西电期末复习用——构件与中间件

- 课件内容概要

基本概念

软件面临问题：

- 复杂度高
- 开发周期长
- 可靠性保证难

网络环境：一群通过网络互相连接得处理系统，每个处理节点由处理机硬件、操作系统及基本通信软件等组成

分布式软件：运行在网络环境中的软件系统

- 反映了现实世界中的分布性
- 用来改进某些应用程序的运行性能

客户机/服务器模型（C/S）：

- 客户机提出对信息或服务的请求
- 服务器提高对这种请求的服务

C、S分离的优点：

- 更好的支持平台无关性
- 更好的可拓展性

缺点：

- 使得软件系统复杂化
- C、S两端需要分别编写程序，并保持一致性
- 调试、部署、运行更加困难
- 需要考虑更多的可靠性、安全性、性能等软件质量因素

两层结构缺陷：

- 客户端的负担比较重
- 客户端可移植性差
- 系统可维护性差
- 数据安全性差

3层结构：

- 客户向中间层服务器请求
- 由中间层服务器向数据库发送请求

优点：

- 减轻了客户端的负担

- 提高了客户端的可移植性
- 增强了数据安全性
- 增强了系统的可维护性
- 更好的性能和可伸缩性
- 大量的中间层中间件平提供了丰富的系统级服务

构件软件：

- 构件是系统中可以明确辨识的构成成分
- 有一定意义、相对独立，可以重复使用的软件实体
- 提供了软件复用的基本支持
- 语言未知
- 物理位置未知
- 基于对象实现的

数据类型：

- CORBA：枚举、常量、结构
- EJB：JAVA以及任何类型的对象（实现了Serializable或Remote接口）
- Web Serivice：XML描述的都行

对象撤销：

- CORBA：没有提供删除服务
- EJB：提供对Bean的生命周期管理策略
- WS：没有系统级支持

中间件特性：

- 平台性
- 支撑性
- 复用性
- 解耦性
- 互操作性

类型：

- 数据访问中间件
- 远程过程调用中间件
- 消息中间件
- 事务（交易）中间件
- 构件中间件（分布式对象中间件）

提供：

- 提供构件运行环境：异构屏蔽性
- 提供互操作机制：互操作
- 提供公共服务：软件复用

运行环境：

- 管理生命周期
- 实例
- 元信息

互操作：

- 高层通信协议（RPC\IIOP\DCOM\JRMP\RMI）
- 编组和解组
- 远程调用

公共服务：

- 通用格式
- 系统级服务
- 共性功能
- 命名服务、事务服务、安全服务、消息服务等

实现过程：

- 定义接口
- 实现分布式对象
- 实现服务程序
- 实现客户端程序
- 生成客户端桩和服务端框架
- 启动RMI远程对象注册表
- 启动服务程序
- 运行客户端程序

CORBA基本原理

规范：

- ORB体系结构
- OMG接口定义语言LDL
- 网络通讯协议GLOP和LLOP
- 可移植对象适配器POA
- 组件模型CCM

优势：

- 提供了一个工业标准而不是一个软件产品
- 高度可互操作性，保证了分布式对象可互相通信
- 可伸缩性和容错性
- 有效处理大量客户请求的服务程序

未解决的问题：

- 时间延迟
- 随机时序
- 死锁
- 负载均衡

CORBA应用程序开发过程：

- 面向对象分析与设计
- LDL编写对象接口
- 编译LDL文件生成桩和框架
- 编写客户程序代码/编写对象实现和服务程序
- 编译客户端程序/编译服务端程序
- 部署应用程序
- 运行应用程序

接口对外提供的信息：

- 对象/接口的名字
- 对象上可进行的操作
- 相关的数据类型定义、常量定义、异常定义等信息

词法规则：

- 关键字大小写敏感、标识符大小写无关
- C++子集与LDL独特的调用机制
- 以自然语言表述
- 规格说明（6类）：模块、常量、类型（9种基本的数据类型）、异常、接口（核心内容）、值

接口类型：

- 操纵型接口
- 工厂型
- 查找与选择型
- 管理型

J2EE

构件依赖于容器（Container）：

- 生命周期管理
- 构件的部署
- 为构件的运行指派线程（构件不是独立运行的）

提供：

- 应用构件（组件）
- 服务（开发、运行）
- 通信

客户端构件：

- Applets
- Application Clients

服务端构件：

- Web构件（Servlets,JSPs）

- EJBs

特点：

- 基于容器
- 开发与部署分离
- 基于角色的开发过程

EJB过程：

- 客户端利用JNDI查找EJB Home
- 客户端利用EJB Home的create方法创建一个EJB
- 本地EJB Home通知容器创建一个远程EJB构件
- 容器向客户端返回EJB Object stub
- 客户端调用EJB Object上的方法，调用被转发到远程的EJB完成

JNDI

- 类似于通用的目录服务
- 屏蔽了不同目录服务之间的差异
- 提供与厂商无关的数据库连接
- 提供一种通用的方法来查询、更新关系型数据库表
- 事务交易管理

JDNI支持的事件：

- 建立与数据库的连接
- 向数据源发送查询和更新语句
- 处理结果

安全控制级别：

- 认证
- 授权

安全性控制：

- 声明角色
- 声明规则
- 编程实现

EJB优点：

- 提供系统资源的利用率和效率
- 应用程序与具体的资源无关

生命周期映射到了6个角色上：

- I2EE Product Provider
- Tool Provider
- Application Component Provider

- Application Assembler
- Deployer
- System Administrator
- 一个角色的输出经常是另一个角色的输入
- 同一个人可能会执行两个或多个角色的任务
- 一个角色的任务也可能由几个人完成

J2EE优点：

- 支持多层应用开发模型，尤其是基于Web的应用
- 将实现多层应用的工作分为两部分
    - 系统服务由平台提供
    - 开发者关注于商业逻辑和表示逻辑的实现
- Write Once, Run Anywhere
- 丰富的部署时定制特性
    - 安全控制级别资源管理
- 明确划分任务和责任
- 灵活的安全性模型

EJB

构件技术：

- 分布式对象技术
- 服务端构件技术
- Component

服务框架：

- 支持大量的、由应用服务器提供的系统级服务
- 开发者可以关注于应用商业逻辑的实现
- 大大提高了开发效率，缩短了应用的开发周期
- 平台独立性
- 封装特性
- 可定制性
- 协议无关性
- 通用性
- 系统资源的可伸缩性

遗产系统：一个已经运行了很长时间，对于系统而言很重要，但不知道如何处理的软件系统

集成软件系统：

- 使遗产系统成为可以在网络中访问的平台无关系统
- 支持遗产系统访问各种数据库管理系统

集成方式：

- Connector
- CORBA
- JDBC

构件模型定义了开发可重用构件的方式

EJB与JavaBeans

EJB vs. JavaBeans

EJB	JavaBean
EJB用于 服务端应用开发	**JavaBean**用于 客户端应用开发
EJB构件是可部署的	**JavaBean**构件是不可部署的
EJB支持使用部署描述符 对**EJB**应用进行定制化	**JavaBean**中对应用的 定制化只能在开发阶段
EJB构件是分布式对象	**JavaBean**构件 不是分布式对象
EJB构件对终端用户不可见	部分**JavaBean**构件 对终端用户可见

38

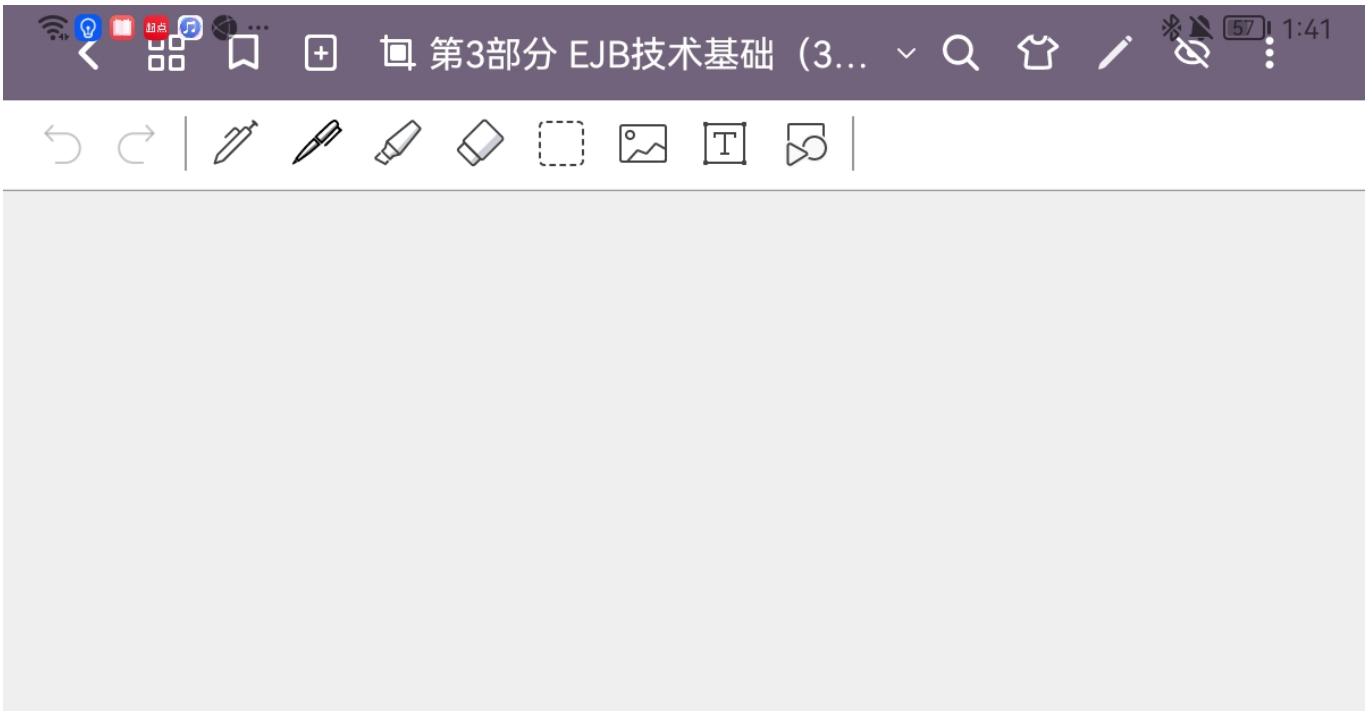
EJB体系结构组成：

- Enterprise Bean
- Home Interface
- Remote Interface
- EJB Container
- EJB Server
- EJB Client

开发角色：

- Enterprise Bean Provider
- Application Assembler
- EJB Deployer
- EJB Server Provider
- EJB Container Provider
- System Administrator

Remote接口设计原则：



Remote接口设计原则

规则	原因
所有的**remote**接口 应该继承**EJBObject**接口	该接口定义了所有**enterprise bean** 必须支持的服务
Remote接口中定义的每一个方法都 必须在相应的**enterprise bean**类中 有一个对应方法	客户应用不能直接访问 商业方法的实现
Remote接口中定义的方法 应该遵循**Java RMI**的规则	**EJB**对象是分布式**Java**对象
Remote接口中定义的方法 应该抛出**RemoteException**异常	
Remote接口中定义的方法的参数 应该是合法的**Java RMI**类型的参数	

102

95/122

Home接口设计原则：

Home接口设计原则

规则	原因
所有的home接口 应该继承EJBHome接口	EJBHome中定义了基本的服务
Home接口中定义的每一个create方法 都必须在相应的enterprise bean类中 有一个对应的ejbCreate方法	客户应用不能直接访问enterprise bean的初始化方法
Home接口中定义的方法 应该遵循Java RMI的规则	分布式Java对象
Home接口中定义的方法 应该抛出RemoteException异常	
Home接口中定义的create方法 应该抛出CreateException异常	
Home接口中定义的方法的参数 应该是合法的Java RMI类型的参数	

110

103/122

Enterprise bean类设计原则：

Enterprise bean类设计原则

规则	原因
Enterprise bean类必须实现**EnterpriseBean**接口	该接口定义了所有**enterprise bean**必须支持的服务
Enterprise bean类必须定义为**public**类	允许其它类使用其中定义的方法
Enterprise bean类必须实现定义在**remote**接口中的商业方法	实现特定的任务
Enterprise bean类必须实现定义在**home**接口中的**create**方法对应的**ejbCreate**方法	使用客户程序传过来的参数初始化**enterprise bean**

120

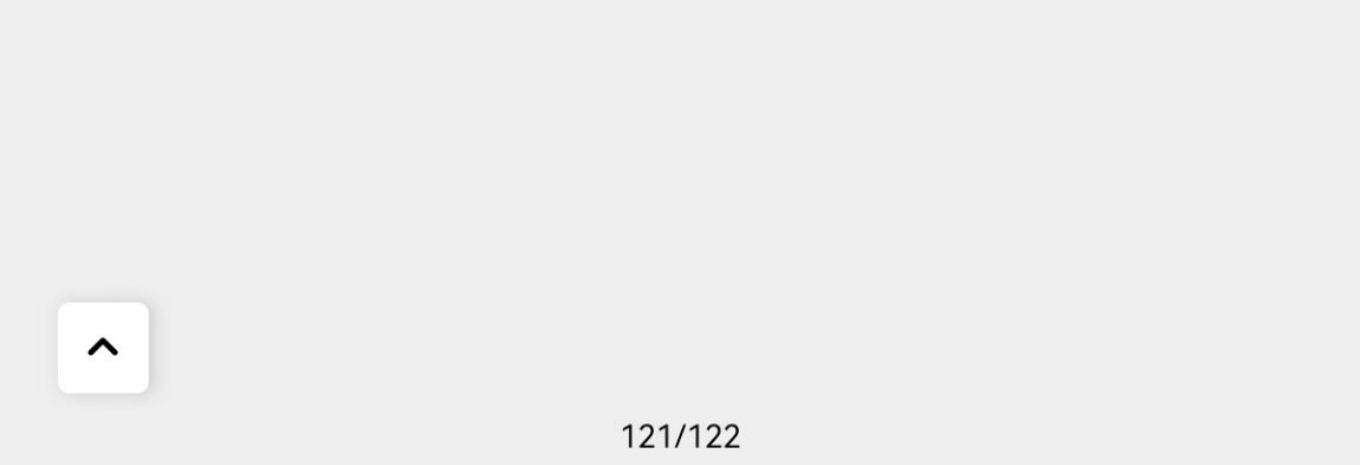
113/122

Primary key类设计原则：

Primary key类设计原则

规则	原因
Primary key类必须实现java.io.Serializable接口	支持在网络中传递primary key的值
Primary key类必须定义为public类	支持数据库表中的字段总可以被entity bean访问
Primary key类必须遵循Java RMI的规则	支持远程调用

128



Web开发技术

Ajax对比传统开发技术：

Servlet与CGI对比：

- 线程与进程
- Java平台无关性
- 三层架构与两层架构（Web服务器 + 连接池 + 数据库服务器）

Servlet缺点：

- HTML与Java代码混合，维护困难
- 每一行都是一个println语句，效率低下

JSP：

- 将HTML与Java分离，降低耦合性，提高可维护性
- 会进行语法检测，可维护性好
- 执行前转译为Java文件

SOA

产生背景：

- 信息化需求拉动
- 软件技术推动
    - 目的：提高软件质量、加快软件研发效率
    - 技术途径：屏蔽异构性、实现互操作、软件复用技术的持续探索

面向过程、面向对象、面向组件、面向接口、面向服务

服务：精确定义、封装完善、独立于其他服务所处环境和状态的函数

特征：

- 一个架构
- 提供软件资产展示和交互的标准方式
- 使之成为构件，可复用
- 关注于应用组装而不是实现细节
- 基于已有创建新的应用
- 用于与企业外部应用整合

要素：

- 互操作
- 软件复用
- 解耦

Web服务特征：

- 完好的封装性
- 松散耦合
- 使用协约的规范性
- 使用标准协议规范
- 高度可集成的功能

Web Serivice是SOA的一种重要实现方式：

- 提供了一种分布式计算的方法
- 实现语言、操作系统和硬件环境无关
- 服务提供者和请求者之间松散耦合
- 基于开发的标准提高了不同应用之间的互操作性

架构演化：

- 建立知识领域
- 使用通用语言描述
- 构建关系模型图
- 确定实体、服务、值对象、模块、聚合、工厂、仓库

SOA和微服务区别：

- 组件大小：大块业务逻辑/单独任务或小块业务逻辑
- 耦合：通常松耦合/总是松耦合
- 公司架构：任何类型/小型、专注于功能交叉团队
- 管理：着重中央管理/着重分散管理
- 目标：确保应用能够交互操作/执行新功能、快速拓展开发团队

共同点：

- 服务化
- 敏捷快速

传统架构缺点：

- 效率低下
- 维护困难
- 不灵活
- 稳定性差
- 扩展性差

微服务优点：

- 可快速迭代和演化
- 可维护性强
- 容错性好
- 每个服务可以各自拓展
- 各个服务之间可以独立部署
- 足够内聚，代码易于理解，开发效率高
- 不会被长期限制在某个技术栈上
- 容易扩大开发团队，针对每个服务组建开发团队

缺点：

- 开发人员要处理分布式系统的复杂性
- 服务管理复杂
- 应用时机难把握

思考题

1.描述三层分布式软件体系结构，并简要分析其相较于三层结构的特点和优势

- 分为表示层、业务逻辑层、数据层，将双层结构中客户端的业务逻辑层迁移出来，客户端只保留表示层
- 客户向中间层服务器请求，再由中间层服务器向数据库发送请求
- 减轻了客户端的负担
- 提高了客户端的可移植性
- 增强了数据安全性
- 增强了系统的可维护性
- 更好的性能和可伸缩性
- 大量的中间层中间件平提供了丰富的系统级服务

2.分析构件与普通对象的不同

- 构件要求是自描述，独立性更强，对象更为“白盒”
- 构件提供的是服务，对象提供的是功能
- 构件接口描述了问题域，而对象接口描述了方案域
- 对象是类的实例，是动态的，构件有静态的，有动态的
- 构件 = 对象 + 接口
- 构件具有语言无关性
- 构件可以在任何粒度上透明的使用，无需考虑位置与实现

3.采用Stub/Skeleton结构，完成一次远程调用的过程

- 启动RMI远程对象注册表
- 启动服务程序
- 运行客户端程序
    - 根据主机名和对象标识解析远程对象
    - 根据接口定义调用服务

4.现有集成中间件为分布式系统开发提供的主要支持

- 提供构件运行环境
- 提供互操作机制
- 提供公共服务

5.为什么要用值类型

- 值类型直接存储其值，变量本身就包含了其实例数据
- 值类型变量就永远不会影响到其他的值类型变量，而两个引用类型变量则很有可能指向同一地址，从而发生相互影响