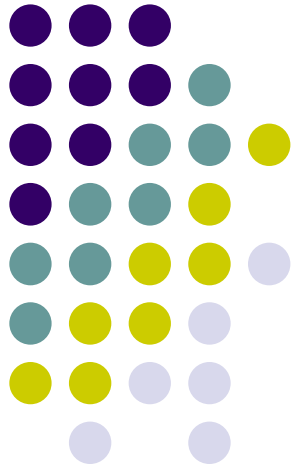
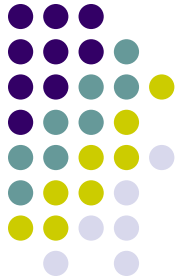
第九章　EJB的原理与开发





主要内容

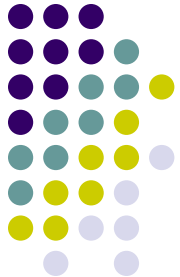
- EJB的概念及工作原理
- EJB的开发

EJB的概念及工作原理

- 什么是EJB
- JavaBean与EJB的区别
- EJB分布式计算的角色组成
- EJB的体系结构
- EJB的工作原理
- EJB的分类及其功能

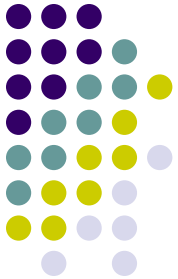
什么是EJB

- EJB(Enterprise JavaBean)并不是一个产品。它是Java 服务器端服务框架的规范。
- EJB规范定义了如何编写应用服务器端组件，提供了组件与管理组件的应用服务器之间的标准约定，使得开发人员能够快速开发出具有伸缩性的企业级应用。



- 软件厂商根据EJB规范来实现应用服务器
- 企业应用的开发者可以专注于支持应用所需的那些商业逻辑的开发，而不用担心周围框架的实现问题，例如：处理事务行为、安全、连接共享或线程的代码等等。

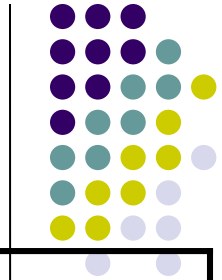
JavaBean与EJB的区别



- JavaBean是Java 的组件模型。在JavaBean规范中定义了事件和属性等特征。
- Enterprise JavaBeans 也定义了一个Java 组件模型，但是Enterprise JavaBeans 组件模型和JavaBeans 组件模型是不同的。
- JavaBean重点是允许开发者在开发工具中可视化的操纵组件。JavaBean规范详细地解释了组件间事件登记、传递、识别和属性使用、定制和持久化的应用编程接口和语意。

- Enterprise JavaBeans 的侧重点是详细地定义了一个可以移植地部署Java组件的服务框架模型。因此，其中并没提及事件，因为enterprise bean 通常不发送和接受事件。同样也没有提及属性------属性定制并不是在开发时进行，而是在运行时（实际上在部署时）通过一个部署描述符来描述。
- 它们都是组件模型规范，但是前者说明了开发工具中应用程序组装的问题，而后者则侧重于部署组件的服务框架的细节。

- 不要错误地认为JavaBeans是用于客户端的开发，Enterprise JavaBeans是用于服务器端的开发。JavaBeans也可作为进行非图形化服务器端Java应用开发的组件模型。
- 区别是当你使用JavaBeans创建服务器应用时，你还得设计整个的服务框架。用Enterprise Javabeans框架是现成的，你只需遵守它的APIs.对于复杂的服务器端应用程序，显然使用Enterprise JavaBeans比重新开发更简单。



- 两者之间一些具体区别：

JavaBean	Enterprise JavaBean
运行时决定其可见性，例如，可视化GUI JavaBean组件、按钮、下拉列表或表格。	基本是不可见的，远程对象。
主要为运行于本地客户端，服务于单一线程。	主要为远程可执行组件，只运行于服务器端
是一种组件技术，用户生成通用Java组件，进而用于开发Java应用程序或Java小应用程序。	也是一种组件技术，但其并不是建立于JavaBean技术之上。
具有外部Properties接口，用于开发工具解释并修改bean的功能。	其具有部署描述文件，用于描述EJB面向外部开发工具的功能。
可能具有多种属性编辑器，例如BeanInfo类。	不具有任何属性编辑器的概念，所有描述信息基本包含在部署描述文件内。
内部没有分类。	分为会话bean、实体bean和消息驱动的bean三种。
内部没有显示的事务处理的技术支持。	是面向事务的，EJB容器提供事务处理的技术支持。

EJB分布式计算的角色组成

- EJB 组件结构是基于组件的分布式计算结构，是分布式应用系统中的组件。
- 一个完整的基于EJB 的分布式计算结构由六个角色组成，这六个角色可以由不同的开发商提供，每个角色所作的工作必须遵循Sun 公司提供的EJB 规范，以保证彼此之间的兼容性。这六个角色分别是：

（1）EJB 组件开发者(Enterprise Bean Provider)

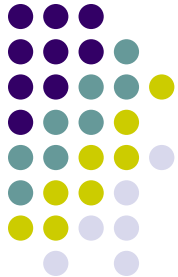
EJB 组件开发者负责开发执行商业逻辑规则的EJB 组件，开发出的EJB 组件打包成ejb-jar 文件。EJB 组件开发者负责定义EJB的remote和home接口，编写执行商业逻辑的EJB class,提供部署EJB的部署文件(deployment descriptor)。部署文件包含EJB的名字，EJB 用到的资源配置，如JDBC 等。EJB 组件开发者是典型的商业应用开发领域专家。

EJB 组件开发者不需要精通系统级的编程，因此，不需要知道一些系统级的处理细节，如事务、同步、安全、分布式计算等。

（2）应用组合者(Application Assembler)

应用组合者负责利用各种EJB 组合一个完整的应用系统。

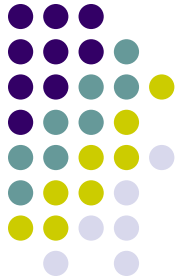
应用组合者有时需要提供一些相关的程序，如在一个电子商务系统里，应用组合者需要提供JSP(Java Server Page)程序。应用组合者必须掌握所用的EJB 的home和remote 接口，但不需要知道这些接口的实现。



（3）部署者(Deployer)

部署者负责将ejb-jar 文件部署到用户的系统环境中。系统环境包含某种EJB Server 和EJB容器。部署者必须保证所有由EJB组件开发者在部署文件中声明的资源可用，例如，部署者必须配置好EJB 所需的数据库资源。

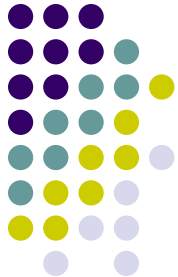
部署过程分两步：部署者首先利用EJB容器 提供的工具生成一些类和接口，使EJB容器能够利用这些类和接口在运行状态管理EJB。部署者安装EJB 组件和其他在上一步生成的类到EJB容器中。 部署者是某个EJB 运行环境的专家。



（4）EJB 服务器提供者(EJB Server Provider)

EJB 服务器提供者是系统领域的专家，精通分布式交易管理，分布式对象管理及其它系统级的服务。

EJB 服务器提供者一般由操作系统开发商、中间件开发商或数据库开发商提供。在目前的EJB 规范中，假定EJB 服务器提供者和EJB 容器提供者来自同一个开发商，所以，没有定义EJB 服务器提供者和EJB 容器提供者之间的接口标准。



（5）EJB 容器提供者(EJB Container Provider)

EJB 容器提供者提供以下功能：提供EJB 部署工具为部署好的EJB组件提供运行环境 。EJB 容器负责为EJB 提供交易管理，安全管理等服务。

EJB 容器提供者必须是系统级的编程专家，还要具备一些应用领域的经验。EJB 容器提供者的工作主要集中在开发一个可伸缩的，具有交易管理功能的集成在EJB 服务器中的容器。EJB 容器提供者为EJB 组件开发者提供了一组标准的、易用的API 访问EJB 容器，使EJB组件开发者不需要了解EJB 服务器中的各种技术细节。

EJB 容器提供者负责提供系统监测工具用来实时监测EJB 容器和运行在容器中的EJB 组件状态。

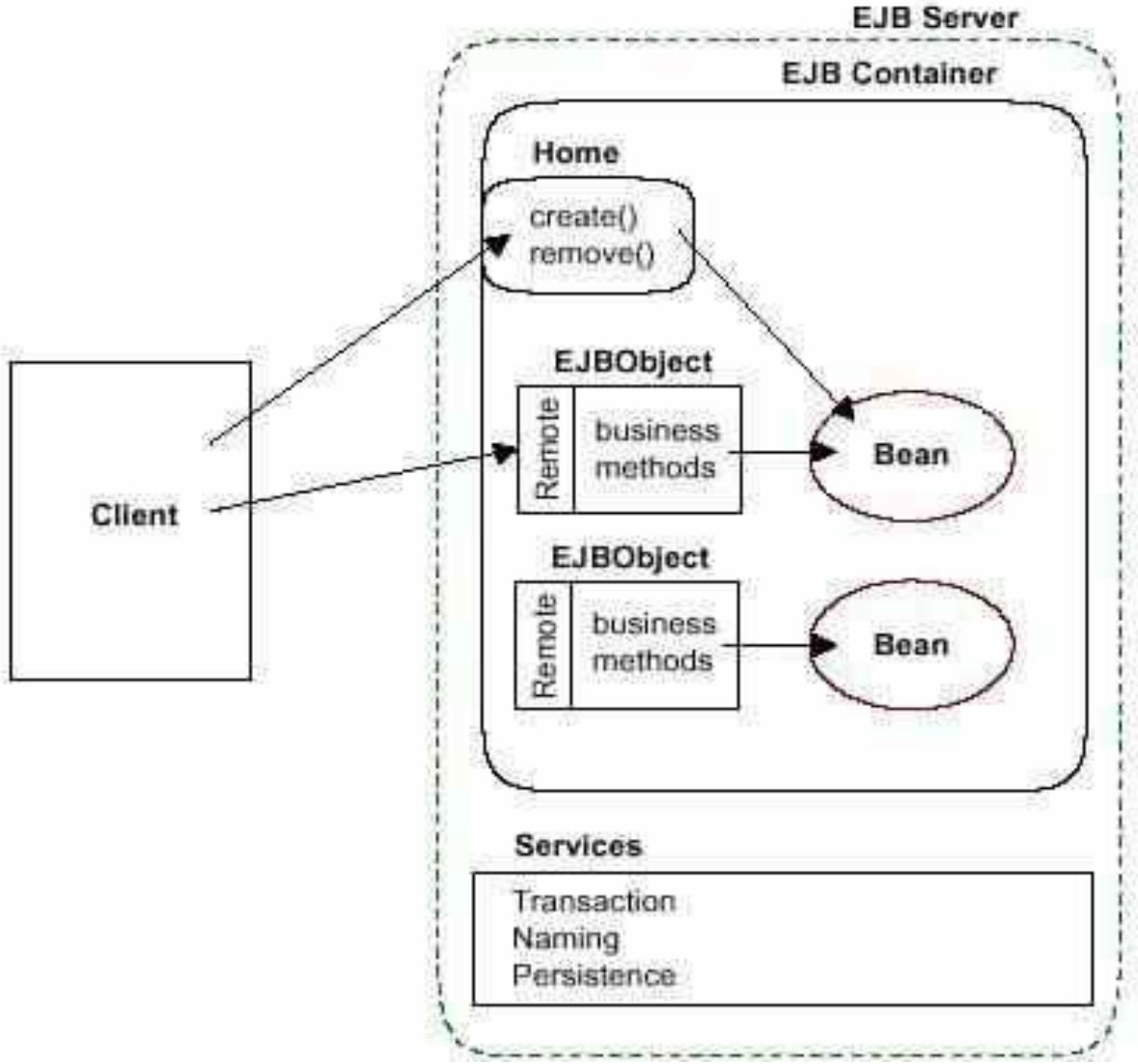
（6）系统管理员(System Administrator)

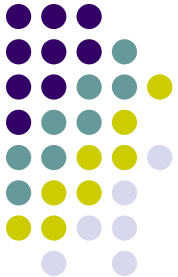
系统管理员负责为EJB服务器和容器提供一个企业级的计算和网络环境。

系统管理员负责利用EJB 服务器和容器提供的监测管理工具监测EJB 组件的运行情况。

EJB的体系结构

EJB的体系结构描述如右图所示：





1、EJB服务器(EJB Server)

- EJB 服务器是管理EJB 容器的高端进程或应用程序，并提供对系统服务的访问。
- EJB 服务器也可以提供厂商自己的特性，如优化的数据库访问接口，对其他服务（如CORBA 服务）的访问，对SSL 3.0 的支持等。一个EJB 服务器必须提供对可访问JNDI 的名字服务和事务服务支持。

2、EJB容器(EJB Container)

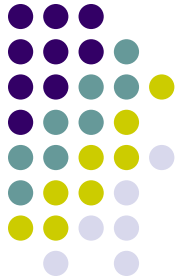
- EJB 容器是一个管理一个或多个EJB 类/实例的抽象。它通过规范中定义的接口使EJB 类访问所需的服务。容器厂商也可以在容器或服务器中提供额外服务的接口。
- 现在没有EJB 服务器和EJB 容器间接口的规范。因为目前容器通常由EJB服务器来提供，所以一旦接口标准化了，厂商就可能提供可以在任何兼容的EJB 服务器上运行的容器。

3、Home接口(Home Interface)

- Home 接口列出了所有定位、创建、删除EJB 类实例的方法。Home 对象是home 接口的实现。
- EJB 类开发者必须定义home 接口。容器厂商应该提供从home 接口中产生home 对象实现的方法。

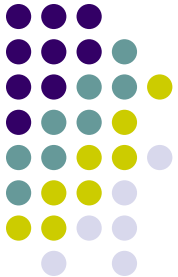
4、Remote接口(Remote Interface)

- 远程接口（remote interface）列出了EJB 类中的商业方法



5、EJB对象(EJBObject)

- EJB对象实现远程接口，并且客户端通过它访问EJB实例的商业方法。EJB 类开发者定义远程接口，容器开发商提供产生相应EJB对象的方法。
- 客户端不能得到EJB实例的引用，只能得到它的EJB对象实例的引用。当客户端调用一个方法，EJB对象接受请求并把它传给EJB实例，同时提供进程中必要的包装功能。



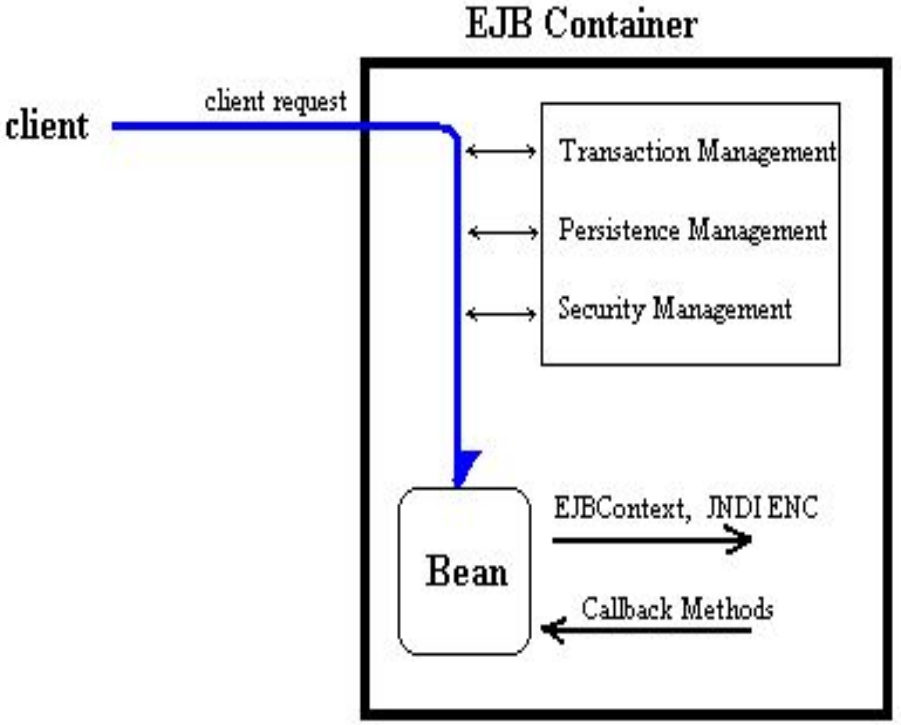
6、客户端(Client)

- 客户端应用程序通过home 对象来定位、创建、删除EJB类的实例，通过EJBObject 来调用实例中的商业方法。
- 客户端可以用Java 来编程，通过Java RMI来访问访问home 对象和EJBObject,或用其他语言编程并通过CORBA/IIOP 访问，使得部署的服务器端组件可以通过CORBA 接口来访问。

EJB的工作原理

1、EJB容器

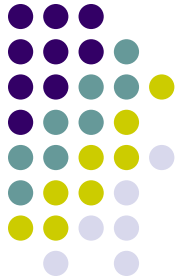
EJB组件是在EJB容器的特殊环境下运行，不能在EJB容器外部运行，EJB容器在运行时管理EJB的各个方面，包括：远程访问EJB组件、安全性、持续、事务、并行性和资源的访问与合用。如右图所示：



EJB Containers manage enterprise beans at runtime

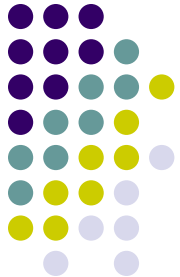
- 容器不允许客户端程序直接访问EJB组件，当客户端程序调用EJB组件上的远程方法时，容器首先拦截调用，以确保持续、事务和安全性都正确应用于客户机对EJB组件执行的每一个操作。
- 容器自动为EJB组件管理安全性、事务和持续，于是EJB组件开发人员不必将这种类型的逻辑写入EJB代码本身，可以集中精力与封装企业商业规则。

- 和Web容器管理Servlet一样，EJB容器同时管理许多EJB，为了减少内存消耗，当不使用某个EJB时，容器将它放在池中以便另一个客户重用，或者可能将它移出内存，当且仅当需要时再将它调回内存。
- 因为客户端应用程序不直接访问EJB，所以客户端程序就完全不知道容器的资源管理活动。



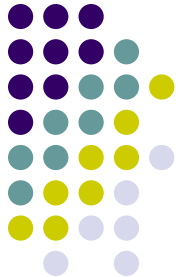
- EJB组件依赖EJB容器来获取它的资源请求，EJB通过三种机制与容器进行交互：

（1）回调方法：每个EJB都会实现EnterpriseBean接口的子类型，该接口定义了一些方法，称为回调方法。每个回调方法在EJB的生命周期期间向它提示一个不同事件，当容器要执行调用某个EJB、将其状态存储到数据库、结束事务、从内存中除去EJB等操作时，它将调用这些方法来通知该EJB，回调方法可以让EJB在事件之前或之后立即执行内部掉整。



（2）EJBContext对象

每个EJB都会得到一个EJBContext对象，它是对容器的直接引用。EJBContext接口提供了用于与容器交互的接口方法，因此EJB可以通过EJBContext对象请求关于环境的信息，如客户机的身份或事务的状态，或者EJB可以获取它自身的远程引用。



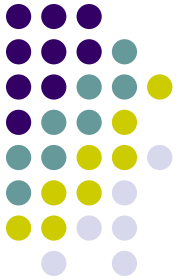
（3）Java命名和目录接口(JNDI)

JNDI是Java平台的标准扩展，用于访问命名系统，如LDAP、NetWare和文件系统等。每个EJB自动拥有一个对特定命名系统ENC(Environment Naming Context)的访问权，ENC由容器管理，EJB使用JNDI来访问ENC，JNDI ENC允许EJB访问各种资源，如：JDBC连接、其他EJB及特定于该EJB的属性。

2、EJB组件

为了创建EJB组件，除了EJB实现类外，EJB开发人员还需要提供两个接口类：本地接口和远程接口，它们分别扩展javax.ejb.EJBHome接口和javax.ejb.EJBObject接口。

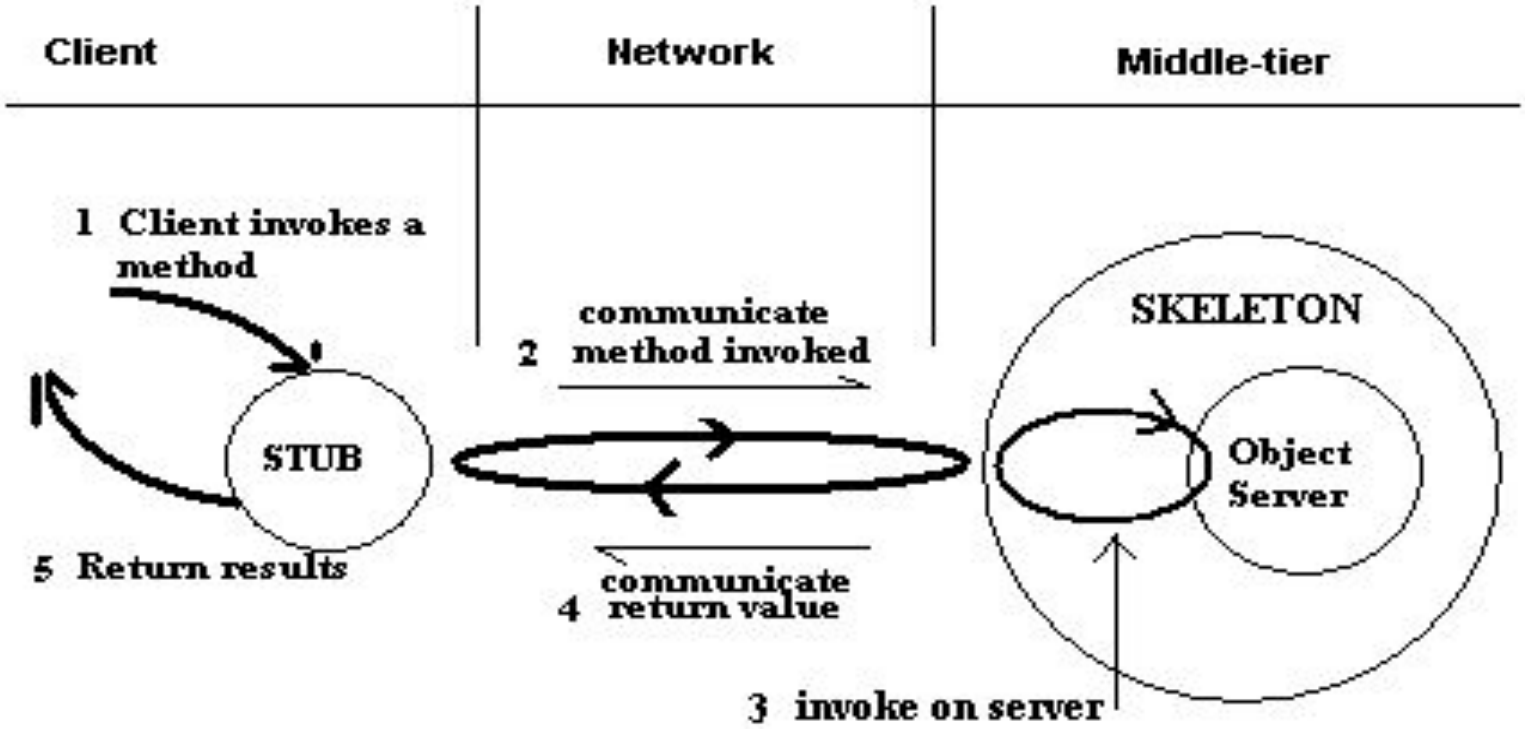
EJB实现类包含真正实现商业逻辑的代码，而本地接口和远程口暴露EJB的能力，并提供创建、更新、交互和删除EJB所需的全部方法，其中本地接口表示组件的生命周期方法(创建、更新、交互和删除)，而远程接口表示EJB的商业方法。客户机使用本地接口获取对EJB的商业的引用，使用远程接口执行商业方法。



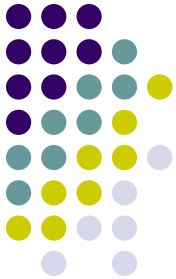
远程和本地接口是两种Java RMI远程接口，分布式对象使用java.rmi.Remote接口来表示不同地址空间（进程或机器）中的Bean。

EJB组件本身是一种分布式对象(而JavaBean是本地组件对象），指的是EJB被实例化后，其他地址空间中的应用程序也访问它，不同地址空间的应用程序之间的交互是一个复杂的过程。

作为分布式对象，EJB组件与外部的交互过程如下图所示：



- 要使一个地址空间中的对象实例在另一个地址空间中可用，需要一些涉及网络套接字的技巧。要使该技巧生效，应将实例封装在一个称作“框架”的特殊对象中，该对象拥有到另一个叫作“存根”的特殊对象的网络连接。
- 存根实现远程接口，因此它类似于一个商业对象。但存根不包含商业逻辑；它拥有到框架的网络套接字连接。每次在存根的远程接口上调用商业方法时，存根将网络消息发送到框架，告诉它调用了哪个方法。框架从存根接收到网络消息时，它标识所调用的方法以及自变量，然后调用真正的实例上的相应方法。实例执行商业方法，并将结果返回给框架，然后框架将结果发送给存根。



- 存根将结果返回给调用其远程接口方法的应用程序。从使用存根的应用程序的角度来看，存根就象在本地运行。实际上，存根只是个哑网络对象，它将请求通过网络发送给框架，然后框架调用真正实例上的方法。实例完成所有工作；存根和框架只是通过网络来回传递方法和自变量。
- 在 EJB 中，由容器实现远程和本地接口的框架，而不是 bean 类。这可以确保由客户机应用程序在这些引用类型上调用的每一个方法都先由容器处理，然后再委托给 bean 实例。容器必须拦截这些针对 bean 的请求，这样它可以自动应用持续（实体 Bean）、事务和访问控制。

3、部署EJB

前面提到，容器自动为Enterprise Bean处理持续、事务、并行性和访问控制。

EJB规范描述了一个声明机制，用于通过使用XML部署描述信息来处理这些事。将bean部署到容器中时，容器将读取部署描述信息以了解应如何处理事务、持续（实体Bean）和访问控制。

部署bean的人员将使用此信息，并指定附加信息，以便在运行时将bean与这些设施联系起来。

部署描述信息有一个预先定义的格式，所有符合EJB的bean必须使用此格式，而所有符合EJB的服务器必须知道如何读取此格式。这种格式在XML文档类型定义（DTD）中指定。

部署描述信息描述了bean的类型（会话或实体）以及远程接口、本地接口和bean类使用的类。它还指定了bean中每个方法的事务性属性、哪些安全性角色可以访问每个方法（访问控制），以及实体Bean中的持续是被自动处理还是由bean来执行。

要部署一个bean时，必须将它的远程、本地和bean类文件以及XML部署描述信息封装到jar文件中。部署描述信息在jar中必须以特定名称META-INF/ejb-jar.xml存储。这个jar文件称作 ejb-jar，是独立于供应商的；可以将它部署到支持完整EJB规范的任何EJB容器中。

将bean部署到EJB容器中时，会从jar中读取XML部署描述信息以确定如何在运行时管理bean。部署bean的人员会将部署描述信息的属性映射到容器的环境中。包括将访问安全性映射到环境的安全性系统、将bean添加到EJB容器的命名系统，等等。

EJB的分类及其功能

在EJB2.0规范中定义了三种不同类别的EJB：Session Bean（会话Bean）、Entity Bean（实体Bean）和Message-Driven Bean（消息驱动Bean）。

1、会话Bean

会话Bean是商务过程对象，执行商务逻辑、规则和工作流程。会话Bean之所以被称为会话Bean，是因为它代表的是一个动作、是一个过程，它的生存期就是调用它的客户端与它进行会话的过程。

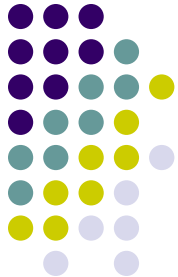
会话Bean根据其是否保存客户的状态，又分为状态会话Bean和无状态会话Bean。状态会话Bean是一种保持会话状态的服务，每个实例都与特定的客户机相关联，在与客户机的方法调用之间维持对话状态。

与之相反，无状态会话Bean不保存与特定客户的对话状态。因此状态会话Bean比无状态会话Bean具有更多的功能，而无状态会话Bean实例可以通过 EJB容器自由地在客户机之间交换，从而少量的会话Bean就可以服务于大量的客户机。

2、实体Bean

实体Bean代表商务上的实体，比如商务数据，应该包含与数据相关的逻辑。实体Bean是对应到数据库中的一个视图，一个实体Bean实例和底层数据库完全是一回事。因此，一个简单的实体bean实例代表一个特殊的记录。更复杂的实体bean可以代表数据库表间关联视图。

实体Bean有两种操作类型：BMP（Bean管理持久性）和CMP（容器管理持久性）。BMP是指由Bean自己来实现实体Bean的持久性，即在Bean中实现数据库操作。而CMP则是由容器实现Bean的持久性，使我们不需要在Bean内再编写数据库操作的代码。



3、消息驱动Bean

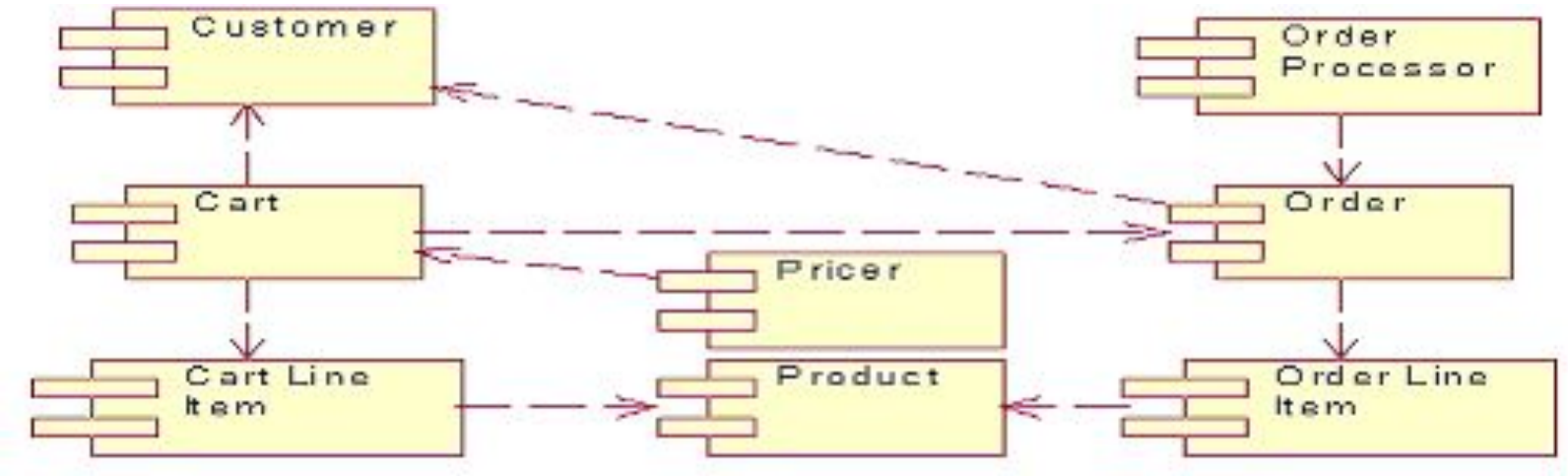
消息驱动Bean是EJB2.0新引入的一种Bean类型。它的主要目的是，通过允许容器去聚合并且管理消息驱动Bean实例，以此来提供传入JMS消息的并发处理。

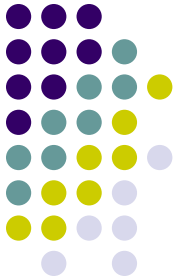
4、会话Bean与实体Bean的区别和联系

会话Bean执行应用逻辑，它们隐含地使用实体Bean作为它们操作的数据。在EJB分布式对象体系结构中，会话Bean被用做代表实际商务过程的高层接口来屏蔽底层实体Bean子系统。实体Bean是实际恒定商务实体的模型，因此它通常比会话Bean具有更高层次的重复使用。

5、EJB的应用

接下来，通过分析电子购物环节的业务逻辑层的对象模型来说明各种EJB在业务逻辑层中的不同应用。业务逻辑层包含了一系列EJB组件。首先我们将其抽象成若干个对象模型，如下图所示：





对象模型图说明：

本图首先反映了电子商务中各EJB组件之间的静态关系。

由多个购物篮条目（Cart Line Item）组成的一个购物篮（Cart）为一个顾客 (Customer)存储产品的临时选择；由多个订单条目（Order line Item）组成的一个订单（Order）为一个顾客存储产品的永久选择。购物篮能将自身转换为订单。

一个购物篮条目代表一个产品（Product）的临时选择，一个订单条目代表一个产品的永久选择。

估价器（Pricer）在顾客查看购物篮时计算购物篮的价格，并且在顾客最终生成订单时计算订单的价格。

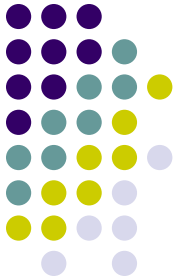
订单处理器（Order Processor）为订单验证信用卡，发送E-mail确认，并标识为永久。

同时从本图中也可以了解一个电子购物的过程：

首先，在购物时顾客把自己感兴趣的产品放入购物篮中，同时由估价器对购物篮进行及时估价。

然后，顾客在确认购买后，购物篮能自动生成订单。再由估价器计算出订单的价格。

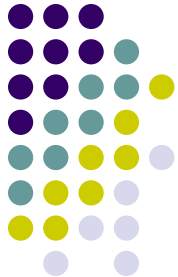
接着，由订单处理器验证顾客信用卡的合法性，在交易完成后为顾客发送E-mail确认交易成功，并将本交易标识为永久。



对象使用各类EJB实现：

首先，顾客、订单、产品、订单条目这几个对象是永久性、持续性对象，例如，顾客信息、产品信息都需要存入数据库，并且在适当的时候从数据库中读取。所以，这几项都需要用实体Bean来实现。

其次，购物篮和购物篮条目只在顾客购物的过程中起作用，所以不是永久性的，而且每一个购物篮都对应于一个特定的顾客，对应于若干条特定的购物车条目，因此购物篮和购物篮条目用状态会话Bean来充当最合适不过。



然后，估价器的作用是计算出购物篮和订单的价格，它并没有和特定的顾客绑定，可以作用于任意的购物篮，而且也不是永久对象，因此估价器可以用一个无状态会话Bean来充当。

最后，订单处理器是一个特殊的对象，它通过顾客所要求的不同的付款方式产生不同的订单，也就是说，它是由不同的付款方式来驱动的。所以在这里用消息驱动Bean是最恰当的。

通过上面的分析，我们清楚的了解到不同类型的EJB在实际应用中如何发挥自己的作用。

EJB的开发

- 无状态会话Bean的开发
- 有状态会话Bean的开发
- CMP实体Bean的开发
- BMP实体Bean的开发
- 消息驱动Bean的开发