中间件：

分布式🡪 不同系统平台🡪 需要交换信息🡪中间件

中间件位于操作系统和应用之间，相当于提供一个服务

DCE 分布式计算环境

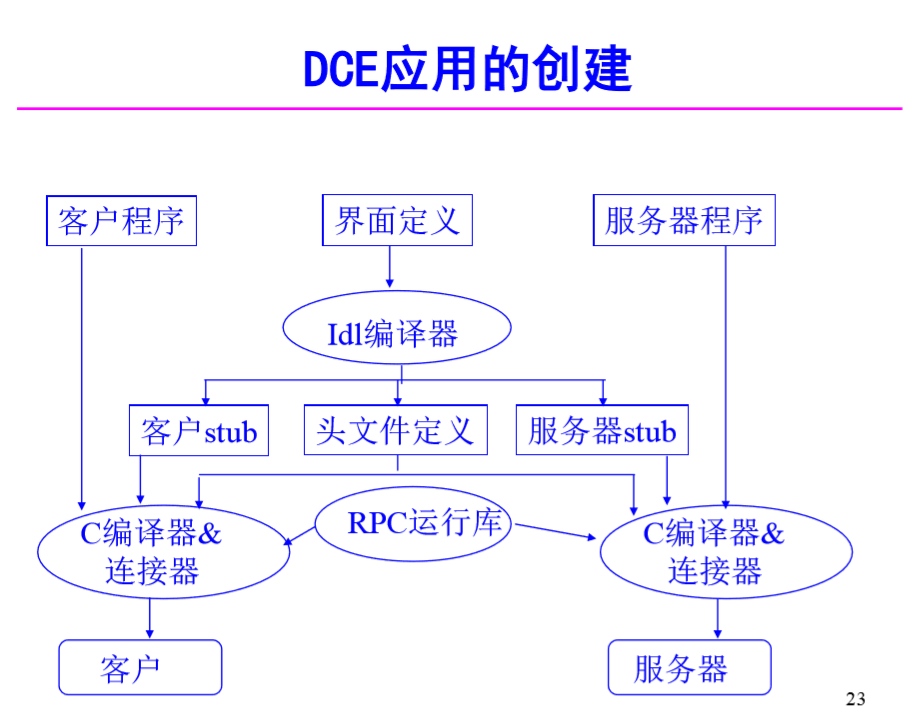
2类：基本服务和扩充服务

基本服务：线程（并行）、RPC（通讯工具，统一格式数据）、目录（服务器：服务注册、用户：目录查找，相当于中介，提供列表）、安全（身份认证）

扩充性服务：分布式文件（文件访问透明性）和时间（时间一致性）

DCE应用的三个文件：

1. 界面定义文件：定义服务器所提供远程过程调用的界面
2. 客户程序：定义用户界面，对远程过程的调用和客户方处理功能
3. 服务器程序：提供远程过程调用实现



RM-ODP 开放式分布处理标准模型

组成：

1. 观点
2. 透明性

观点：

把一个分布式系统分成若干个不同的侧面进行描述

分类：企业观点、信息观点、计算观点、工程观点、技术观点

透明性：

透明性屏蔽了由系统分布带来的复杂性（即用户和开发者不用考虑环境带来的影响）

分类：

1. 访问透明性
2. 位置透明性
3. 迁移透明性
4. 失败透明性
5. 重定位透明性
6. 复制透明性
7. 持久透明性
8. 事务处理透明性

透明性决定了用户和开发者能看见系统的什么部分和不能看见系统的什么部分

功能组成：

1. 管理功能（结点管理、类结构等）
2. 协作功能（项目、检查点、组等）
3. 仓库功能（存储、信息组织结构）
4. 安全功能（访问控制等）

ODP独立于其他标准，但又是其他标准的标准，规定了开放分布式处理领域的其他标准必须遵循的参考模型，比如说CORBA和DCOM

OMA 对象管理结构

面向对象应用的一个公共框架

核心：CORBA规范

组成

1. 对象模型：定义如何描述分布式异构环境中的对象
   1. 组成：
      1. 唯一标识
      2. 一个或多个服务
   2. 访问方式:提供向对象的请求，每个对象的实现和位置，对客户都是透明的
2. 参考模型：描述对象之间的交互
   1. 定义ORB（对象总线）：对象公用的通信总线
   2. 进出ORB的:
      1. 对象服务：命名服务、持久性服务、生存周期服务、事务服务、安全服务
      2. 公共设施
      3. 应用界面
      4. 域界面

CORBA：公共对象请求代理体系结构

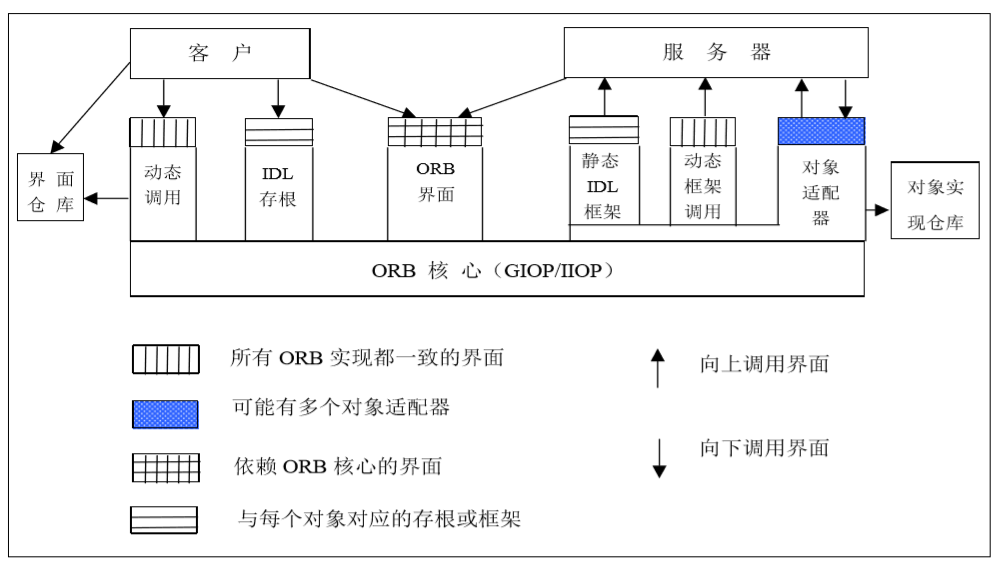
OMG提出的应用软件体系结构和对象技术规范

核心是一套标准的语言、接口和协议

用于支持 异构分布应用程序的互操作性 及 独立于平台和编程语言的对象重用

内容：

1. ORB核心：用户发出请求给对象，把结果返回用户
   1. 特征：交互透明性
      1. 对象位置
      2. 对象实现
      3. 对象执行状态
      4. 对象通信机制
      5. 数据标识
2. 核心服务（对象服务）：ORB的对象服务
3. IDL语言：定义CORBA接口
4. 存根和框架：
   1. 存根：代表用户发送请求（客户端）
   2. 框架：把请求给CORBA对象（服务器）
5. 动态调用
6. 接口仓库和实现仓库：lib
7. 对象适配器：ORB对对象的操作
8. ORB间互操作
   1. GIOP 解决不同厂商的ORB不能互操作
      1. 组成：
         1. 公共数据表示：数据
         2. GIOP消息格式：请求类型
         3. 传输层假设：适应各种传输协议
      2. GIOP是一种抽象协议，使用时必须映射



优点：异构能力强，多家应用商提供CORBA服务

不足和缺陷：

1. 没有能独立提供不同网络所有系统实现的供应商
2. 不同CORBA实现之间缺乏互操作性
3. 缺乏多线程环境规范，C/C++不兼容
4. CORBA过于复杂

第四章

客户：图形化界面

服务器：处理后台持久性数据

中间件：处理客户和服务器的交互的介质

问题：

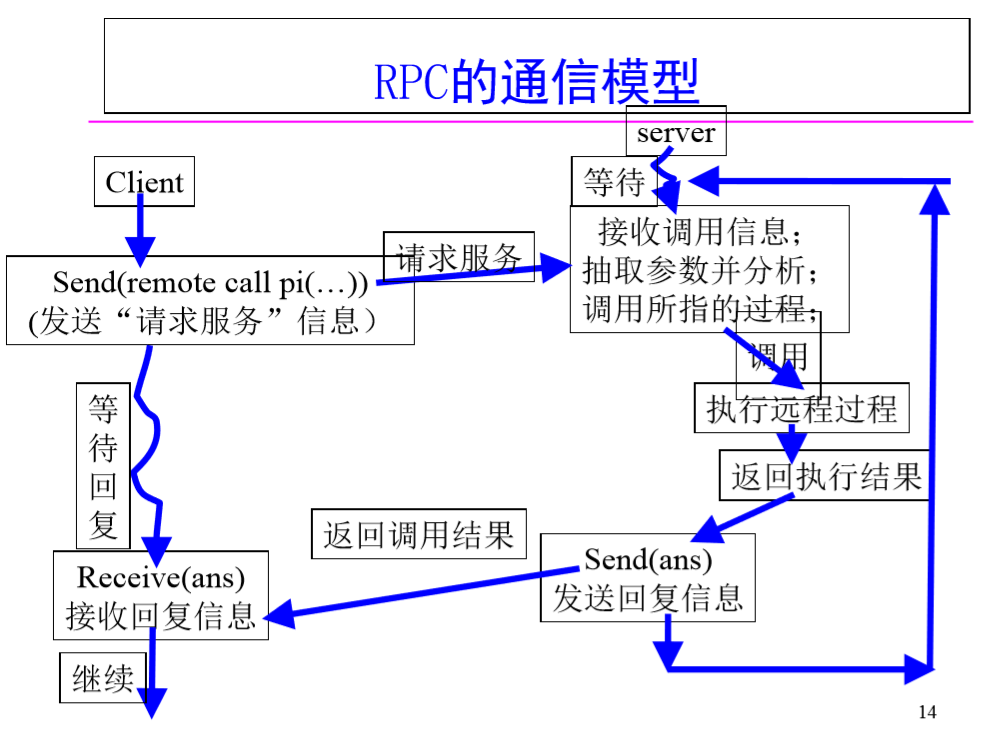
1. 客户机的性能限制
2. 安全性

解决方案：RPC分布式计算（用于服务器密集型）

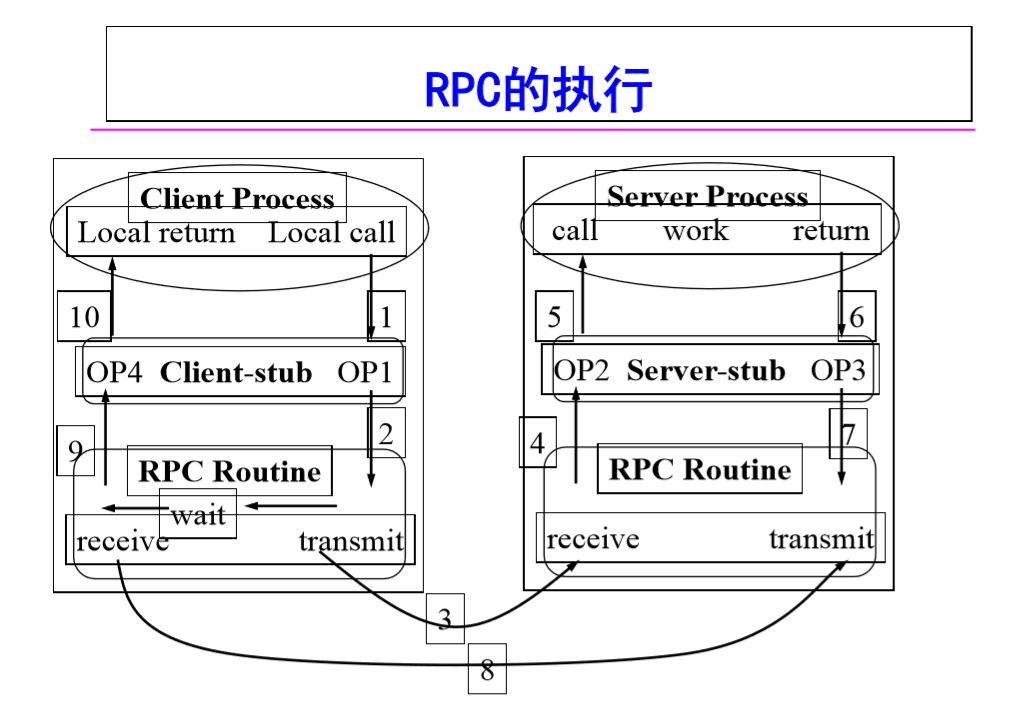
1. 客户只负责显示、捕获和验证数据
2. 中间层是支持客户机进行复杂处理的服务器层
3. 后台服务器管理持久化数据

RPC 远程过程调用

1. 概念：进程间通信IPC
2. 远程过程调用协议：通过网络从远程计算机程序上请求服务，不需要了解底层
3. 通信模型



1. 执行

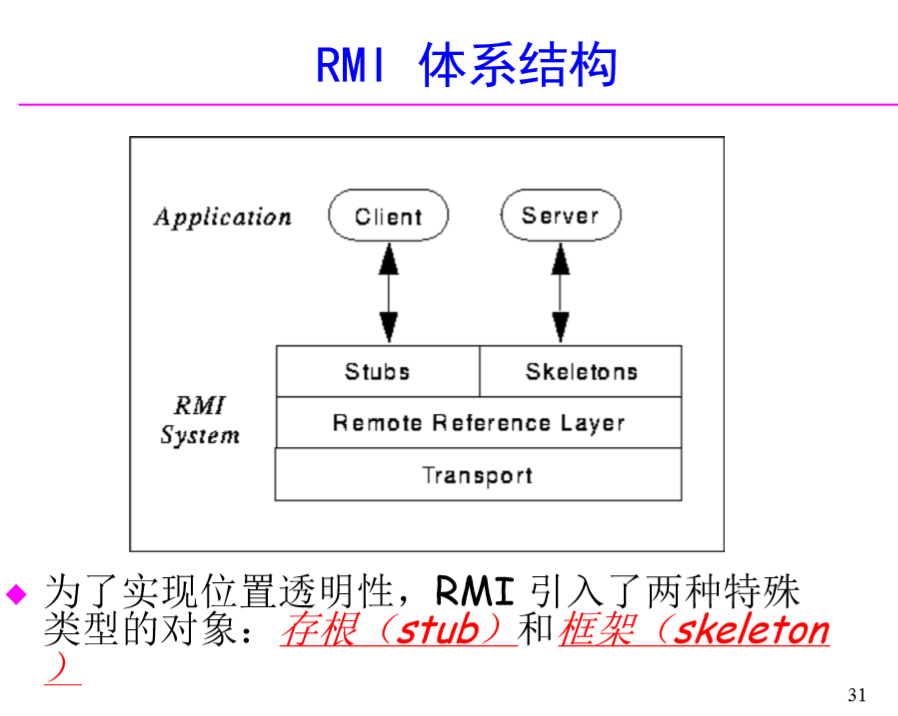


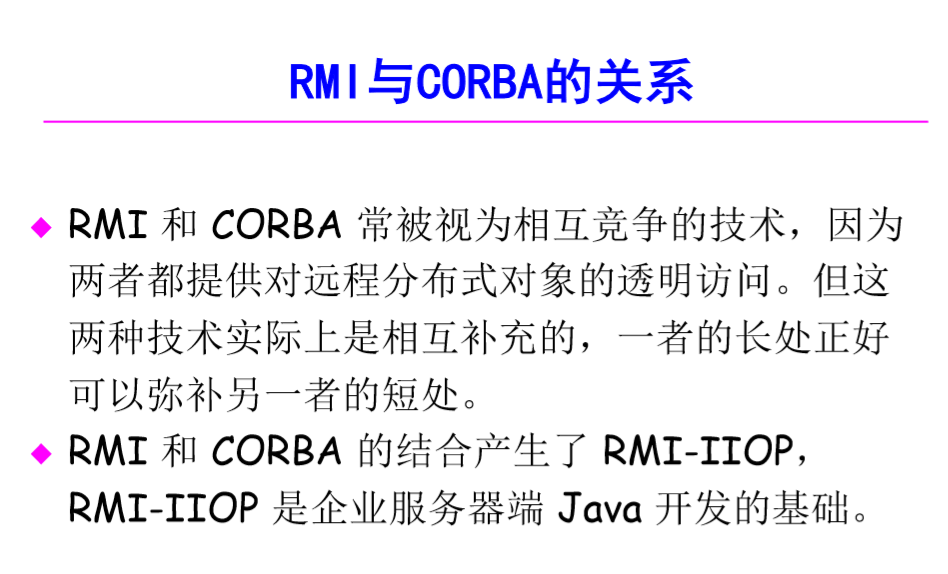
本地调用（用户进程）->操作1（用户存根）->传输（RPC例程）->服务器接受（RPC例程）->操作2（服务器存根）->服务器调用（服务器进程）->返回

1. 互斥：单个远程程序只能支持一个远程过程调用，调用完成之前会阻塞
2. 通信协议：TCP、UDP
3. 优势：
   1. 位置透明性
   2. 开发人员只用考虑功能，而不需要考虑通信和协议
   3. 可拓展3层到多层
   4. 安全性、多线程、多传输协议

Java RMI Java远程方法调用

1. 用途一种基于java的远程过程调用的应用程序编程接口，Java网络分布式系统应用的核心解决方案
2. 与RPC：Java的RPC，支持存储不同地址空间程序级对象彼此通信，实现远程对象之间的无缝远程调用
3. 特点：
   1. 简化远程接口对象的使用
   2. 使分布在不同虚拟机的对象外表和行为都和本地一样
4. 目标：
   1. 支持不同虚拟机对象的无缝远程调用
   2. 支持服务器对客户端的回调
   3. 分布式对象模型集成在java语言里，使编写尽可能简单的同时保留java的安全性
5. 体系结构



1. 存根/框架技术：
   1. 对象串行化：对象与字节流的相互转化
   2. 动态类加载
2. 位置透明性
3. 
4. 优点：
   1. 面向对象（rpc面向过程）
   2. 可移动属性
   3. 安全
   4. 便于编写
   5. 一次编译
   6. 多线程
   7. 垃圾回收

RMI和RPC

相同：大框架上相同

不同：面向对象与面向过程

CORBA和RMI

相同：面向对象、分布式计算

不同：

1. CORBA面向所有语言（IDL与平台无关），通用性强，RMI仅Java
2. CORBA复杂，RMI简单
3. RMI由容错和垃圾回收等、多线程支持更好

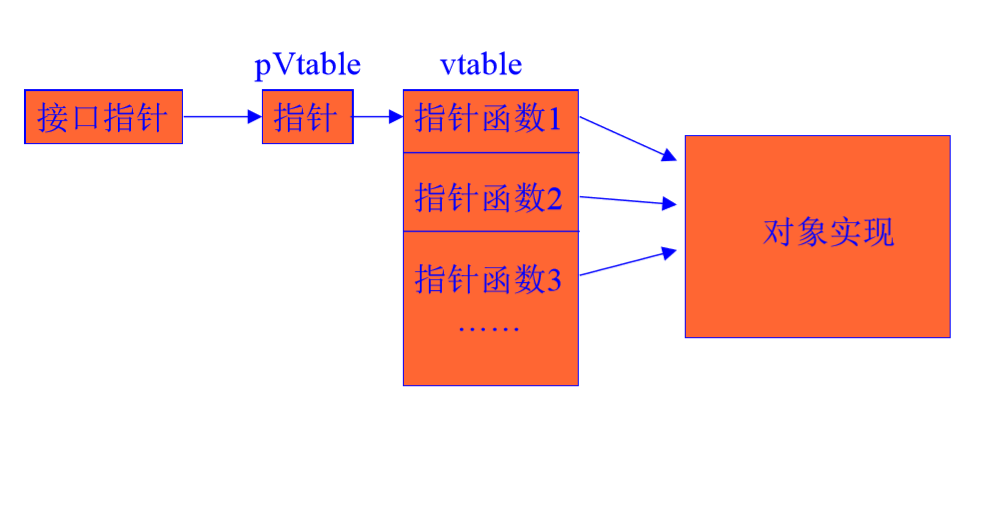
第五章

组件技术：

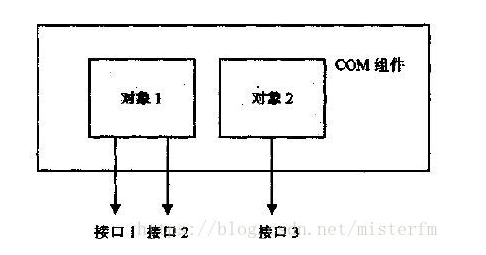
1. 组件：数据和方法的封装对象
2. 组件技术使一种优秀的软件重用技术
3. 连接建立在目标代码级别，和平台无关（可执行的二进制程序）
4. 面向对象和分布式
5. 接口由IDL定义
6. 主要技术
   1. DCOM
   2. CORBA
   3. JAVA EE
   4. .NET
   5. SOA

COM组件对象模型：

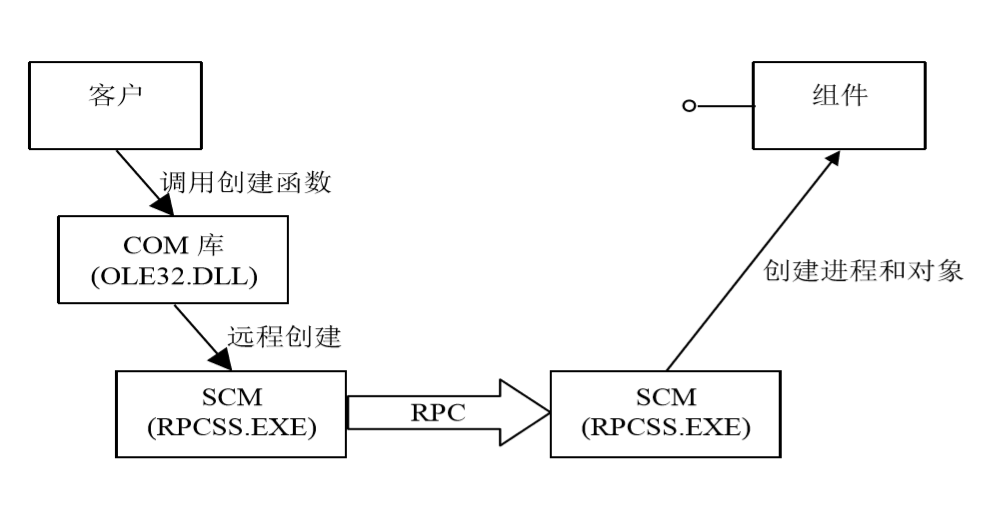
1. 概念：一种基于COM的、已经编译好的软件组件
2. COM包含一套标准API、一个标准接口集以及支持的分布式网络协议
3. 组成：
   1. 接口：客户与对象的协议，调用COM服务
      1. 接口标识 IID：GUID，全球唯一性
      2. IUnknown：所有接口都从IUnknown派生
      3. 二进制结构



* 1. 对象：实现接口，提供服务
     1. CLSID ：class id 简称，GUID，身份一致性检测
  2. 组件：对象载体，独立发布
  3. 结构图



1. DCOM
   1. 创建过程



第六章

Web服务器：

1. 提供文档显示界面
2. 处理HTTP请求
3. 分离抽象出服务器的基本执行流程，例行流程交给服务器，个性化、非标准化给用户

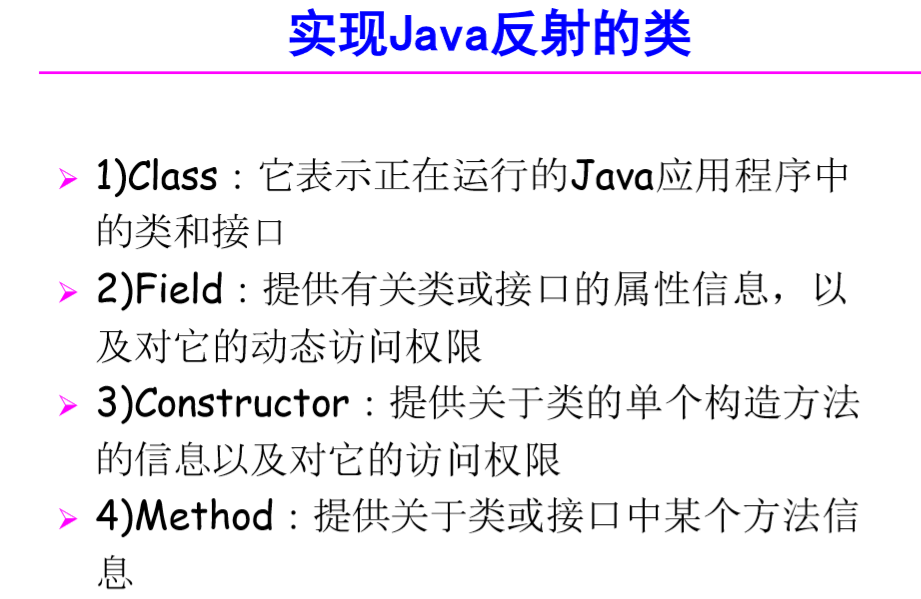
MVC

1. model
2. view
3. controller

Web 容器

1. 管理对象的生存周期、依赖关系
2. 配置，定义对象的名称、产生方式、关系等
3. 无需用户，自动产生对象

概念

1. 反射：
   1. 概念：程序可以访问、检测和修改它本身状态或行为的一种能力
   2. 实现：
2. 注解：
   1. 概念：一种代码级别的说明
3. 依赖注入：
   1. 概念：容器通过配置和上下文信息，为组件和对象注入它们之间的依赖关系
   2. 解耦合：把复杂系统分解成相互合作的对象，通过封装以后，其内部实现对外部是透明的
4. 面向切面：
   1. 将一些通用的功能如日志和核心业务逻辑分开，前者作为“切面“存在

Spring ：通过POJO对象实现复杂业务逻辑，IOC和AOP服务器容器框架，采用DI设计模式

EJB：运行在容器中，让开发者只关注Bean业务逻辑

1. Session Bean：处理某一具体的业务，只存在部分时期，分为有状态和无状态
   1. 无状态
      1. 不维护会话状态，服务任务在一个方法调用中结束
      2. 没有中间状态
      3. 每一次调用都独立于前一次
      4. 会话池设置最大会话数量
      5. @Stateless
   2. 有状态
      1. 请求一个Bean实例时，客户端得到一个会话实例，为客户端维持
      2. @Stateful
2. 消息驱动（Message Driver）Bean
   1. 异步通信
   2. 创建MDB接收异步JMS信息
   3. @MessageDriven，实现MessageListener接口
3. 实体 Bean（被JPA取代）
   1. 管理持久层的对象

共同核心设计理念：将中间件服务传递给耦合松散的POJO对象

第七章

通信局限性:

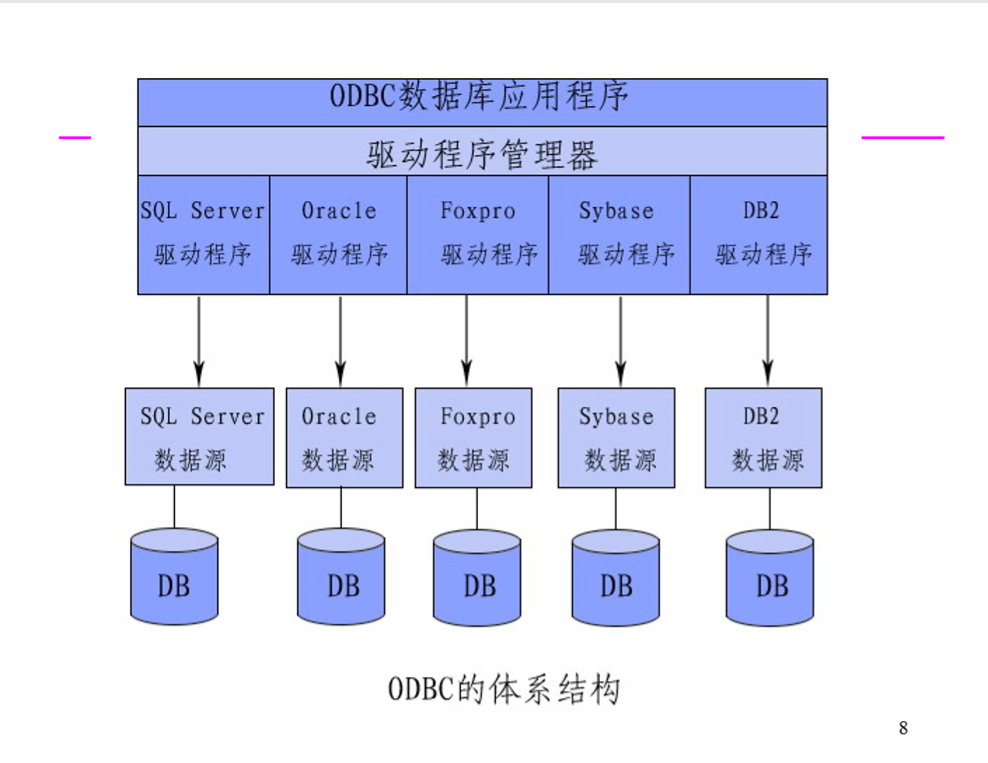
1. 同步通信
2. 异常

消息中间件：

1. 概念：分布式系统中完成消息的发送和接收的基础软件
2. 松耦合、透明

第八章

ODBC

1. 概念：用于关系或非关系数据库管理系统中存取数据的标准应用程序数据接口
2. 一组规范和一组对数据库访问的标准API
3. 结构：4部分
   1. 应用程序接口：为用户提供统一SQL接口
   2. 驱动器管理器：管理驱动器
   3. 数据库驱动器：ODBC调用
   4. 数据源：存取数据和DBMS
   5. 

JDBC

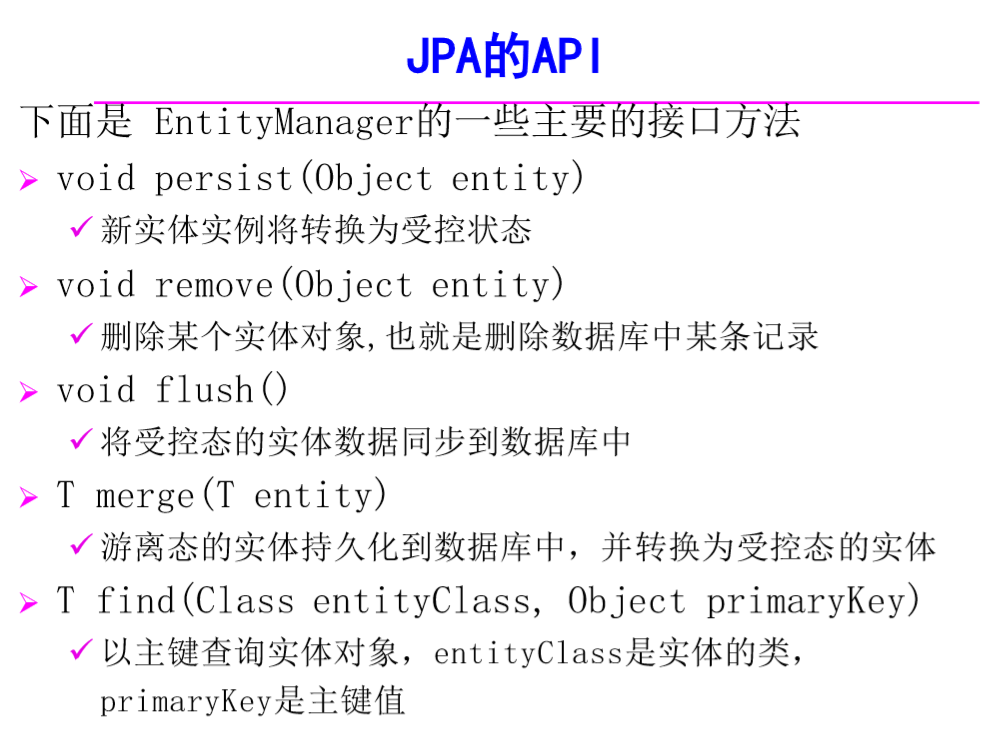
1. 解决问题：ODBC无法符合日渐复杂的数据存取应用，也无法让脚本语言使用
2. 概念：Java与数据库的接口规范
3. 和ODBC相比：
   1. 思想上沿袭ODBC：接口规范、统一SQL接口等
   2. API：数据库连接、SQL指令、结果集、数据库元数据
   3. 允许Java程序员发送SQL指令并处理结果
4. 优势：
   1. 移植性
   2. 面向对象
   3. 简单
   4. 不同数据库仅需选取不同驱动程序

ORM

1. 概念：程序中的对象通过映射自动持久化到关系数据库中
2. Hibernate：用户创建持久化类，映射到数据库表内，自动映射

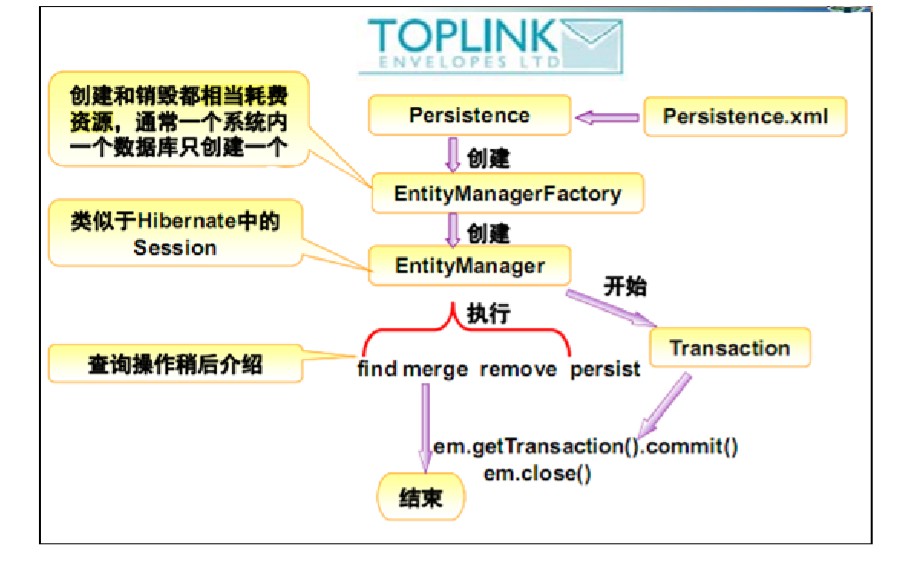
JPA Java持久化API

1. 概念：提供以pojo编程模型为持久化对象的机制，通过注解或xml描述对象—关系表的映射关系，将运行期的实体对象持久化到数据库中
2. 与ORM软件关系：JPA是hibernate的一个抽象或者可以理解为接口（JPA是规范，ORM软件是实现）
3. 优势：
   1. 简单易用，集成方便
   2. 媲美JDBC查询能力
   3. 支持面向对象的高级特性
4. 不足：
   1. 不是一个产品
   2. 仅提供JPA API
5. 内容：
   1. ORM映射元数据（描述实体对象到表的映射）
   2. API：操作实体对象



* 1. 查询语言：JPQL

1. 执行过程



第九章

事务

1. 概念：访问并可能更新数据库中各种数据项的一个程序执行单元
2. ACID：原子、一致、隔离、持久

JDBC的事务

1. 基于连接管理
2. 模式：自动提交（默认）和手动提交

分布式事务处理：

1. 用途：协助在分布式环境中跨异类的事务识别资源的事务
2. 应用程序可以将不同的活动合并成一个事务性单元
3. 两阶段提交：
   1. 一个主事务管理器负责充当分布式事务协调器的角色，协调者可以看作事务的发起者
   2. 过程：
      1. 第一阶段：
         1. 协调者写一条日志记录，对所有参与者发消息prepare T，询问这些参与者（包括自身）是否能提交
         2. 参与者收到信息，根据自身情况进行预处理：
            1. 若参与者能提交该事务，记录日志，返回协调者ready T，自身进入预提交
            2. 不能提交，记录日志，返回一个not commit T，撤销自身数据库的修改
      2. 第二阶段：
         1. 协调者收集所有参与者的意见，若收到not commit T，将Abort T记录日志，向所有参与者发送Abort T，让所有参与者撤销自身的预操作
            1. 若全是prepare T ，协调者将commit T写入磁盘，向所有参与者发送commit T
            2. 超时没收到，认为发送了VOTE\_ABORT 取消该事务执行
         2. 所有参与者收到Abort T，终止提交，记录日志；Commit T，提交，记录日志
      3. 总结：参与者每一步都会记录日志
      4. 糟糕情况：参与者收不到Commit或Abort
      5. 支持协议：IIOP
4. JTA（Java事务API）、JTS（Java事务服务）提供分布式事务服务，JTS定义了一套规范如何传递事务上下文。JTA约定的这些角色要进行事务上下文的交互，JTS约定了应该怎样进行去交互

JTA：

1.一种高层的、与实现无关的、与协议无关的API