中间件考试参考

**第一章 overview of middleware**

**1.了解中间件的背景**

答：互联网促使分布式系统和网络应用的诞生，中间件就是伴随网络技术的产生、发展而兴起的，可以说没有网络就没有现代意义上的中间件。需要通过中间件解决异构分布网络环境DCE下软件系统的通信、互操作、协同、事务、安全等共性问题，提高异构分布网络环境下软件系统的Interoperability、Portability、Scalability、

Availability等问题。

**2.什么是中间件（in english）**

答：Middleware is computer software that connects software components and applications.

The software consists of a set of enabling services that allow multiple processes running

on one or more machines to interact across a network. This technology evolved to

provide for interoperability in support of the move to coherent distributed architectures,

which are used most often to support and simplify complex, distributed applications.

It includes web servers, transaction monitors, and messaging-and-queueing software.

**中间件是一类连接软件组件和应用的计算机软件，它包括一组服务，**

**以便于运行一台或多台机器上的多个软件通过网络进行交互。**

**该技术所提供的互操作性，推动了一致分布式体系架构的演进。**

**该架构通常用于支持分布式应用程序并简化其复杂度，它包括web服务器、**

**事务监控器和消息队列软件。**

**3.IDC的中间件分类**

答：可以分为六类：

1）数据访问中间件(UDA) universal data access

**数据访问中间件是为了建立数据应用资源互操作的模式，对异构环境下的数据库实现联接或文件系统实现联接的中间件。这类中间件产品：ODBC、JDBC、JDO、ADO等数据标准接口，大都基于SQL语句，采用同步通信方式。**

2）远程过程调用中间件(RPC) Remote Procedure Call

**远程过程调用是一种广泛使用的分布式应用程序处理方法。一个应用程序使用RPC来远程执行一个位于不同地址空间的过程，从效果上看和执行本地调用相同。**

3）消息中间件(MOM) Message Oriented Middleware

**消息中间件是利用高效可靠的消息传递机制进行平台无关的数据交流，并基于数据通信来进行分布式系统的集成。MOM中间件产品：IBM的MQSeries、BEA的MessageQ、ActiveMQ**

4）交易中间件(TPM) Transaction Processing Monitors

**专门针对联机交易处理系统设计，为期提供支持大规模事务处理的可靠运行环境。事务处理监控有以下功能：进程管理，包括启动server进程、为其分配任务、监控其执行并对负载进行平衡。**

**事务管理，即保证在其监控下的事务处理的一致性、独立性和持久性。**

**通讯管理，为client和server之间提供了多种通讯机制 。广播、订阅发布等**

5）对象中间件(ORB) Object Request Broker

**对象中间件提供一个标准的构建框架，能使不同厂家的软件通过不同的地址空间、网络和操作系统实现交互访问。**

6）应用服务器(AS) Application Server

**4.中间件的特点及优势**

答：特点：满足大量应用的需要，运行于多种硬件、数据库及操作系统平台，支持分布式计算，支持互操作性Interoperability，支持标准协议 HTTP SOAP SMTP ……，支持标准的接口(组件接口) JMS

优势：1. 应用开发：业务逻辑程序和应用逻辑程序占总程序量的30%，基础程序占70%。使用中间件可以节省25%--60%应用开发。

2. 系统运行：没有使用中间件的应用系统，其初期的资金及运行费用的投入要比同规模的使用中间件的应用系统多一倍。

3. 开发周期：基础软件的开发是一件耗时的工作，使用标准商业中间件则可缩短开发周期50%--70%。

4. 减少项目开发风险：没有使用标准商业中间件的关键应用系统开发项目的失败率高于90%。

5. 合理运用资金：例如投入少量资金企业可以将旧系统换成分布式系统。

6. 应用集成：依靠标准的中间件可以将现有的应用、新的应用和购买的商务组件融合在一起进行应用集成。

7. 系统维护：方便、简单更省钱。

8. 质量： 标准中间件的规范化模块可以有效地保证应用系统的质量减少新旧系统维护开支。

9. 技术革新：大中间件厂商(IBM、Oracle)提供最新技术。

10. 增加产品的吸引力：更容易增添新的表现形式和新的服务项目。

**5.中间件在应用软件开发中的作用。举例说明**

答：这里以JDBC为例，JDBC（Java Data Base Connectivity,java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系数据库提供统一访问，它由一组用Java语言编写的类和接口组成。JDBC提供了一种基准，据此可以构建更高级的工具和接口，使数据库开发人员能够编写数据库应用程序。有了JDBC，向各种关系数据发送SQL语句就是一件很容易的事。换言之，有了JDBC API，就不必为访问Sybase数据库专门写一个程序，为访问Oracle数据库又专门写一个程序，或为访问Informix数据库又编写另一个程序等等，程序员只需用JDBC API写一个程序就够了，它可向相应数据库发送SQL调用。同时，将Java语言和JDBC结合起来使程序员不必为不同的平台编写不同的应用程序，只须写一遍程序就可以让它在任何平台上运行，这也是Java语言“编写一次，处处运行”的优势。

**第二章 AS与JavaEE**

**1.什么是应用服务器（in english）**

答：An application server’s fundamental job is to provide its clients with access to what is commonly called business logic, which generates dynamic content; that is, it’s code that transforms data to provide the specialized functionality offered by a business, service, or application. An application server’s clients are often applications themselves, and can include web servers and other application servers. Communication between the application server and its clients might take the form of HTTP messages, but that is not required as it is for communication between web servers and their clients. Many other protocols are popular, including the variants of CGI.

**2.为什么Application Server是中间件软件**

答：因为应用服务器是指通过各种协议把商业逻辑曝露给客户端的程序。它提供了访问商业逻辑的途径以供客户端应用程序使用。通过它能够实现运行一台或多台机器上的多个软件通过网络进行交互。

**3.多层应用体系的优势（in english）**

答：

**C/S体系结构特点：**

1. **无论是客户端还是服务端都需要特定的软件支持。**

**没能提供用户期望的开放环境，适用于Intranet。**

1. **服务器端运行负荷较轻。**
2. **数据的存储管理功能较为透明。**
3. **C/S体系结构的劣势是高昂的维护成本且投资大。**

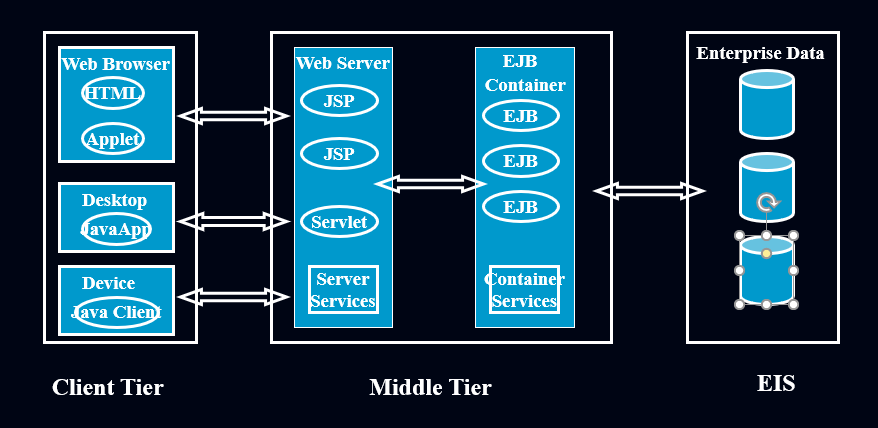
**B/S体系结构特点：**

1. **简化了客户端的工作，瘦客户端结构。**
2. **对数据库的访问和应用程序的执行将在Web Server上完成。**
3. **把技术维护人员从繁重的维护升级工作中解脱出来。**

**4.什么是JavaEE（in english）**

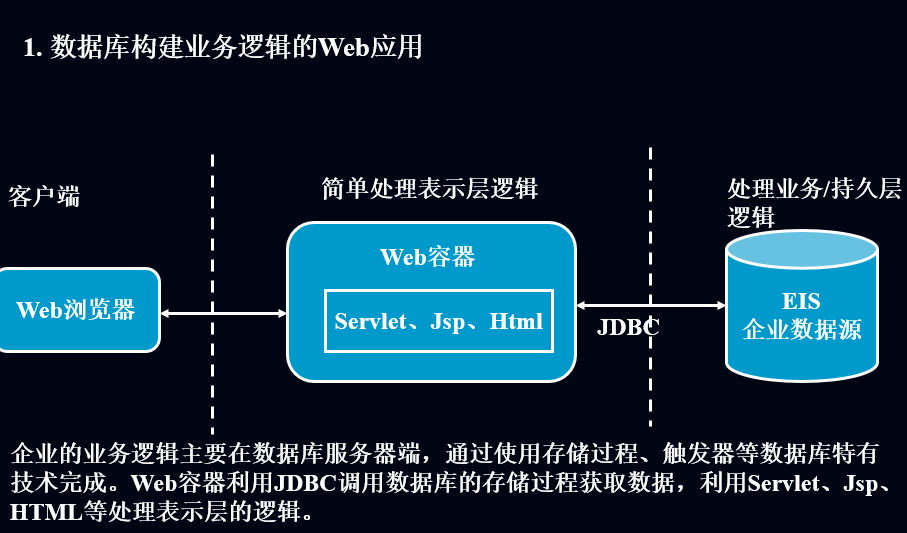
答：Open and standard based platform for developing, deploying and managing n-tier, Web-enabled, server-centric, component-based enterprise applications.

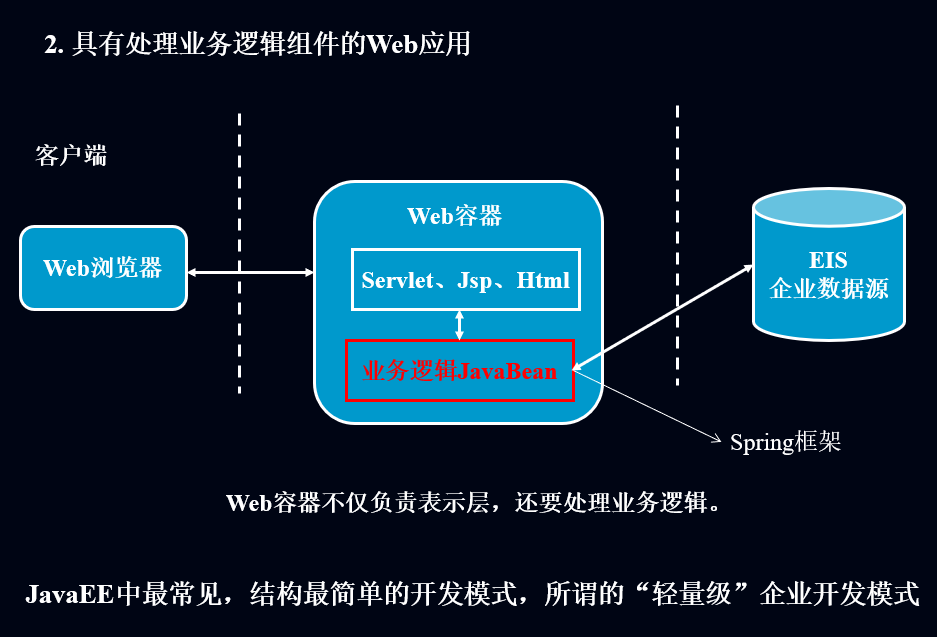
**5. JavaEE and N-Tier Architecture**

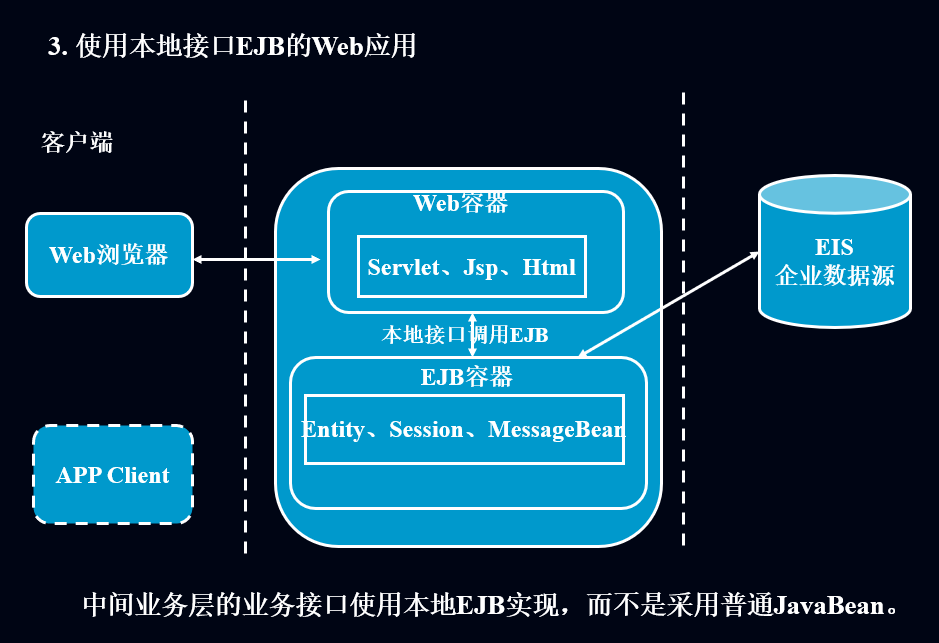
答：

**6.JavaEE开发模式（分布式与非分布式）**

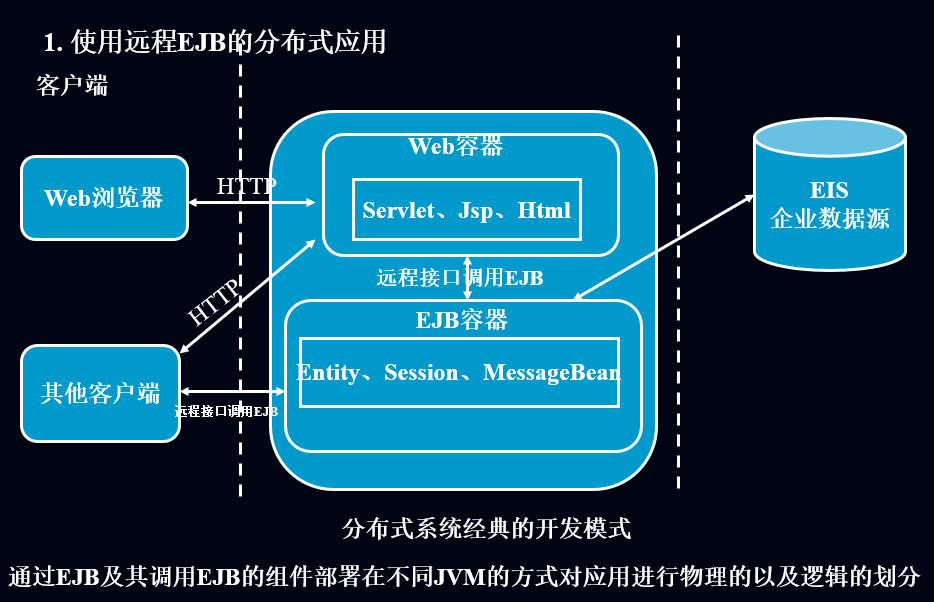
答：非分布式







分布式：



**第三章 UDA and Hibernate**

**1.JDBC驱动类型**

答：Oracle OCI Driver and Oracle Thin Driver

**2.DAO设计模式（in english）**

答：DAO=DataAccessor + Domain Object

**3.ORM/Hibernate**

答：Hibernate是一种Java语言下的对象关系映射(ORM)解决方案。为面向对象的领域模型到传统的关系型数据库的映射，提供了一个使用方便的框架。

Hibernate is an object-relational mapping (ORM) library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional relational database.

**第四章 MOM**

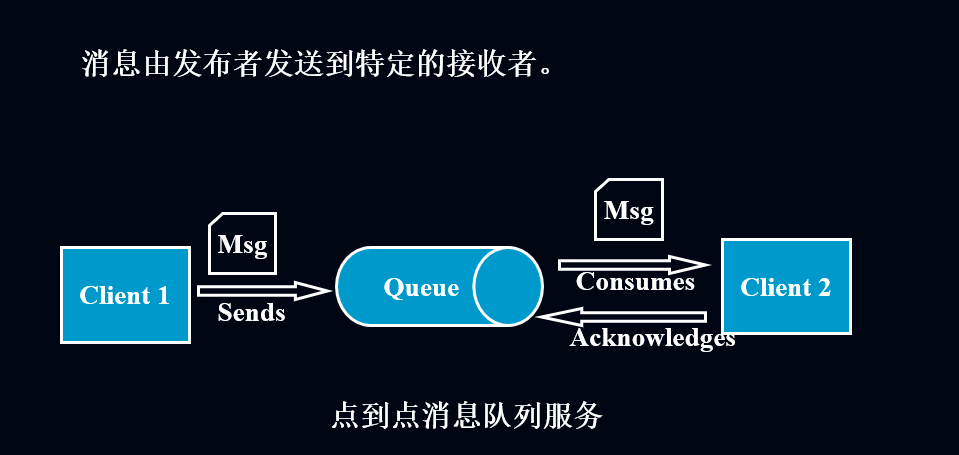
**1.什么是Message-Oriented Middleware（in english）**

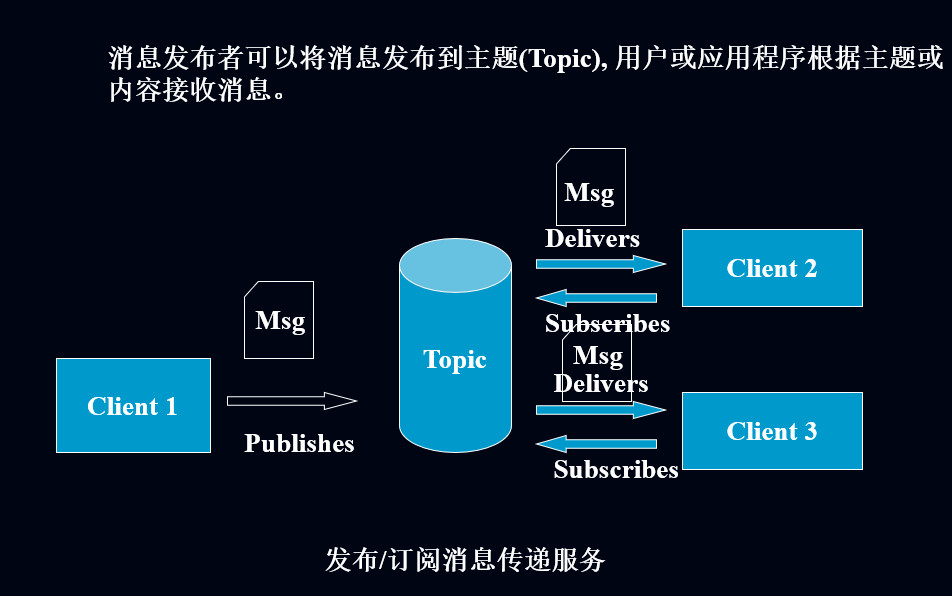
答：Message-oriented middleware (MOM) is software infrastructure supporting sending and receiving messages between distributed systems. MOM allows application modules to be distributed over heterogeneous platforms and reduces the complexity of developing applications that span multiple operating systems and network protocols. The middleware creates a distributed communications layer that insulates the application developer from the details of the various operating systems and network interfaces. APIs that extend across diverse platforms and networks are typically provided by MOM.

**2.企业为什么需要MOM**

答：1. 利用MQ (message queue)实现企业/部门间的数据交换2. 实现企业级大规模数据传送3. 分布对象技术ORB局限性

**3.Queue和Topic模式**

答： 



**4.同步和异步通信：UDA\RPC\ORB\TMP\MOM\RMI**

答：

**5.MOM特点**

答：1) 要通信的应用程序可以运行在不同时间；

2) 对于应用间的结构没有限制；

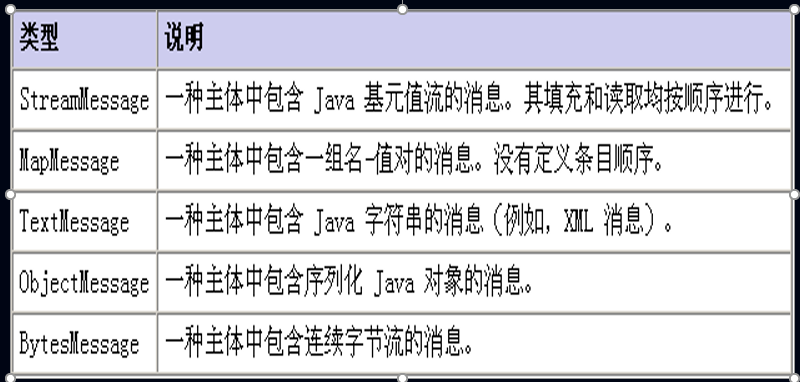
3) 对于应用程序来说，底层的环境差异被屏蔽掉

**第五章 JMS和ActiveMQ**

**1.什么是JMS（in english）**

答：The Java Message Service (JMS) API is a Java Message Oriented Middleware (MOM) API for sending messages between two or more clients.It is an implementation to handle the Producer-consumer problem. JMS is a part of the Java Platform, Enterprise Edition, and is defined by a specification developed under the Java Community Process as JSR 914.It is a messaging standard that allows application components based on the Java Enterprise Edition (Java EE) to create, send, receive, and read messages. It allows the communication between different components of a distributed application to be loosely coupled, reliable, and asynchronous

**2.** **JMS的消息类型**

答：

**3.被管理的对象**

答：ConnectionFactory： 客户端用来创建同JMS提供者之间的连接的对象

Destination ：这个对象是客户端用来指明消息被发送的目的地以及客户端接收消息的来源。

**4.** **JMS客户端编程**

答：1. 使用JNDI查找ConnectionFactory对象。//事先通过管理工具绑定到JNDI

2. 使用JNDI查找一个或者多个Destination对象。 //事先通过管理工具绑定到JNDI

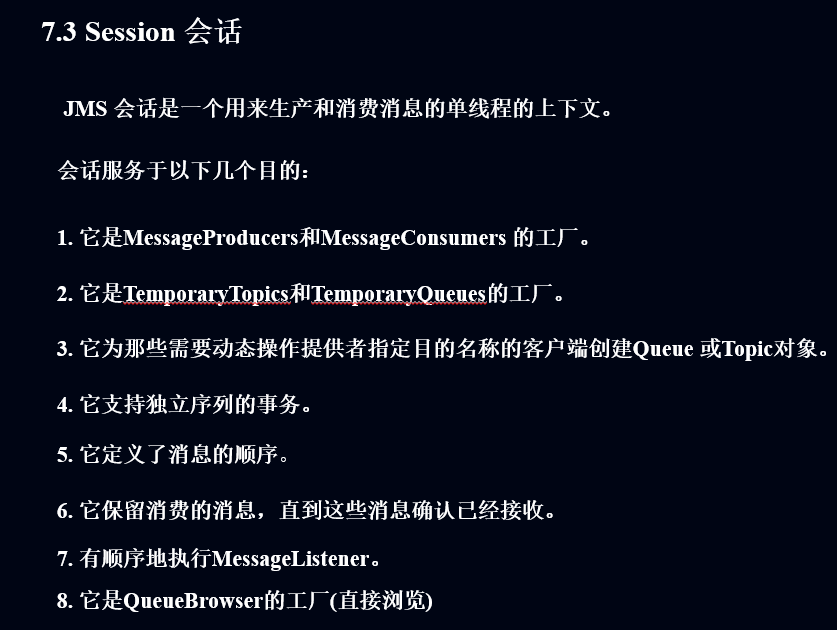
3. 使用ConnectionFactory创建一个JMS连接。

4. 使用连接创建一个或者多个JMS Sessions 。

5. 使用Session and Destinations 创建所需的MessageProducers 和MessageConsumers。

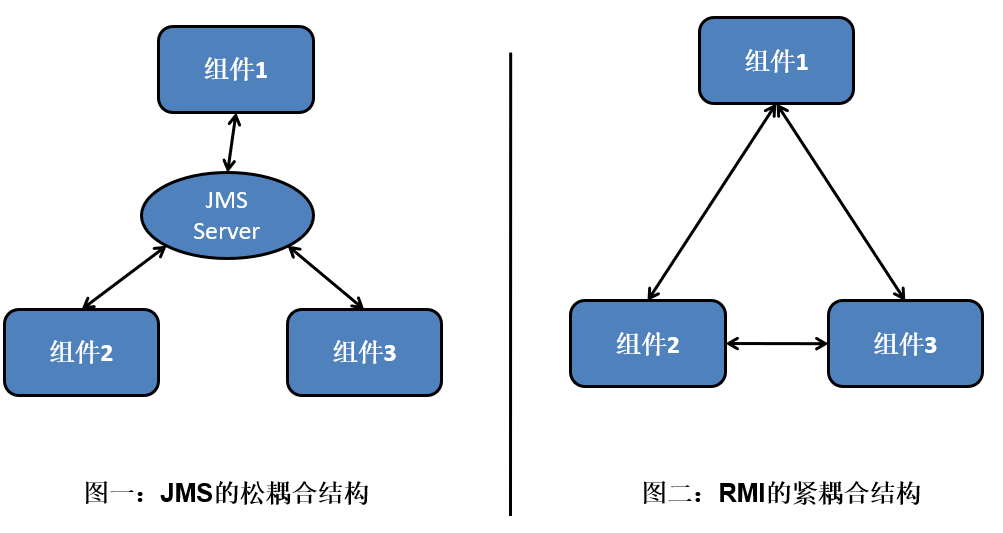
6. 告知Connection 开始传送消息。

**5. JMS中的Session**

答：

**6. JMS与RMI**

答：JMS与RMI的区别：1. 通信方式不同：JMS支持同步和异步消息处理模式。RMI基于请求/应答的同步通信模式2. 耦合度不同：JMS建立了一个松散耦合的通信框架。图一RMI是紧耦合结构。图二



3. 需求对象获得方式：

JMS：对象从网络的某个节点移动到另一个节点。

RMI：对象是绑定在本地JVM 中，只有参数和返回值是通过网络传送。

**7. ActiveMQ是如何发送大文件的**

答：1. 通过JMS Streams进行文件传输

2. 通过文件分割/合并传输文件3. 通过Blob Messages传输文件

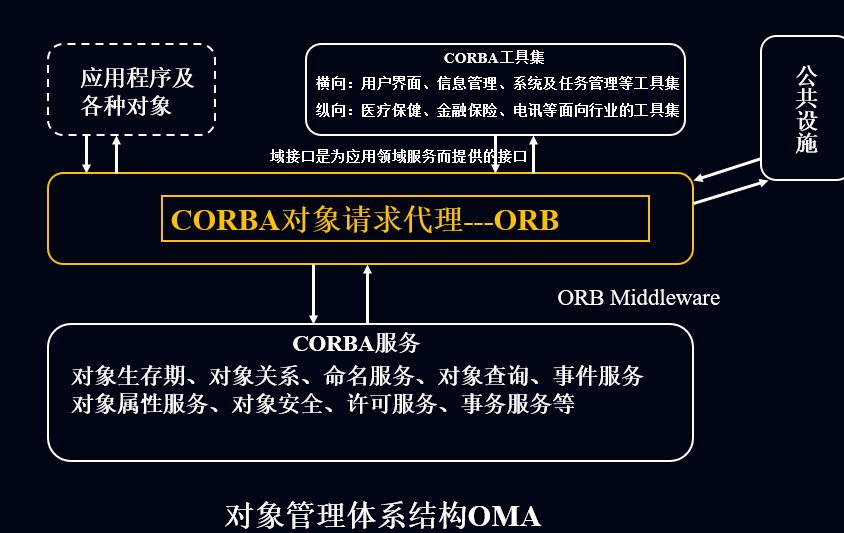
**第六章 CORBA**

**1. 什么是CORBA和ORB(in English)**

答：OMG：Object Management Group

CORBA is the acronym for Common Object Request Broker Architecture™, OMG®'s open, vendor-independent architecture and infrastructure that computer applications use to work together over networks. Using the standard protocol IIOP, a CORBA-based program from any vendor, on almost any computer, operating system, programming language, and network, can interoperate with a CORBA-based program from the same or another vendor, on almost any other computer, operating system, programming language, and network.

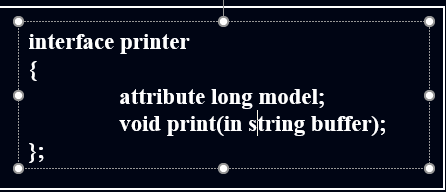
**2. OMA的CORBA体系结构**

答：

**3. 对象模型和参考模型**

答：对象模型：定义如何描述分布式异构环境中的对象。参考模型：描述对象之间的交互。

在OMA对象模型中，对象是一个被封装的实体，它具有一个不可改变的标识，并能给客户用户提供一个或多个服务。





**4. 什么是POA(in English)**

答：POA is an object manager, similar to the EJB container, or Spring's IoC container, and is primarily used to create, activate, locate, and recycle objects

**5.** **CORBA的命名服务**

答：

**第七章 JavaRMI**

**1. 会定义一个远程接口**

答：**import java.rmi.Remote;**

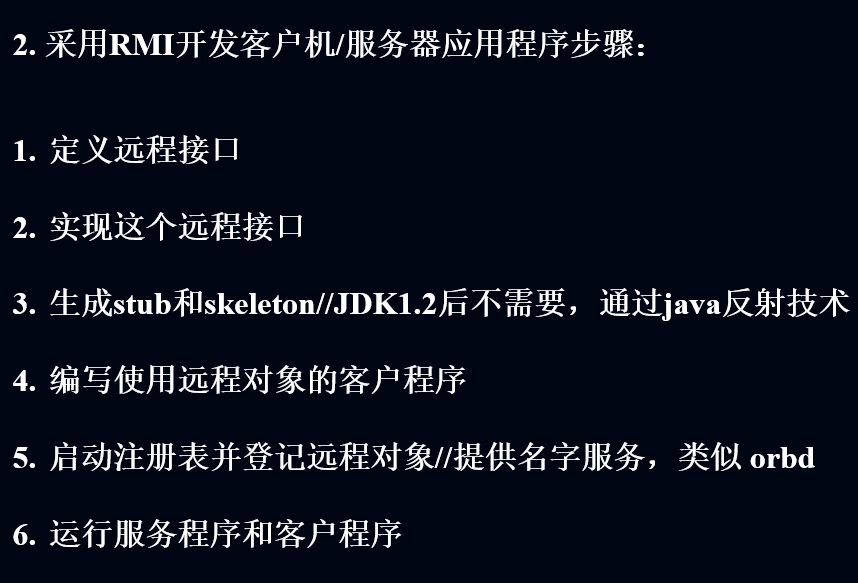
**import java.rmi.RemoteException;**

**public interface InterfaceService extends Remote {**

**Object service(Object obj) throws RemoteException;**

**}**

**2. JavaRMI 的应用**

答：

1. 定义远程接口(服务接口)

import java.rmi.Remote;

import java.rmi.RemoteException;

/\*\*

\* RMI调用对象接口定义

\* @author 王华东

\*/

public interface InterfaceService extends Remote {

/\*\*远程服务对象所必须实现的方法

\* 所有的远程调用的方法,必须声明throws RemoteException

\* \*/

Object service(Object obj) throws RemoteException;

}

2. 实现远程接口

/\*\*

\* RMI服务对象接口实现, 此对象将被RMI服务器导出做为远程服务对象

\* @author 王华东

\*/

// 远程接口必须继承java.rmi.server.RemoteServer或它的子类

public class ImpService extends java.rmi.server.UnicastRemoteObject

implements InterfaceService {

public ImpService() throws java.rmi.RemoteException {

super();//

}

/\*\*

\* 客户端将要调用的方法示例

\*/

public Object service(Object obj) {

System.out.println("RMI客户机请求: "+obj);

return System.currentTimeMillis()/1000 + ": "

+ obj.toString();

}

}

3. 启动RMI服务器

import java.rmi.\*;

import java.rmi.registry.\*;

/\*\*

\* 启动RMI服务器,导出服务对象

\* @author 王华东

\*/

public class StartRMIServer {

/\*\*启动方法\*/

public static void main(String[] args) {

int listerPort=9911; //设置RMI监听器在9911端口

String serverIP="localhost"; //监听的IP

String serviceObjName="service"; //要导出的服务对象名字

try {

LocateRegistry.createRegistry(listerPort); //设置RMI服务器监听端口

//ImpService.setLog(System.out); //设置日志对象,打印到控制台

ImpService remoteObj = new ImpService(); //创建导出的对象, 绑定服务

Naming.rebind("rmi://"+serverIP+":"+listerPort+"/"+serviceObjName, remoteObj);

System.out.println("RMI启动在"+serverIP+": "+listerPort+" 服务名为: "+serviceObjName);

} catch (Exception e) {

System.err.println("Server exception: " + e.toString());

e.printStackTrace();

}

}

}

4. 客户机代码

import java.rmi.\*;

/\*\*

\* RMI调用客户端

\* @author 王华东

\*/

public class RmiClient {

public static void main(String[] args) {

int listerPort=9911;//设置RMI监听器在9911端口,1099是默认端口

String serverIP="localhost";//监听的IP

String serviceObjName="service";//要导出的服务对象名字

try {

//查找服务器上的服务对象

InterfaceService stub = (InterfaceService)

Naming.lookup("rmi://"+serverIP+":"+listerPort+"/"+serviceObjName);

for(int i=0;i<10;i++){

//调用对象的服务方法

Object response = stub.service("请说话....."+i);

System.out.println("RMI服务器应答:" + response.toString());

Thread.sleep(1000);

}

} catch (Exception e) {

System.err.println("Client exception: " + e.toString());

e.printStackTrace();

} } }

**第八章 XML**

**1. XML 与 HTML**

答：XML是eXtensible Markup Language的缩写； 是一种类似于HTML的标记语言； 是用来描述数据的； XML的标记不是在XML中预定义的，你必须定义自己的标记； XML使用文档类型定义（DTD）或者模式（Schema）来描述数据； XML使用DTD或者Schema后就是自描述的语言； XML与HTML都源于SGML The Standard Generalized Markup Language。

XML是用来存放数据。

XML不是HTML的替代品，XML和HTML是两种不同用途的语言。

XML是被设计用来描述数据的，重点是：什么是数据，如何存放数据。

HTML是被设计用来显示数据的，重点是：显示数据以及如何显示数据更好上面。

HTML是与显示信息相关的, XML则是与描述信息相关的。

**2. 理解DOM 和SAX接口**

答：DOM(Document Object Model 文档对象模型)，在应用程序中，基于DOM的

XML解析器将XML文档转换成对象模型的集合(DOM树)，通过对这个对象

模型的操作，来实现对XML文档数据的操作。

优点：  
    a、由于整棵树在内存中，因此可以对xml文档随机访问  
    b、可以对xml文档进行修改操作  
    c、较sax，dom使用也更简单。  
缺点：  
    a、整个文档必须一次性解析完  
    b、由于整个文档都需要载入内存，对于大文档成本高

SAX，基于事件驱动的，因此无需将整个文档载入内存，使用者只需要监听自己感兴趣的事件即可。

优点：  
    a、无需将整个xml文档载入内存，因此消耗内存少  
    b、可以注册多个ContentHandler  
缺点：  
    a、不能随机的访问xml中的节点  
    b、不能修改文档

**第九章 EJB**

**1. 什么是EJB(in english)**

答：Ejbs are sun's JavaEE server-side component model, and the design goal and core application is to deploy distributed applications，Javabean is reusable components of JAVA.It must be concrete and public, and has a non-parametric constructor.

**2. JNDI(in english)**

答：JNDI refers to accessing different naming and directory services by providing a unified Java API

**3. JNDI的应用**

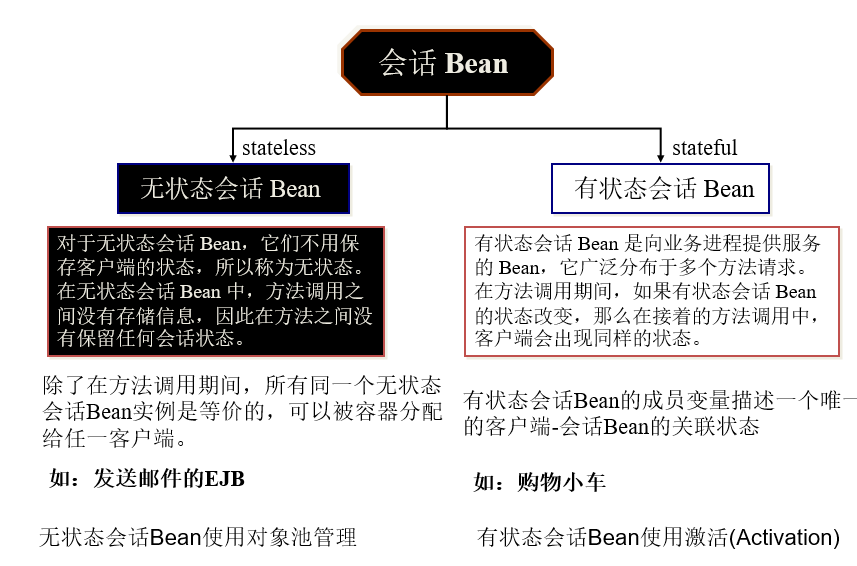
答：要通过JNDI进行资源访问，我们必须设置初始化上下文的参数，主要是

设置JNDI驱动的类名(java.naming.factory.initial)和提供命名服务的URL(java.naming.provider.url)。

java.naming.factory.initial的值因提供JNDI服务器的不同而不同，java.naming.provider.url的值包括提供命名服务的主机地址和端口号。

**4. 有状态会话Bean和无状态会话Bean**

答：



**5. 会话Bean生命周期**

答：