件。EJB 3.0 中的打包方式如图 1.1 所示。

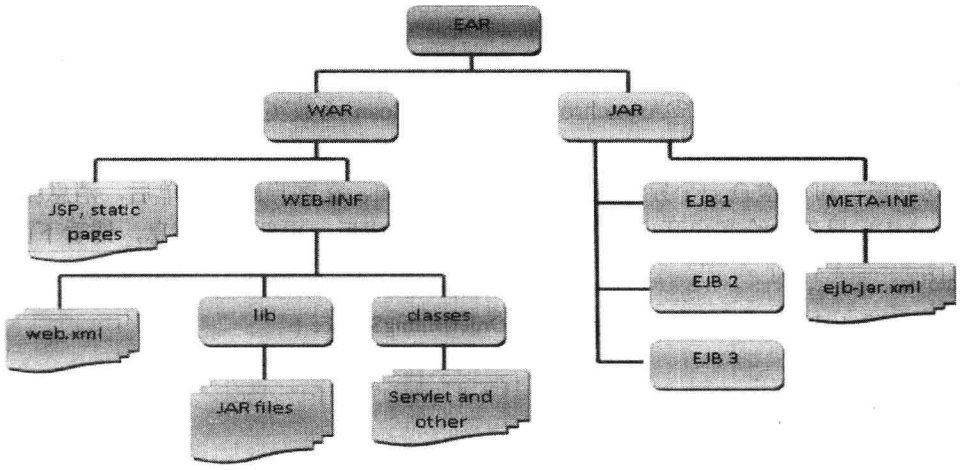

图 1.1　EJB 3.0 打包方式结构图

在使用 EJB 3.1 后，开发人员可以直接将 classes 目录下的 EJB 和 servlets 一起打包到 WAR 文件中，如图 1.2 所示。

除了上述新特性外，EJB 3.1 的特性还包括更简化的 JMS，JavaMail 和数据库注入，提供了接口服务的支持便于把第三方库整合进来，可运行的 embedded 的容器中便于 Java SE 环境

进行 JUnit 测试。EJB 3.1 中还定义了统一的全局 JNDI 命名方式，采用了统一的方式来获取注册的 Session Beans，这样就避免了同一个应用程序中的那些 Session Beans 在不同供应商的容器中 JNDI 命名不同的问题。有关 EJB 3.1 更多的特性可从以下链接获得：http://jcp.org/en/jsr/detail?id=318。

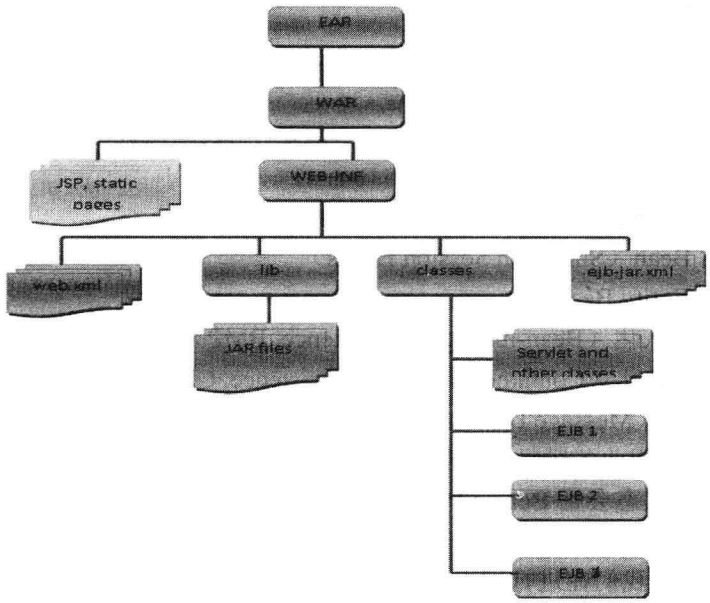

图 1.2　EJB 3.1 的打包方式结构图

1.3　EJB 3.1 结构简介

在 Java EE 6 中，EJB 3.1 让开发变得更简单，并且功能更强大，比如可以省略业务接口、引入了 Singleton Beans、可以直接用 WAR 文件打包 EJB 组件、异步会话 Bean、轻量级的 EJB、在 Java SE 环境中嵌入 API 执行 EJB 等。图 1.3 展示了 EJB 3.1 在 Java EE 6 体系结构中的位置。

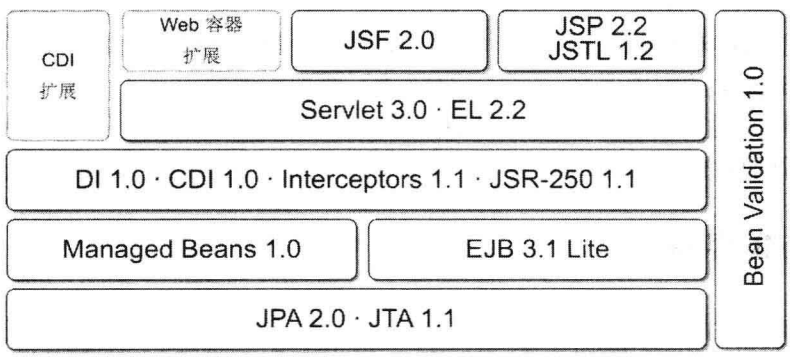

图 1.3　EJB 3.1 在 Java EE 6 体系结构中的位置

图 1.3 展示了 Java EE 6 平台体系结构中各元素间的逻辑关系，每个独立矩形部分标明的容器是 Java EE 6 运行时环境，它为应用程序组件提供了必要的服务。图中层次结构表示必须提供对 Java EE 6 平台对应部分的访问，其中 Web 容器提供用户界面、接收用户输入、数据

输出属于视图层组件。EJB 容器实际上属于业务逻辑层，根据视图层传送来的数据，进行实际业务逻辑处理，然后再把处理后的结果返回给视图层。因此，EJB 3.1 在 Java EE 6 体系结构中处于重要地位。

1.4 EJB 3.1 组件的种类

EJB 容器可以接受如下三种类型的 EJB。

- 会话 Bean（Session Beans）
  - 无状态会话 Bean（Stateless Session Beans）
  - 有状态会话 Bean（Stateful Session Beans）
  - 单例会话 Bean（Singleton Session Beans）
- 实体 Bean（Entity Beans）
  - Bean 管理持久化（Bean-managed persistence, BMP）
  - 容器管理持久化（Container-managed persistence, CMP）
- 消息驱动 Bean（Message Driven Beans， MDBs）

会话 Bean 主要负责业务逻辑的处理，根据处理时的状态保持与否，会话 Bean 又可分为有状态会话 Bean、无状态会话 Bean 和单例会话 Bean。而对象的“状态”是指对象的属性值，也就是对象所携带的数据。有状态会话 Bean 是包含状态的分布式对象，在整个会话过程中都携带客户端信息。比如我们在进行网上支付操作时，就需要一个有状态会话 Bean，因为支付过程可能要进行多步，服务器端必须随时了解用户已经进行到了哪一步。无状态会话 Bean 是一类不包含状态信息的分布式对象，允许来自多个客户端的并发访问。由于不必控制与用户间的对话信息而减少开销，无状态会话 Bean 不像有状态会话 Bean 那样具有资源集约性。比如，一个发送邮件的 EJB 就可被设计为一个无状态会话 Bean，因为在整个会话期间，用户只要向服务器提交一个发送邮件指定地址的操作即可。单例会话 Bean 是 EJB 3.1 引入的一种新的 Bean。

实体 Bean 是含有持久化状态的分布式对象，主要负责数据库的访问。实体 Bean 的一个实例所表示的数据通常代表了关系数据库中特定表的特定一行数据。在 EJB 3.1 中，实体 Bean 部分的功能可以通过 JPA 来实现。根据对数据库持久化的程度不同，实体 Bean 又可分为 Bean 持久化操作（主要针对数据库的访问，数据的创建、更新、删除等）和容器管理持久化（主要是容器根据 deploy 的配置信息 ejb-jar.xml 完成）两种类型。

消息驱动 Bean 是支持异步行为的分布式对象，主要用来处理异步消息。客户端调用会话 Bean 和实体 Bean 方法时，信息驱动 Bean 会一直处于等待状态，直到被调用的方法处理完毕。

1.5 EJB 容器

容器在 EJB 中占有重要地位，因为它向程序开发人员提供了最大程度的方便。安装在应用程序服务器中的企业 Bean 并不是直接与服务器通信，而是通过 EJB 容器提供企业 Bean 和服务器之间的接口进行通信。EJB 容器将企业 Bean 从低层应用程序服务器中分离出来，并提供一个在 Bean 和容器之间的应用程序设计接口（API）。程序开发人员可以实现许多基础的服务，如线程化、对事务处理的支持及对数据存储的管理和检索等。在 Java EE 6 的规范中，EJB 有以下四种容器类型。

- 小应用程序（applet）
- 应用程序客户端容器（swing）
- WEB 容器（servlet 和 JSP）
- EJB 容器

1.6　EJB 的组成

一旦企业 Bean 准备好后，就被打包成一个标准的 Java 文件（ejb-jar 文件）。该文件可以包括一个或多个企业 Bean。对于每一个企业 Bean，由以下几部分组成。

- Remote 接口：说明了客户端能调用的函数。
- Home 接口：使客户生成和定位一个企业 Bean 的实例。
- Bean 类：实现了企业 Bean 的语义。
- 配置描述器：用 XML 配置描述器或用 deloyer 通过配置描述器来提供没有在代码中申明的信息。

1.6.1　Bean 类

Bean 类用来实现在 Remote 接口中定义的业务逻辑方法，对于有状态会话 Bean，还要保持用户的状态，是 EJB 组件的主要功能。在编写 Bean 类时，要遵循 EJB 规范所定义的接口和相关规则。在 Bean 类继承 EJB 规范所定义的接口后，开发人员必须实现接口中定义的相关方法，EJB 容器会调用这些接口中定义的方法，从而管理 EJB 组件。前面所介绍的 EJB 容器所接受的三种类型的 Bean（会话 Bean、实体 Bean，消息驱动 Bean）都继承于 javax.ejb.EnterpriseBean。因此，这三种类型的 Bean 都必须实现 javax.ejb.EnterpriseBean 接口，即使这一接口并没有定义任何方法，具体的细节请查看 EJB 3.1 规范文档。

1.6.2　EJB 对象

当用户希望使用 Bean 类生成的实例时，用户根本不能调用到 Bean 实例中的方法。因为 EJB 容器会拦截这一调用，并委派给 Bean 实例。在拦截调用请求期间，EJB 容器能够隐式地提供中间件给它服务。EJB 容器实际上是用户代码和 EJB 对象间的中间件，它会依据 EJB 组件的具体情况而生成相应的 EJB 对象。

1.6.3　Remote 接口

每一个企业 Bean 都必须有一个 Remote 接口。Remote 接口定义了应用程序规定客户可以调用的逻辑操作。这些是一些可以由客户调用的公共的方法，通常由企业 Bean 类来实现。

所有的 Remote 接口中的方法必须声明为公有（public）的，并必须抛出 java.rmi.RemotException 异常。另外，所有的 Remote 接口中的方法定义的参数和都必须是在 RMI-IIOP 中有效的。对每一个在 Remote 接口中定义的方法，在企业 Bean 类里面都要有相应的方法。相应的方法必须要有同样的名字、同样类型和数量的参数，同样的返回值，而且还要抛出同样的例外。

企业 Bean 类的客户并不直接访问 Bean，而是通过 Remote 接口来访问的。

1.6.4 Local 接口

上面提到的 Remote 接口其实是远程接口，而 Local 接口是本地接口，供 EJB 组件的本地客户操作业务功能使用。当 Web 层调用 app 层时，使用 Remote 接口；当会话 Bean 和实体 Bean 之间调用时，使用的是 Local 接口。由于 Remote 接口对性能影响很大，所以在程序设计的时候我们尽量使用 Local 接口，即 façade 模式。如果为了让 EJB 组件同时支持本地和远程客户，开发者就必须同时实现 Local 接口和 Remote 接口。

1.6.5 Home 接口

Home 接口定义了用于创建、销毁、查找本地或远程 EJB 对象的若干方法，各个 EJB 组件需要提供相应的 Home 接口，依据具体 EJB 组件的不同，所有的 Home 接口必须继承 javax.ejb.EJBHome 或 javax.ejb.EJBLocalHome 接口。

1.6.6 配置描述器

为了将中间件需求告知给 EJB 容器，企业 Bean 需要在 XML 配置描述器中声明相应的中间件服务需求。比如要配置一个企业 Bean，意味着将这个企业 Bean 安装到容器中去。安装过程包括了生命周期管理、事务控制、安全性服务等。EJB 容器会分析配置描述器，并将相应的服务需求提供给 EJB 组件。

1.7 EJB 调用过程

容器和组件实现了 EJB 的基础构造，组件主要完成业务逻辑部分，容器不仅是组件的运行环境，而且向组件提供公共服务接口。这些基础构造主要提供了声明周期管理、代码产生、持续性管理、安全、事务管理、锁和并发行管理等服务。EJB 的调用过程，如图 1.4 所示。

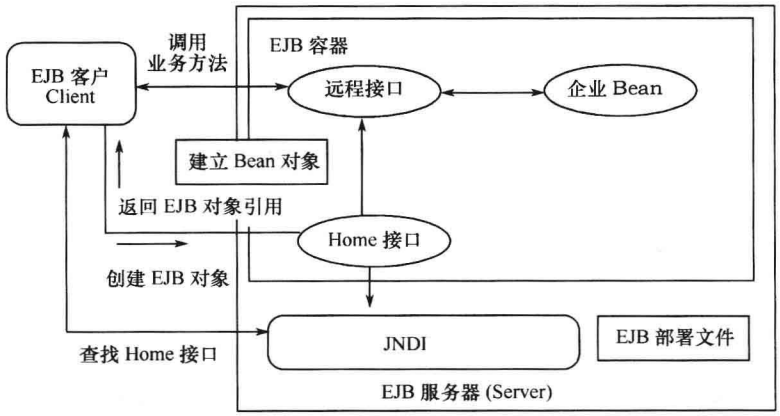

图 1.4 EJB 的调用过程

首先，客户端程序使用 lookup 方法查找 JNDI，EJB 服务器的 JDNI 服务根据事先登录的信息匹配 lookup 查询内容，生成 Home 实例。然后，客户端得到对 Home 实例的引用并调用 Home.create 方法，创建相应的 Bean 实例，生成相对应的 Remote Object 实例，客户端得到对 Remote Object 实例的引用，最后，客户端调用业务逻辑方法。

第 2 章　EJB 开发基础

本书第 1 章介绍了 EJB 的发展历史及 EJB 3.1 的新特性。本章将逐步深入到 EJB 的具体细节中，并通过实例展现 EJB 3.1 的开发过程，可以使开发者从零开始研究并使用 EJB。本章内容主要介绍 EJB 环境的搭建，编程中最为重要、基础的内容。

2.1　开发环境和工具

EJB 是 Java EE 的重要组成部分，要进行 EJB 开发，至少需要以下软件工具。

- JDK：Java 开发者工具包，用来对编写的 Java 源程序进行编译，对部署描述符、类文件等进行打包，生成.jar 文件。
- Java EE 兼容的服务器：用来对上一步生成的.jar 文件进行部署。
- Java 源代码编辑工具

由于 Java 的开放性，各种免费、开源的 Java EE 开发工具也十分流行，它们具有占用内存小、运行速度快等优点，常用的免费 Java EE 工具如下。

- Ant：用来对 Java EE 程序进行编译和部署。
- Eclipse：IBM 公司将 Websphere Application Developer Studio 基础源代码公开，交给开源社区后发展非常迅速的免费的 Java 源代码编辑工具，并且 Eclipse 支持第三方的插件来扩展其功能。目前 Eclipse 可以通过安装第三方的 Java EE 插件良好地支持 EJB 的开发，并且内置了支持 Ant 插件。
- Netbeans：SUN 公司将 Java Enterprise Developer Studio 的基础源代码公开，交给开源社区后形成的免费的 Java 开发工具。

本书采用 Eclipse 结合相应的 MyEclipse 插件，介绍 EJB 的开发和测试。

在 EJB 容器中，常见的应用服务器如下。

- SUN 公司的 Java EE 企业版：免费的 Java EE 容器，可作为 Java EE 功能的演示和教学版。
- IBM 公司的 Websphere Application Server：市场占有率最高的应用服务器，因为具有非常好的稳定性，常被作为重要电子商务场合的应用服务器。
- BEA 公司的 Weblogic：市场占有率仅次于 Websphere，对 Java EE 标准支持得比较好，具有较好的执行速度。
- 开源免费的 JBoss ：无需安装，速度、性能都十分优异，对系统要求较低，部署 EJB 速度非常快。

本书采用 JBoss 作为 EJB 的开发测试容器，因为 JBoss 内置了 Web 容器、EJB 容器、JBoss 消息服务器和 WebService 支持；JBoss 还内置了一个免费的关系数据库 Hypersonic，通过 JBoss 的 JMX 控制台，就可直接启动 Hypersonic 的管理工具；JBoss 的安装很简单，只要将下载的

zip 文件解压缩到硬盘上某一个目录中就可以了；由于 JBoss 是一个用纯 Java 编写的应用服务器，所以运行时需要 JDK 的支持。

2.2 构建开发环境

在前面已经介绍要进行 EJB 开发，至少需要的软件工具，本书中的 EJB 实例采用了 JBoss AS 6.0 服务器，JDK 1.6，Eclipse IDE for Java EE Developers 以及 MyEclipse 9.0 插件进行构建 EJB 环境。

2.2.1 JDK 的安装配置

JDK 的最新版本（目前 JDK 最新版本为 JDK1.6）可以从官网下载（网址为：http://www.oracle.com/technetwork/java/javase/downloads/index.html）。下载完成后，进行安装，安装步骤如下。

- 安装 JDK1.6

单击 jdk-6u26-windows-i586.exe，进入如图 2.1 所示的界面。

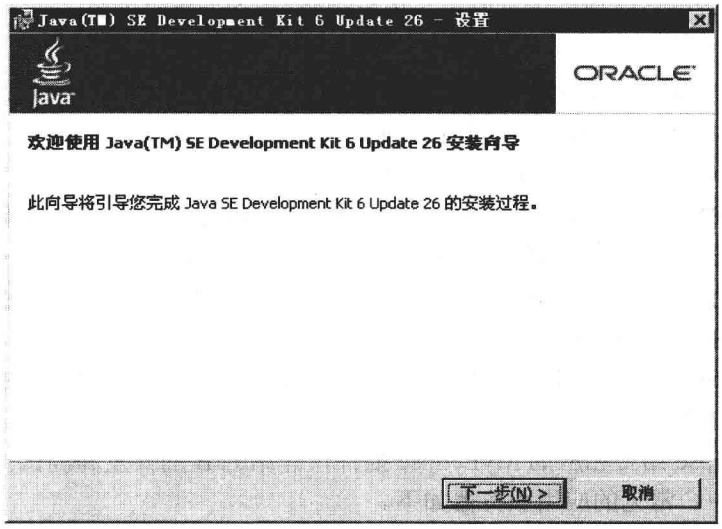

图 2.1 JDK1.6 安装界面

若要采用默认安装，只需要按照安装向导一步一步选择【下一步】即可，直到单击到【完成】按钮，JDK 安装成功。

- 设置环境变量

安装完 JDK 后，还需要设置环境变量，在这里一共设置了三个环境变量：PATH，CLASSPATH，JAVA_HOME（这里变量名最好用大写）。设置方法如下。

- 右击“我的电脑”→“属性”→“高级”→“环境变量”→“系统变量”，出现如图 2.2 所示的界面。
- 在“系统环境变量”中设置上面提到的三个环境变量，如果变量已经存在就选择“编辑”，否则选“新建”。单击系统变量下的“新建”按钮，出现新建系统变量对话框，如图 2.3 所示。

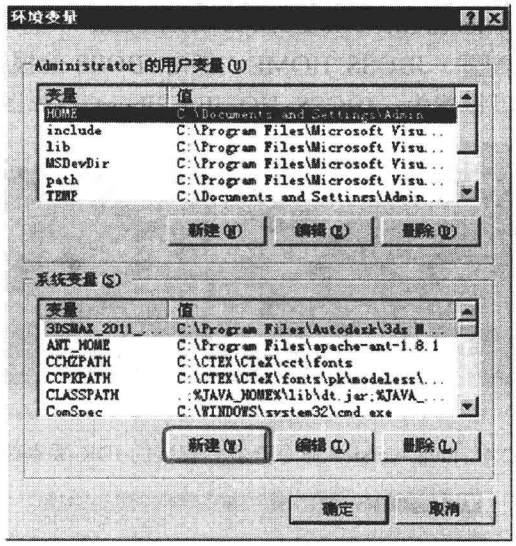

图 2.2　JDK 环境变量设置

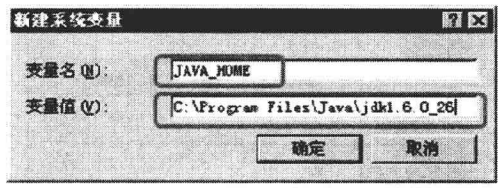

图 2.3　“新建系统变量”对话框

三个变量的设置如表 2.1 所示。

表 2.1　三个变量的设置

变量名	变量值
JAVA_HOME	指明 JDK 安装路径，就是刚才安装时所选择的路径（默认路径为：C:\Program Files\Java\jdk1.6.0_26），此路径下包括 lib，bin，jre 等文件夹
PATH	使得系统可以在任何路径下识别 java 命令，设为：%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin
CLASSPATH	CLASSPATH 为 java 加载类路径，只有类在 CLASSPATH 中，java 命令才能识别，设为：.;%JAVA_HOME%\lib;%JAVA_HOME%\lib\tools.jar（注意：CLASSPATH 变量以“.;”开始，表示当前路径。）

■ 环境变量设置完成后，单击“确定”，退出环境变量设置。

■ JDK 检验：JDK 安装完成后，可以检验是否安装成功，在 DOS 命令行窗口下，键入 java –version 命令可以查看安装的 JDK 版本信息，如图 2.4 所示；键入 java 命令，可以看到此命令的帮助信息，则说明安装成功。

2.2.2　MyEclipse+JBoss 的安装配置

安装 JBoss，实际上就是将下载到的 JBoss 包解压到指定的位置即可（可在官网下载：