**JavaEE开发四大常用框架**

**Struts**

Struts是一个基于Sun Java EE平台的MVC框架，主要是采用Servlet和JSP技术来实现的。

Struts框架可分为以下四个主要部分，其中三个就和MVC模式紧密相关：

1、模型 （Model），本质上来说在Struts中Model是一个Action类（这个会在后面详细讨论），开发者通过其实现商业逻辑，同时用户请求通过控制器（Controller）向Action的转发过程是基于由struts-config.xml文件描述的配置信息的。

2、视图（View），View是由与控制器Servlet配合工作的一整套JSP定制标签库构成，利用她们我们可以快速建立应用系统的界面。

3、控制器（Controller），本质上是一个Servlet，将客户端请求转发到相应的Action类。

4、一堆用来做XML文件解析的工具包，Struts是用XML来描述如何自动产生一些JavaBean的属性的，此外Struts还利用XML来描述在国际化应用中的用户提示信息的（这样一来就实现了应用系统的多语言支持）。

**Spring**

Spring是轻量级的Java EE应用程序框架。

Spring的核心是个轻量级容器（container），实现了IoC（Inversion of Control）模式的容器，Spring的目标是实现一个全方位的整合框架，在Spring框架下实现多个子框架的组合，这些子框架之间彼此可以独立，也可以使用其它的框架方案加以替代，Spring希望提供one-stop shop的框架整合方案 。

Spring不会特別去提出一些子框架来与现有的OpenSource框架竞争，除非它觉得所提出的框架夠新夠好，例如Spring有自己的 MVC框架方案，因为它觉得现有的MVC方案有很多可以改进的地方，但它不强迫您使用它提供的方案，您可以选用您所希望的框架来取代其子框架，例如您仍可以在Spring中整合您的Struts框架 。

Spring的核心概念是IoC，IoC的抽象概念是「依赖关系的转移」，像是「高层模组不应该依赖低层模组，而是模组都必须依赖于抽象」是 IoC的一种表现，「实现必须依赖抽象，而不是抽象依赖实现」也是IoC的一种表现，「应用程序不应依赖于容器，而是容器服务于应用程序」也是IoC的一种表现。

**Hibernate**

Hibernate是一个开放源代码的对象关系映射框架，它对JDBC进行了轻量级的对象封装，使得Java程序员可以使用对象编程思维来操纵数据库。Hibernate可以在应用EJB的Java EE架构中取代CMP，完成数据持久化。它还可以应用在任何使用JDBC的场合，既可以在Java的客户端程序实用，也可以在Servlet/JSP的Web应用中使用

**Swing**

图形用户接口（GUI）库最初的设计目的是让程序员构建一个通用的GUI，使其在所有的平台上都能够正常的显示。但是比较遗憾的是AWT产生的是在各系统看来都同样欠佳的图形用户接口，JAVA1.2为老的java1.0 AWT添加了Java基础类（JFC），这是一个被称为“Swing”的GUI的一部分。Swing是第二代GUI开发工具集，AWT采用了与特定平台相关的实现，而绝大部分Swing组件却不是。Swing是构筑在AWT上层的一组GUI组件的集合，为了保证可移植性，它完全用Java语言编写，与AWT相比，Swing提供了更完整的组件，引入了许多新的特性和能力。Swing提供了更多的组件库，如：JTable，JTree，Jcombox。Swing也增强了AWT中组件的功能。正是因为Swing具备了如此多的优势所以我们以后在开发中都使用Swing。JComponent类是Swing组件的基类，而JComponent继承自Container类，因此，所有的Swing组件都是AWT的容器。Swing采用了MVC设计模式。

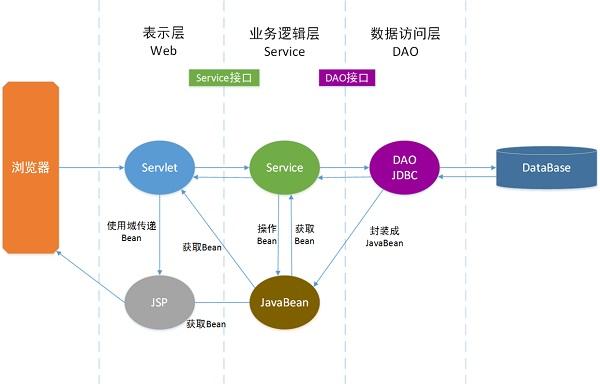
**JavaEE应用的分层模型**

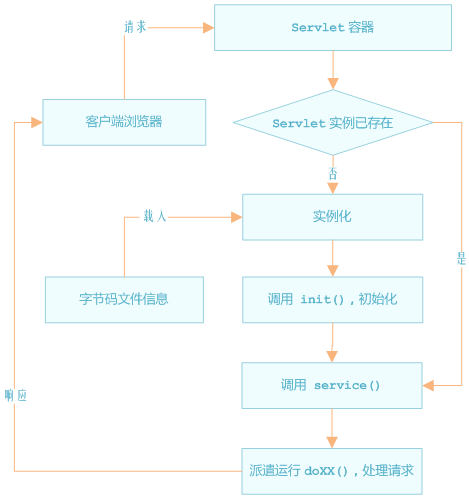
1.Domain Object（领域对象）层：此层由一系列的POJO（Plain Old Java Object, 普通的、传统的Java对象）组成，这些对象是该系统的Domain Object, 往往包含了各自所需实现的业务逻辑方法。

2. DAO(Data Access Object,数据访问对象）层：此层由一系列的DAO组件组成，这些DAO实现了对数据库的创建、查询、更新和删除(CRUD)等原子操作。在经典JavaEE应用中，DAO层也被改称为EAO层，EAO层组件的作用与DAO层组件的作用基本相似。只是EAO层主要完成对实体(Entity)的CRUD操作，因此简称为EAO层。

3. 业务逻辑层：此层由一系列的业务逻辑对象组成，这些业务逻辑对象实现了系统所需要的业务逻辑方法。这些业务逻辑方法可能仅仅用于暴露Domain Object对象所实现的业务逻辑方法，也可能是依赖DAO组件实现的业务逻辑方法。

4. 控制器层：此层由一系列控制器组成，这些控制器用于拦截用户请求，并调用业务逻辑组件的业务逻辑方法，处理用户请求，并根据处理结果转发到不同的表现层组件。

5. 表现层：此层由一系列的JSP页面、velocity页面、PDF文档视图组件组成，负责收集用户请求，并显示处理结果。



**说明Web服务的体系结构模型？它的三个核心协议分别是什么？**

Web服务是一种崭新的分布式计算模型，是Web上数据和信息集成的有效机制。

三个构成元素为：Serverice Broker、Service Provider、Service Requester

三个核心协议：简单对象访问协议SOAP；统一描述、发现和集成协议UDDI；Web服务描述语言WSDL

**简答题：**

1. **软件体系结构建模的种类有：**

结构模型、这是一个最直观、最普遍的建模方法。这种方法以体系结构的构件、连接件和其他概念来刻画结构，并力图通过结构来反映系统的重要语义内容，包括系统的配臵、约束、隐含的假设条件、风格、性质等。 研究结构模型的核心是体系结构描述语言

框架模型、框架模型与结构模型类似，但它不太侧重描述结构的细节而更侧重于整体的结构。 框架模型主要以一些特殊的问题为目标建立只针对和适应该问题的结构

功能模型、功能模型认为体系结构是由一组功能构件按层次组成，下层向上层提供服务。 功能模型可以看作是一种特殊的框架模型

动态模型、动态模型是对结构或框架模型的补充，研究系统的“大颗粒”的行为性质。 例如，描述系统的重新配臵或演化。动态可以指系统总体结构的配臵、建立或拆除通信通道或计算的过程

功能模型、过程模型研究构造系统的步骤和过程。 结构是遵循某些过程脚本的结果

1. **软件体系结构模型可以分为哪几种，具体是如何划分的？**

**构件：**是具有某种功能的可重用的软件模板单元，表示了系统中主要的计算元素和数据存储。**连接件**：表示构件之间的交互。 **配置**：表示构件和连接件的拓扑逻辑和约束。**端口**：表示构件和外部环境的交互点。**角色**：定义了该连接交互的参与者。构件、连接件和配置是最基本的元素。

**3、什么是设计模式？分类？它与风格、框架有什么区别与联系？**

总体来说设计模式分为三大类：

创建型模式，共五种：工厂方法模式、抽象工厂模式、单例模式、建造者模式、原型模式。

结构型模式，共七种：适配器模式、装饰器模式、代理模式、外观模式、桥接模式、组合模式、享元模式。

行为型模式，共十一种：策略模式、模板方法模式、观察者模式、迭代子模式、责任链模式、命令模式、备忘录模式、状态模式、访问者模式、中介者模式、解释器模式。

其实还有两类：并发型模式和线程池模式。

设计模式是对通用设计问题的重复解决方案。软件体系结构风格是描述某一特定应用领域中系统组织方式的惯用模式。软件框架是整个或部分系统的可重用设计 模式比框架更加抽象

框架是模式的特例化 设计模式被实现成为框架后，可以极大的减轻从设计到实现的鸿沟

利用了模式的框架比没有利用模式的框架更容易理解、更能被设计与实现重用

通常成熟的框架包含了多种设计模式 一个框架不仅可以具体实现一个模式，还可以具体的实现多个模式 设计模式与风格两者为近义词，通常情况下可以互相通用

风格主要是指大的，宏观的设计。模式既可宏观，又可微观。

**4 软件体系结构测试的内容与准则是什么？**

内容：构件端口行为与连接件约束是否一致、兼容，单元间的消息是否一致、可达，相关端口是否可连接，体系结构风格是否可满足。

准则：测试覆盖所有的构件及各个构件的接口。各个连接件的接口、构件之间的直接连接、构件之间的间接连接。

1. **软件体系结构风格：**

是描述某一特定应用领域中系统组织方式的惯用模式。

**6、软件体系结构**

（Software Architecture）软件体系结构以组件和组件交互的方式定义系统，说明需求与成品系统之间的对应关系，描述系统级别的可伸缩性、能力、吞吐量、一致性和兼容性等属性。 软件体系结构由组件、连接件和属性组成。

**7、层次结构有何优缺点？什么是松弛的层次结构？**

优点：1.层次的复用性

2.对标准化的支持，允许在不同层使用来自不同商家的产品

3.依赖本地化，这样做便于支持系统的可移植性和可测试性4.可替换性，独立层次的实现能够被功能相同的模块替换 缺点：1.改变行为的连锁效应。

2.低效率。分层结构通常比单一层次的结构效率更低。

3.包含多余不必要的工作。

松弛的层次系统

1.是分层模型的一种变种

(1) 每层可以使用其下面所有层的服务而不仅仅是相邻层的服务

(2) 有些服务提供给相邻的上一层，而其余的服务则提供给高于它的所有层

2.灵活性和性能的提高以牺牲可维护性为代价

3.经常用于系统软件，而不常用于应用软件的设计

(1) 系统软件比起应用软件来很少修改

(2) 系统软件对性能的要求高于可维护性的要求

**8、功能与非功能特性？非功能特性的重要性**

功能特性主要是直接针对客户的功能需求，多数是容易感知和判断的。非功能特性主要包括系统的：可变性，互操作性，效率，可靠性，可测试性，可重用性。不成熟的客户、投资决策者、设计者往往片面追求表明功能的要求而忽略内在的结构和非功能特性。软件系统越大越复杂、生命周期越长，非功能特性就越重要。

**9、 B/S体系结构的优点缺点有什么？（Browser/Server）**

**优点**1）基于B/S体系结构的软件，系统安装、修改和维护全在服务器端解决。用户在使用系统时，仅仅需要一个浏览器就可运行全部的模块，真正达到了“零客户端”的功能，很容易在运行时自动升级。

2）B/S体系结构还提供了异种机、异种网、异种应用服务的联机、联网、统一服务的最现实的开放性基础。

**缺点**1）B/S体系结构缺乏对动态页面的支持能力，没有集成有效的数据库处理功能。

2）B/S体系结构的系统扩展能力差，安全性难以控制。

3）采用B/S体系结构的应用系统，在数据查询等响应速度上，要远远地低于C/S体系结构。

4）B/S体系结构的数据提交一般以页面为单位，数据的动态交互性不强，不利于在线事务处理(OLTP)应用。

**10、C/S体系结构的优点缺点有什么？（Client/Server）**

**优点**：能充分发挥客户端PC的处理能力，很多工作可以在客户端处理后再提交给服务器，所以CS客户端响应速度快。

操作界面漂亮、形式多样，可以充分满足客户自身的个性化要求。

C/S结构的管理信息系统具有较强的事务处理能力，能实现复杂的业务流程。

安全性能可以很容易保证，C/S一般面向相对固定的用户群，程序更加注重流程，它可以对权限进行多层次校验，提供了更安全的存取模式，对信息安全的控制能力很强。一般高度机密的信息系统采用C/S结构适宜。

**缺点：**需要专门的客户端安装程序，分布功能弱，针对点多面广且不具备网络条件的用户群体，不能够实现快速部署安装和配置。

兼容性差，对于不同的开发工具，具有较大的局限性。若采用不同工具，需要重新改写程序。

开发、维护成本较高，需要具有一定专业水准的技术人员才能完成，发生一次升级，则所有客户端的程序都需要改变。

用户群固定。由于程序需要安装才可使用，因此不适合面向一些不可知的用户，所以适用面窄，通常用于局域网中。

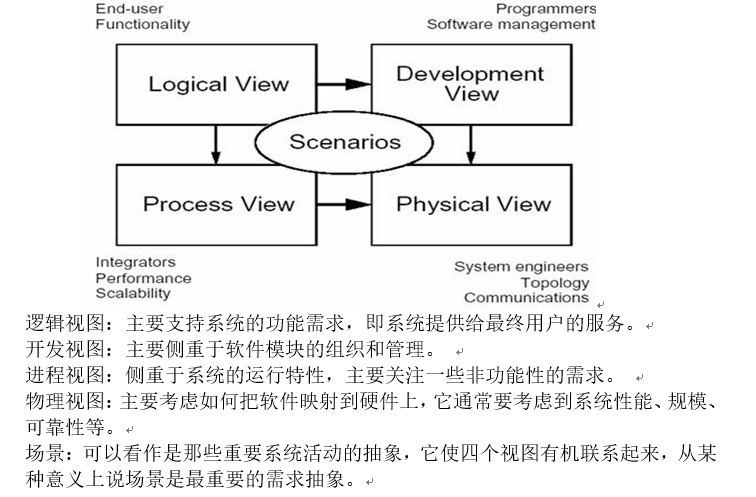
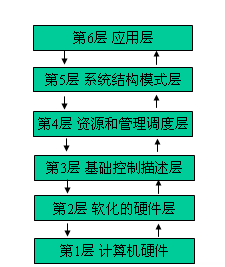
**11、软件体系结构的动态性主要分为以及特征：**

交互式动态性、结构化动态性、体系结构动态性等三类。1、可构造性动态特征2、适应性动态特征3、智能型动态特征

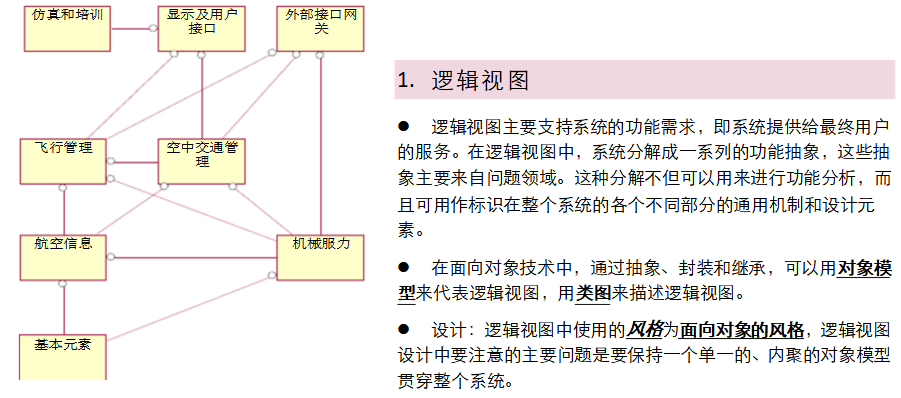
**12、SA的形式化描述，常用的有哪些？**

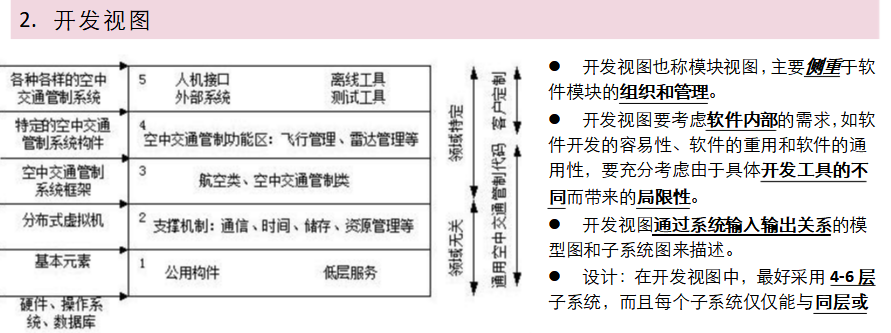
理论的形式化方法1. Z Notation 2. CSP 3. 类属理论 4. 化学抽象机模型

**13、画图，层次结构**

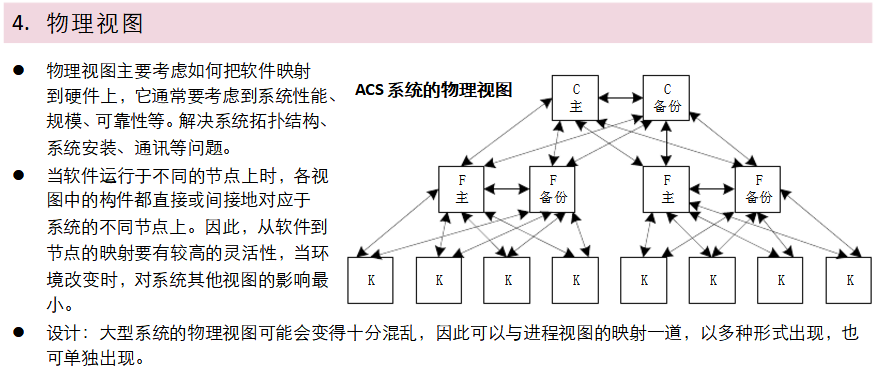


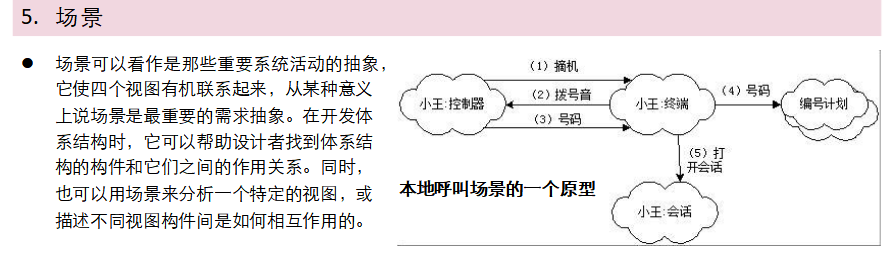
**14、画出“4+1”视图模型图，分析各部分的原理和功能。**

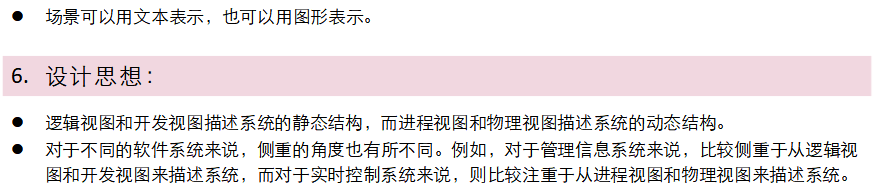












**13、OCP的思想**

开放-封闭法则，软件组成实体应该是可扩展的，但是不可修改的。OCP认为我们应该试图去设计出永远也不需要改变的模块。我们可以添加新代码来扩展系统的行为。我们不能对已有的代码进行修改。符合该法则便意味着最高等级的复用性和可维护性。

**14、工厂与抽象工厂的区别与联系**

工厂方法模式是一种极端情况的抽象工厂模式，而抽象工厂模式可以看成是工厂方法模式的一种推广。

工厂方法模式：一个抽象产品类，可以派生出多个具体产品类。 一个抽象工厂类，可以派生出多个具体工厂类。 每个具体工厂类只能创建一个具体产品类的实例

抽象工厂模式：多个抽象产品类，每个抽象产品类可以派生出多个具体产品类。 一个抽象工厂类，可以派生出多个具体工厂类。 每个具体工厂类可以创建多个具体产品类的实例。

**15、mediator与Façade的区别与联系**

 Facade模式和Mediator模式都是用于连接多个类，简化类之间的联系。 Facade模式是把大系统藏在自己后面，由自己来做一个通用的接口。使用者不用关系下面的具体类，只要了解Facade的接口，就可以使用系统了。这几乎是最常用的设计模式了。Mediator也是连接大系统，但是它的客户并不是系统的使用者，而是系统中的各个模块；相应的，它的目的只是为了降低系统中各个模块之间的依赖关系。所谓策略，即如何使用系统中的模块。两个模式之间更内在的关系。即他们都是为了施加策略，Facade模式是从上面施加，Mediator是从下面施加。

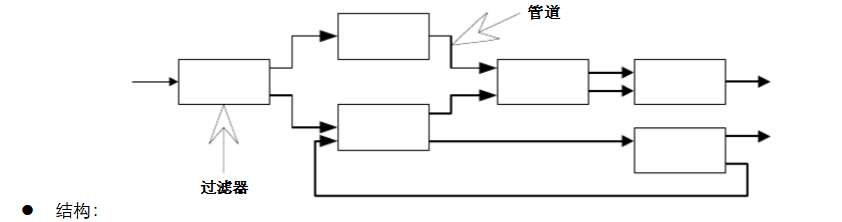
**16、经典的体系结构风格 (Garlan, Shaw)**

数据流风格：批处理序列；管道/过滤器。 调用/返回风格：主程序/子程序；面向对象风格；层次结构。 独立构件风格：进程通讯；事件系统。  虚拟机风格：解释器；基于规则的系统。 仓库风格：数据库系统；超文本系统；黑板系统

**1. 管道/过滤器 **

含义：每个构件都有一组输入和输出，构件读输入的数据流，经过内部处理，然后产生输出数据流。这个 过程通常通过对输入流的变换及增量计算来完成，所以在输入被完全消费之前，输出便产生了。 这里的构件被称为过滤器，这种风格的连接件就象是数据流传输的管道，将一个过滤器的输出传到另一过滤器的输入。 

组成：构件（过滤器）和连接件（管道）



优点（1）使得软构件具有良好的隐蔽性和高内聚、低耦合的特点；（2）允许设计者将整个系统的输入/输出行为看成是多个过滤器的行为的简单合成；（3）支持软件重用。（4）系统维护和增强系统性能简单。（5）允许对一些如吞吐量、死锁等属性的分析；（6）支持并行执行。

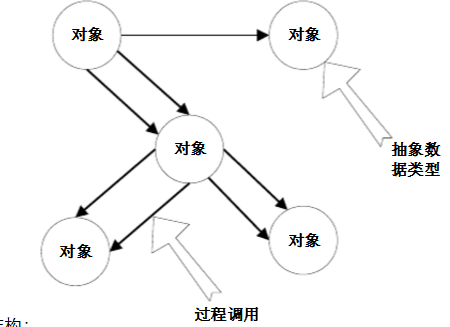
缺点（1）通常导致进程成为批处理的结构。这是因为虽然过滤器可增量式地处理数据，但它们是独立的，所以设计者必须将每个过滤器看成一个完整的从输入到输出的转换。

（2）不适合处理交互的应用。当需要增量地显示改变时，这个问题尤为严重。

（3）因为在数据传输上没有通用的标准，每个过滤器都增加了解析和合成数据的工作，这样就导致了系统性能下降，并增加了编写过滤器的复杂性。

**2. 数据抽象和面向对象组织**

 含义：这种风格建立在数据抽象和面向对象的基础上，数据的表示方法和它们的相应操作封装在一个抽象 数据类型或对象中。这种风格的构件是对象，或者说是抽象数据类型的实例。对象是一种被称作管理者的构件，因为它负责保持资源的完整性。对象是通过函数和过程的调用来交互的。  组成：构件（对象）



优点：  因为对象对其它对象隐藏它的表示，所以可以改变一个对象的表示，而不影响其它的对象；  设计者可将一些数据存取操作的问题分解成一些交互的代理程序的集合。

缺点：  为了使一个对象和另一个对象通过过程调用等进行交互，必须知道对象的标识。只要一个对象的标识改变了，就必须修改所有其他明确调用它的对象；并消除由此带来的一些副作用。例如，如果A使用了对象B，C 也使用了对象B，那么，C对B的使用所造成的对A的影响可能是料想不到的。

**3. 基于事件的隐式调用**

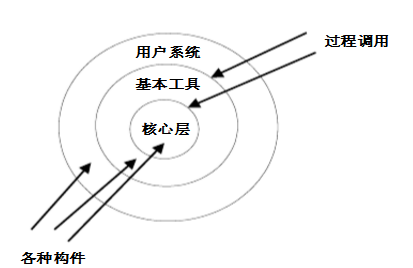
含义：构件不直接调用一个过程，而是触发或广播一个或多个事件。系统中的其它构件中的过程在一个或 多个事件中注册，当一个事件被触发，系统自动调用在这个事件中注册的所有过程，这样，一个事件的触发就导致了另一模块中的过程的调用。这种风格的构件是一些模块，模块既可以是一些过程，又可以是一些事件的集合。过程可以用通用的方式调用，也可以在系统事件中注册一些过程，当发生这些事件时，过程被调用。这种风格的主要特点是事件的触发者并不知道哪些构件会被这些事件影响。这样不能假定构件的处理顺序，甚至不知道哪些过程会被调用，因此，许多隐式调用的系统也包含显式调用作为构件交互的补充形式。 组成：构件（一些模块，模块既可以是一些过程，又可以是一些事件的集合） 

优点： 为软件重用提供了强大的支持。当需要将一个构件加入现存系统中时，只需将它注册到系统的事件中。为改进系统带来了方便。当用一个构件代替另一个构件时，不会影响到其它构件的接口。

缺点： 构件放弃了对系统计算的控制。一个构件触发一个事件时，不能确定其它构件是否会响应它。而且即 使它知道事件注册了哪些构件的构成，它也不能保证这些过程被调用的顺序。数据交换效率的问题。有时数据可被一个事件传递，但另一些情况下，基于事件的系统必须依靠一个共 享的仓库进行交互。在这些情况下，全局性能和资源管理便成了问题。 既然过程的语义必须依赖于被触发事件的上下文约束，关于正确性的推理存在问题。

**4. 分层系统**  

含义：层次系统组织成一个层次结构，每一层为上层服务，并作为下层客户。在一些层次系统中，除了一 些精心挑选的输出函数外，内部的层只对相邻的层可见。这样的系统中构件在一些层实现了虚拟机（在另一些层次系统中层是部分不透明的）。连接件通过决定层间如何交互的协议来定义，拓扑约束包括对相邻层间交互的约束。这种风格支持基于可增加抽象层的设计。允许将一个复杂问题分解成一个增量步骤序列的实现。由于每一层最多只影响两层，同时只要给相邻层提供相同的接口，允许每层用不同的方法实现，同样为软件重用提供了强大的支持。 组成：构件，连接件



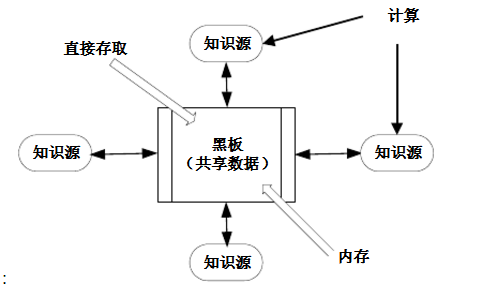
优点： 支持基于抽象程度递增的系统设计，使设计者可以把一个复杂系统按递增的步骤进行分解；支持功能增强，因为每一层至多和相邻的上下层交互，因此功能的改变最多影响相邻的上下层；支持重用。只要提供的服务接口定义不变，同一层的不同实现可以交换使用。这样，就可以定义一组标 准的接口，而允许各种不同的实现方法。 

缺点：并不是每个系统都可以很容易地划分为分层的模式，甚至即使一个系统的逻辑结构是层次化的，出于对 系统性能的考虑，系统设计师不得不把一些低级或高级的功能综合起来； 很难找到一个合适的、正确的层次抽象方法。

**5. 仓库系统及知识库**  

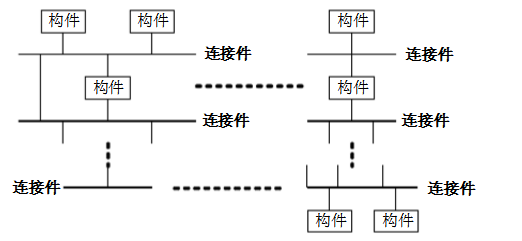
含义：在仓库风格中，有两种不同的构件：中央数据结构说明当前状态，独立构件在中央数据存贮上执行， 仓库与外构件间的相互作用在系统中会有大的变化。控制原则的选取产生两个主要的子类。若输入流中某类时间触发进程执行的选择，则仓库是一传统型数据库；另一方面，若中央数据结构的当前状态触发进程执行的选择，则仓库是一黑板系统。 组成：构件（中央数据结构，独立构件） 

例子：黑板系统的传统应用是信号处理领域，如语音和模式识别。另一个应用是松耦合代理数据共享存取。



**6. C2风格**  

含义：通过连接件绑定在一起的按照一组规则运作的并行构件网络。C2风格中的系统组织规则如下： 系统中的构件和连接件都有一个顶部和一个底部； 构件的顶部应连接到某连接件的底部，构件的底部则应连接到某连接件的顶部，而构件与构件之间的直 接连接是不允许的； 一个连接件可以和任意数目的其它构件和连接件连接； 当两个连接件进行直接连接时，必须由其中一个的底部到另一个的顶部。 组成：构件、连接件



特点：系统中的构件可实现应用需求，并能将任意复杂度的功能封装在一起； 所有构件之间的通讯是通过以连接件为中介的异步消息交换机制来实现的； 构件相对独立，构件之间依赖性较少。系统中不存在某些构件将在同一地址空间内执行，或某些构件共 享特定控制线程之类的相关性假设。

**17、什么是软件危机，软件危机的具体表现有哪些？**

软件危机：落后的软件生产方式无法满足迅速增长的计算机软件需求，从而导致软件开发与维护过程中出现一系列严重问题的现象。

软件危机的表现：软件成本日益增长 开发进度难以控制 软件质量差 软件维护困难

**18、** **MTTF,MTBF,MTRF分别是什么，它们有什么关系**

MTTF(mean time to failure，平均失效前时间) 还叫“正常运行时间”。MTBF,平均故障间隔时间又称平均无故障时间，指可修复产品两次相邻故障之间的平均时间，记为MTBF

MTTR（mean time to repair），叫“平均修复时间”（即修复一个故障平均需用的时间），也叫“故障停机时间”MTBF＝MTTF＋MTTR

**19、UML中的交互图有两种，分别是顺序图和协作图，请分析一下两者之间的主要差别和各自的优缺点。掌握利用两种图进行的设计的方法。**

答：顺序图可视化地表示了对象之间随时间发生的交互，它除了展示对象之间的关联，还显示出对象之间的消息传递。与顺序图一样，协作图也展示对象之间的交互关系。顺序图强调的是交互的时间顺序，而协作图强调的是交互的语境和参与交互的对象的整体组织。顺序图按照时间顺序布图，而协作图按照空间组织布图。顺序图可以清晰地表示消息之间的顺序和时间关系，但需要较多的水平方向的空间。协作图在增加对象时比较容易，而且分支也比较少，但如果消息比较多时难以表示消息之间的顺序。

**20、面向对象编程的四大原则**

1. 优先使用（对象）组合，而非（类）继承 2. 依赖倒转原则：针对接口编程，而非（接口的）实现 3. 开放－封闭法则（OCP） ：软件组成实体应该是可扩展的，但是不可修改的。 4. Liskov 替换法则（LSP） ：使用指向基类（超类）的引用的函数，必须能够在丌知道 具体派生类（子类）对象类型的情况下使用它们

**21、设计模式的六大原则**

1、开闭原则（Open Close Principle）开闭原则就是说对扩展开放，对修改关闭。

2、里氏代换原则（Liskov Substitution Principle）里氏代换原则(Liskov Substitution Principle LSP)面向对象设计的基本原则之一。 里氏代换原则中说，任何基类可以出现的地方，子类一定可以出现。

3、依赖倒转原则（Dependence Inversion Principle）这个是开闭原则的基础，具体内容：真对接口编程，依赖于抽象而不依赖于具体。

4、接口隔离原则（Interface Segregation Principle）这个原则的意思是：使用多个隔离的接口，比使用单个接口要好。降低依赖，降低耦合。

5、迪米特法则（最少知道原则）（Demeter Principle）一个实体应当尽量少的与其他实体之间发生相互作用，使得系统功能模块相对独立。

6、合成复用原则（Composite Reuse Principle）原则是尽量使用合成/聚合的方式，而不是使用继承。

**22、行为型模式**

行为型模式设计到算法和对象间的职责分配，不仅描述对象或类的模式，还描述它们之间的通信方式，刻划了运行时难以跟踪的复杂的控制流，它们将你的注意力从控制流转移到对象间的关系上来。行为对象模式使用对象复合而不是继承。常见行为型模式有11种：CCIIMM（Chain of Responsibility职责链、Command命令、Interpreter解释器、Iterator迭代、Mediator中介者、Memento备忘录），OSSTV（Observer观察者、State状态、Strategy策略、Template Method模版方法、Visitor访问者）。

**23、软件体系结构的评估方法**

质量属性专题研讨会 (QAW)体系结构权衡分析方法 (ATAM)软件体系结构分析方法 (SAAM)

积极的中间设计审核 (ARID)

**名词解释**

**SOA**

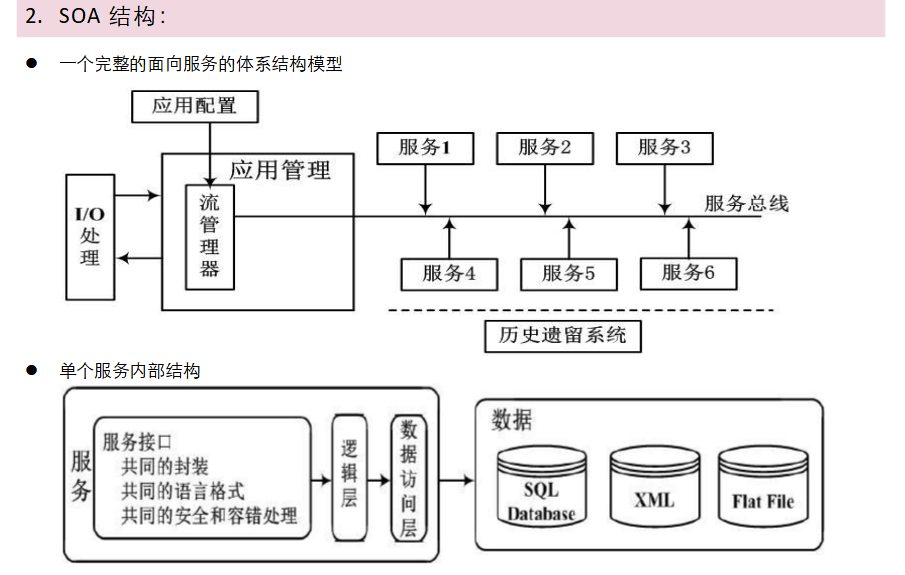
即service-oriented architecture，面向服务架构。它是一个组件模型，它

将应用程序的不同功能单元(称为服务)通过这些服务之间定义良好的接

口和契约联系起来。接口是采用中立的方式进行定义的，它应该独立于

实现服务的硬件平台、操作系统和编程语言。这使得构建在各种这样的

系统中的服务可以以一种统一和通用的方式进行交互。



**Architecture Styles**

定义为根据结构组织模式构成的软件系统族，表达了部件和他们之间的

关系。例如客户/服务器（Client /Server）结构、浏览器/服务器（Browser/Server）

结构、正交（Orthogonal）结构、专用领域（Domain Specific Styles）、

MVC、微核（Microkernel）、反射（Reflection ）、代理（Proxy ）等

**Framework**

是整个或部分系统的可重用设计，从设计模式角度来看，框架为大粒度的可复用的部件。从体系结构角度来看，框架是一个领域体系结构

**MVC**

MVC是三个单词的缩写,分别为：模型(Model)，视图(View)和控制Controller)。

MVC模式的目的就是实现Web系统的职能分工。 Model是应用对象，所有的操作都在这里实现，它若需要取得视图中的对象或更新视图，需通过控制器来进行处理。View是模型在屏幕上的表示，模型在进行操作后，其结果是通过视图显示的。Controller用于管理用户与视图发生的交互，定义用户界面对用户输入的响应方式。一旦用户需要对模型进行处理，不能直接执行模型，而必须通过控制器间接实现的。

**DSSA**

Domain Specific Software Architecture: 特定领域软件体系结构。建立一种基于体系结构的方法,这需要对体系结构,其一般性构件和互联,以及客户的需求按何种方式由构件来集成都要达成共识。

**ADT**

抽象数据类型（Abstract Data Type) 一个数学模型以及定义在该模型上的一组操作。 ADT 包括数据数据元素，数据关系以及相关的操作。

**SAAS**

Software as a service 软件即服务：是随着互联网技术的发展和应用软件的成熟， 而在 21 世纨开始兴起的一种完全创新的软件应用模式。

**Design Pattern**：设计模式=问题+解决方法+场景

**ADL**：

Architecture description language 为体系结构提供概念性框架， 也提供描述的具 体语法，也提供解析、显示、编译、分析戒者仿真体系结构描述使用的工具

**CSP**

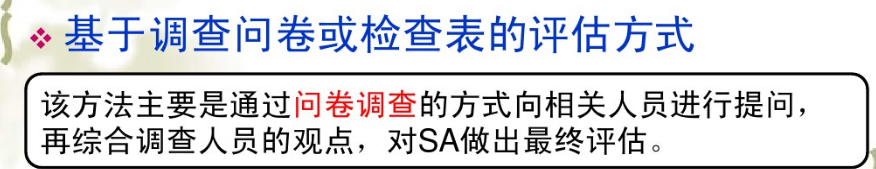
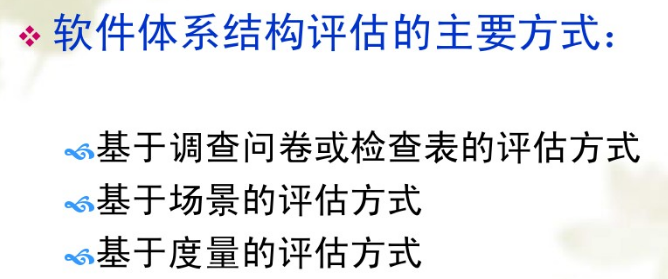
Communication sequential process：这是一种形式化的描述语言，主要表现了一 种基于顺序的通信处理过程。通过 CSP，可以对系统的功能行为以及系统的觃划说明迚入 深入的刻画。

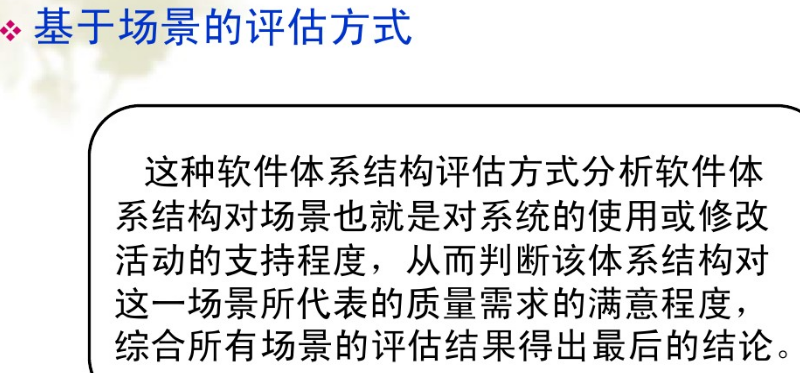
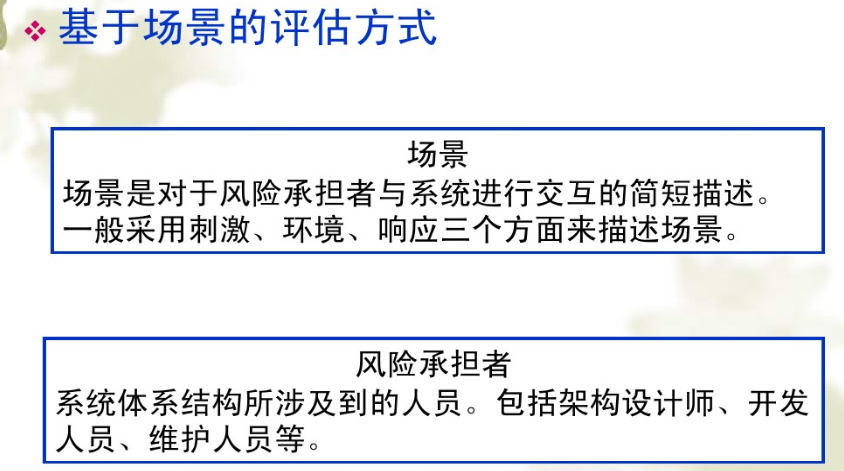
**SessionBean和EntityBean的区别，StatefulBean和StatelessBean的区别**

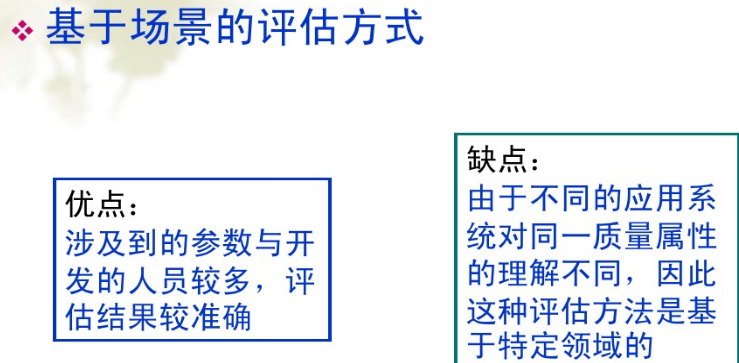
对于客户机，SessionBean是一种非持久性对象，它实现某些在服务器上运行的业务逻辑， EntityBean是一种持久性对象，它代表一个存储在持久性存储器中的实体的对象视图，或是一个由现有企业应用程序实现的实体。

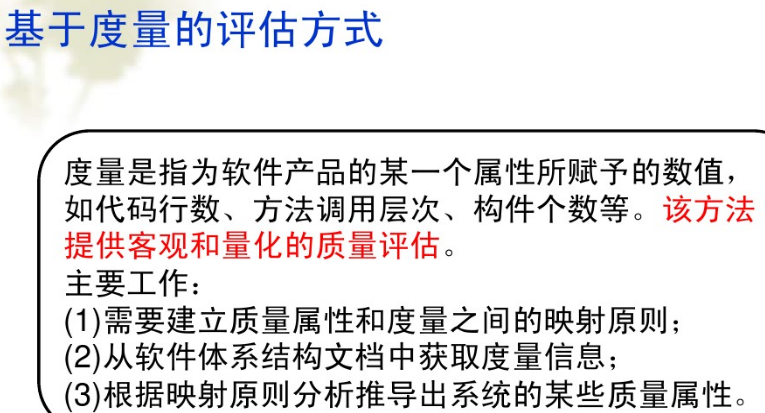
Session Bean还可以再细分为Stateful Session Bean与Stateless Session Bean，这两种的Session Bean都可以将系统逻辑放在method之中执行，不同的是Stateful Session Bean可以记录呼叫者的状态，因此通常来说，一个使用者会有一个相对应的Stateful Session Bean的实体。Stateless Session Bean虽然也是逻辑组件，但是他却不负责记录使用者状态，也就是说当使用者呼叫Stateless Session Bean的时候，EJB Container并不会找寻特定的Stateless Session Bean的实体来执行这个method。换言之，很可能数个使用者在执行某个Stateless Session Bean的methods时，会是同一个Bean的Instance在执行。从内存方面来看，Stateful Session Bean与Stateless Session Bean比较，Stateful Session Bean会消耗J2EE Server较多的内存，然而Stateful Session Bean的优势却在于他可以维持使用者的状态。

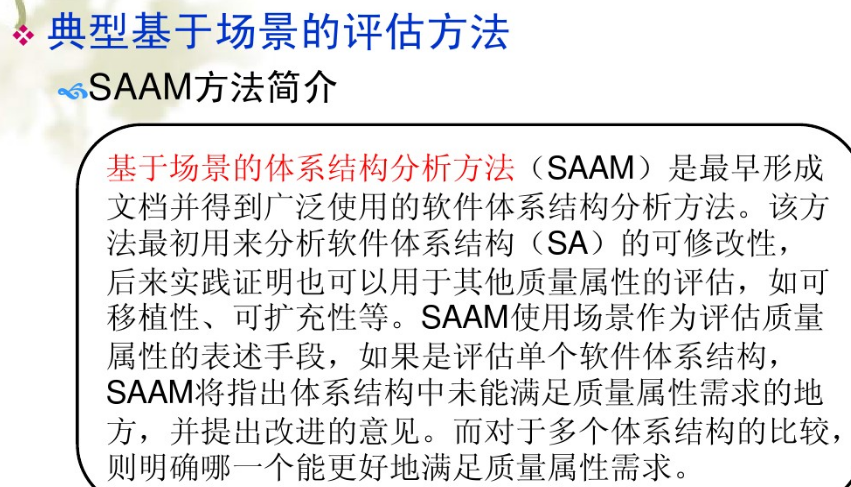
**分析题**

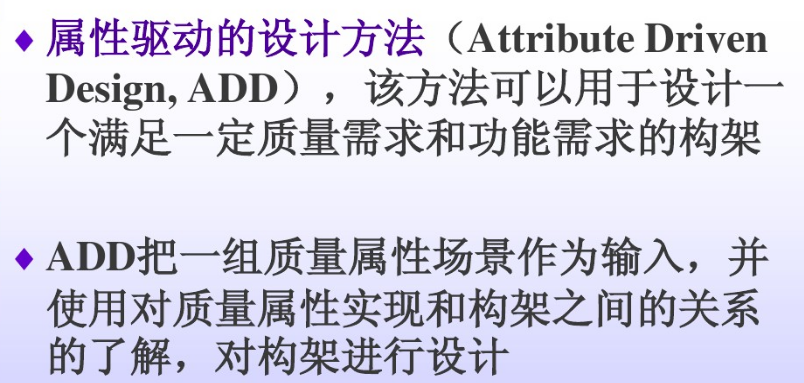
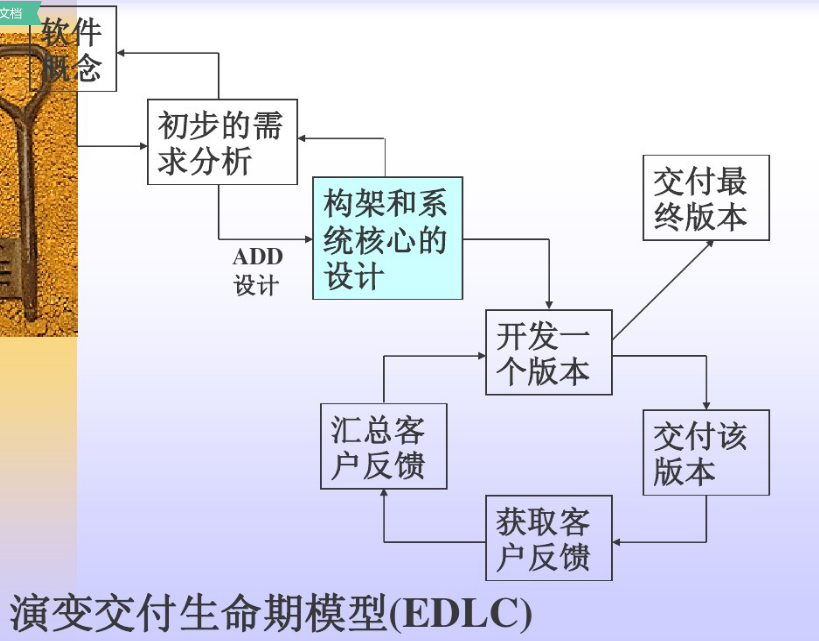


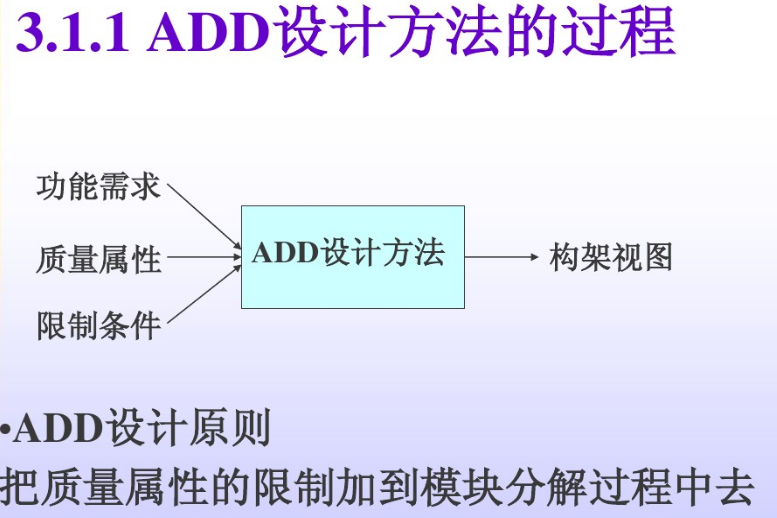


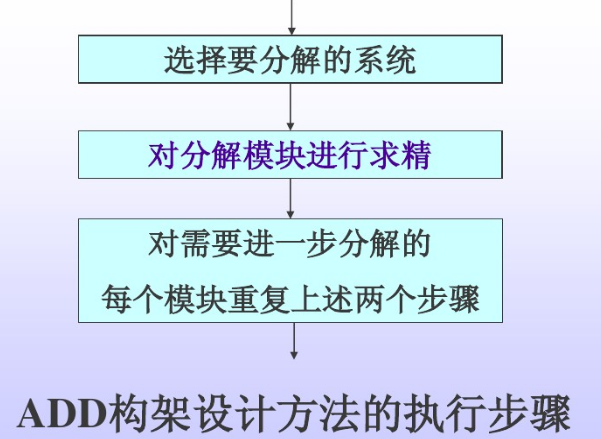


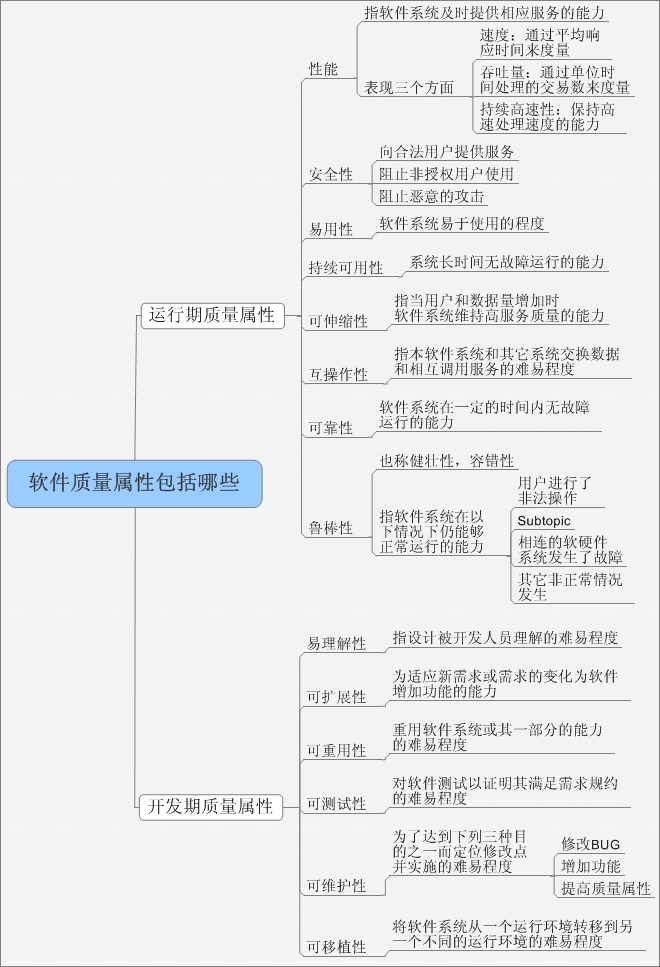










****

**请描述构架设计的ADD方法中对模块求精的5个步骤。**

a. 从具体的质量场景和功能需求集合中选择构架驱动因素。

b. 选择满足构架驱动因素的构架模式。

c. 实例化模块并根据用例分配功能，使用多个视图进行表示。

d. 定义子模块的接口。

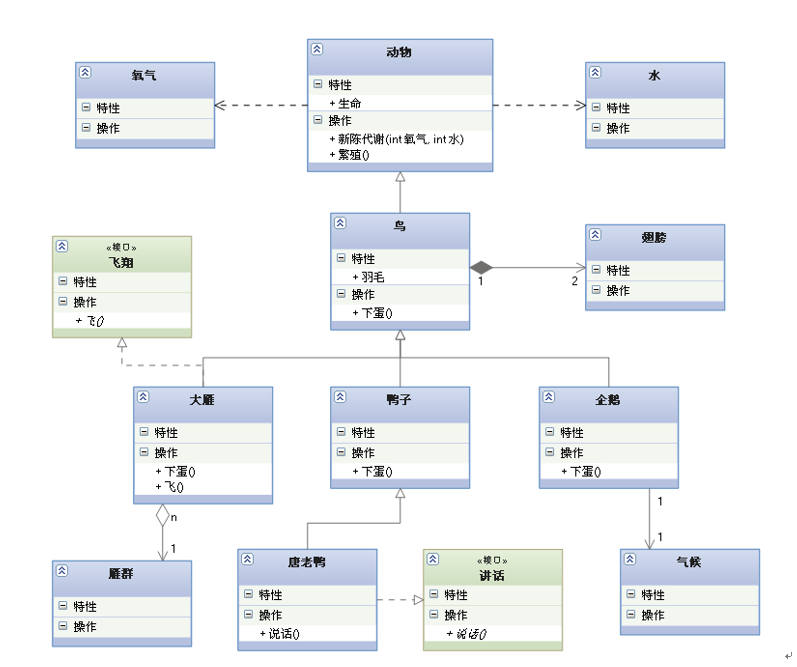
e. 验证用例和质量场景并对其进行求精，使它们成为子模块的限制。

**描述ATAM方法的步骤。**

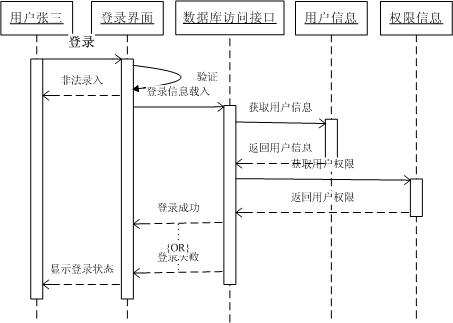
　ATAM方法的表述 商业动机的表述 构架的表述 对构架方法进行分类 生成质量属性效用树 分析构架方法 集体讨论并确定场景优先级 再次分析构架方法 结果的表述

**UML**

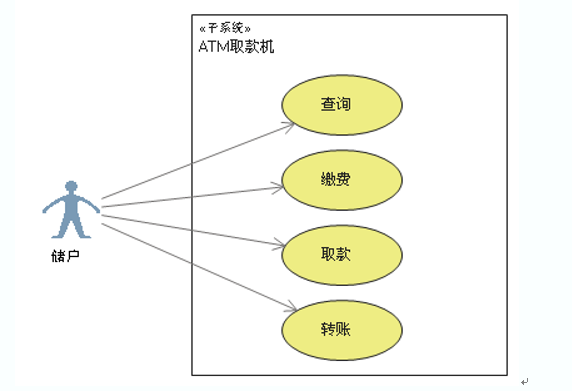
**类图**



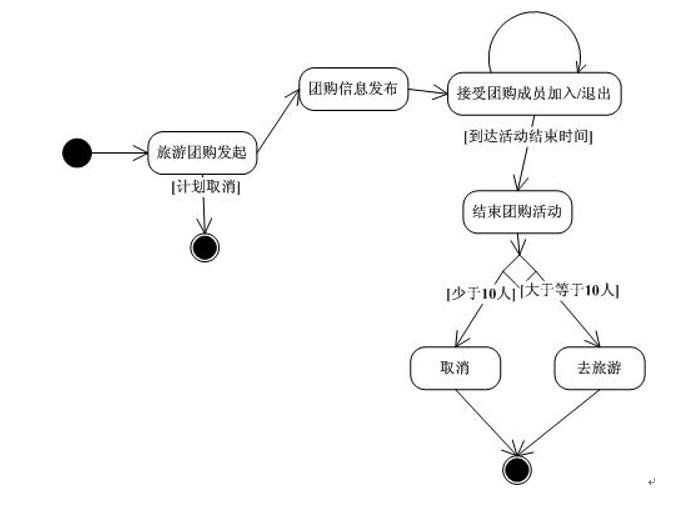
**时序图（sequence diagram）**



**用例图**



**状态图（state diagram）**



1. Which of the following is true about the requirements of a banking system?D. Security is an architectural nonfunctional requirement and the mandatory checking accounts a functional design requirement.

2. Which of the following are nonfunctional requirements?D. Performance, availability, scalability, and security

3. Which of the following is the most important item that should be considered when designing an application? D. Meeting the needs of the customer

***第三章面向对象的分析与设计***

1. Which one of the following items is not one of the phases of the Unified Process? B. Design

2. What is true about a use case? A. It is a complete end-to-end business process that satisfies the needs of a user. B. It is a description that represents a complete unit of functionality provided by something

as large as a system or as small as a class.

3. Which item is not true when speaking of a class? A. A class is a nonunique structure.

4. What is not true about use cases? A. There are three types of use cases: essential, real, and virtual. B. A virtual use case describes the user's virtual view of the problem and is technology independent.

5. What is not true about a sequence diagram? D. Sequence diagrams are static model views.

6. Which item is not an example of things that a state diagram could effectively model? B. A computer system infrastructure.

7. What is not true about a collaboration diagram?C. The interactions in a collaboration diagram are listed as alphabetically collated letters that help to trace the sequence of the interactions.

8. What item is not true about a component?C C. A node may be deployed on a component.

9. Which item(s) is not part of a class in a UML class diagram? D. Comments

10. Which item is not one of the three kinds of relationships a class diagram can have? D. Specialization

11. In a class diagram, what does a line with an arrow from one class to another denote? A. Attribute visibility

12. What is not a type of visibility between objects? B. Method

13. Which statement is not true about state machine and state diagrams? D. State diagrams are used in situations for which all or most of the events represent the completion of internally generated actions (that is, procedural flow of control).

14. Which of the following UML diagrams may be best suited for a business analyst? C. Use case

15. In a UML class diagram, Private, Protected, and Public attributes are shown by which one of

the following sets of symbols? D. –, #, +

***第四单元Chapter 4: Applicability of JEE Technology***

Explain the JEE Architecture and System Requirements

1. Which of the following is not true about JEE containers? C. An application client container manages the execution of all application client components for a single JEE application. Application clients and their accompanying containers run on

the JEE server.

2. Which statement is not true when discussing the EJB tier? B. The Enterprise JavaBeans (EJB) tier does not host system-level services (such as transaction management, concurrency control, and security); they are hosted on the EIS tier.

3. Which of the following is not true when put in the context of JEE transaction processing? D. When using a compensating transaction, it is always possible to undo the effect of a committed transaction.

4. JEE uses the core patterns to enable flexible association of EJB classes with other components.

Which of the following is not used by JEE?( Decorator, factory, and proxy are core patterns to enable flexible association of EJB classes with other components.) C. Designer

5. Which statement is not true when discussing best practices? C. A session bean facade does not shield the client from complex entity bean relationships and manages workflow on a client’s behalf.

6. In which of the following cases would an application not necessarily benefit from the use of Enterprise JavaBeans?

A. Small-scale deployment D. No transactional requirements

7. The JEE platform uses a multi-tiered distributed application model; which of the following is not considered a tier in this architecture? D. Security tier

8. Which of the following are “best practices” for large distributed systems? B. Coding data manipulation logic, format and display code, and user event handling together can make application maintenance simple.

9. Which of the following are not benefits of using the MVC best practice? D. Simplifies database design because only the View components access the database

10. Which of the following is not true of using the MVC best practice? C. The Model determines how the results should be displayed.

11. Which of the following are not components of the MVC?( as Model, View, and Controller are the components of the MVC pattern.)B. Calculator

12. Which of the following is not true of the MVC?( The Model, not the Controller, maintains the application state.) D. The Controller maintains the application state.

13. Which of the following is not typically considered a threat to enterprise-critical assets?(A misappropriation that does not compromise availability is not typically considered a threat to enterprise-critical assets.) D. A misappropriation that does not compromise availability

14. Which of the following are not true about screen scrapers? C. Changes to legacy UI have little or no impact on the new GUI.

D. Screen scraping is best used when the legacy clients have loose coupling with other tiers.

15. If the telephone company were to rewrite its existing legacy code using newer JEE technology, what technology would you choose to accommodate both the block purchase and the individual query? D. Session EJBs for both

16. Your company’s web site offers the customers price comparisons on a variety of different products. You are in charge of converting the web-based solution over to the appropriate JEE technology. Which of the following should you use? A. JSP, servlets

17. Regarding the JEE EIS integration, which of the following statements is not true? B. The JEE Connector architecture provides a Java solution to the problem of connectivity between the many application servers and only new EISs, not those already in existence.

18. Regarding the JEE EIS integration contracts, which of the following statements is not true? D. A Transaction Management contract does not support transactions that are managed internal to an EIS resource manager without the necessity of involving an external transaction manager.

19. Which is the following is not true about enterprise applications and integration? D. Enterprise application development is about building an enterprise application from scratch.

***第五单元UML***

1. Which of the following is not a benefit of using Design Patterns? D. They provide solutions to totally novel problems.

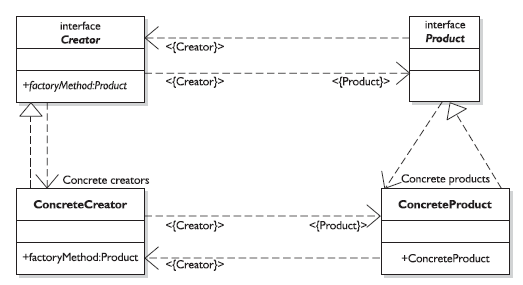
2. The Factory Method design pattern is useful when a client must create objects having different. A. Subclasses

3. What design pattern limits the number of instances a class can create D. Singleton

4. Iterators are useful when dealing with which of the following types of classes? B. Collection

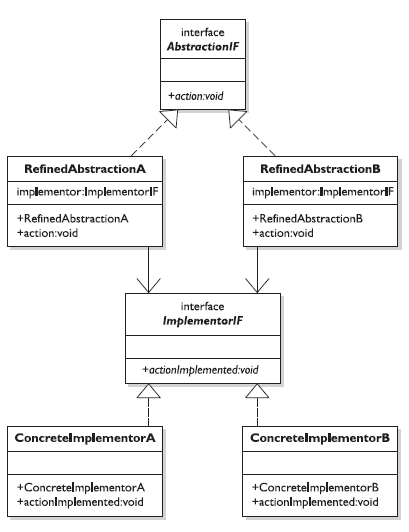
5. What is the Abstract Factory pattern also known as? A. Kit

6.Which pattern is shown in the diagram? B. Factory Method



7. What pattern is also known as Virtual Constructor? D. Factory Method

8.Which pattern is shown in the diagram? C. Bridge



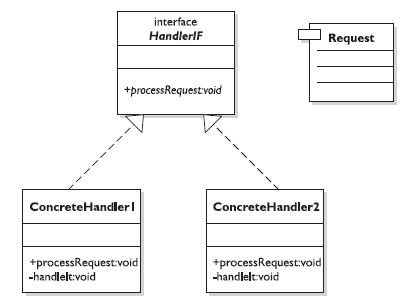
9. What is the Adapter pattern also known as? B. Wrapper

10.Which pattern is shown in the diagram? B. Facade



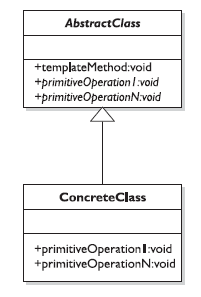
11. What pattern is also known as Handle/Body? D. Bridge

12.Which pattern is shown in the diagram? A. Chain of Responsibility



13. What is the Decorator pattern also known as? A. Wrapper

14.Which pattern is shown in the diagram? A. Template Method



15. What pattern is also known as Surrogate? C. Proxy

16. What is the Command pattern also known as? A. Action B. Transaction

17. The Command design pattern \_\_\_\_\_\_\_\_\_\_\_\_ a request in an object. B. Encapsulates

18. Which of the following elements are parts of the Gang of Four (**GoF**) Design Pattern format? C. Consequences D. Intent

19. The Decorator pattern appears in which of the following Java packages? A. java.io B. java.awt

20. Which Java package contains classes that implement the Iterator design pattern? B. java.util

21. What two methods are defined by the Enumeration interface? A. hasMoreElements()

***第六单元 Chapter 6: Legacy Connectivity（RIM）***

1. For a system consisting of exclusively Java objects, which distributed technology would be most appropriate for communication? B. RMI

2. Which of the following are true about the Interface Definition Language (IDL)? A. Interfaces between CORBA objects can be specified using IDL. C. Interfaces described in IDL can be mapped to other programming languages.

3. An object that implements the interfaces java.rmi.Remote and java.io.Serializable is being sent as a method parameter from one JVM to another. How would it be sent by RMI? B. RMI will send the stub of the object.

4. The RMI compiler rmic runs on which of the following files to produce the stub and skeleton classes? B. On the remote service implementation class file

5. Which distributed object technology is most appropriate for systems that consist of objects written in different languages and that execute on different operating system platforms? B. CORBA

6. Which of the following are used by Java RMI? A. Stubs B. Skeletons

7. Which of the following is not a tier of a three-tier architecture? C. Security

8. Which of the following is not true about RMI ? C. The RMI Registry is used to generate stubs and skeletons.

***第七单元EJB***

1. Which of the following is not true about Enterprise JavaBeans (prior to EJB 3.0) objects? C. The bean implements either the EntityBean interface or the SessionBean interface but need not implement all the methods defined in the remote interface.

2. Which of the following is true about Enterprise JavaBeans (EJB 3.x) objects? A. The home interface is no longer required.

3. Which statement is not true when contrasting the use of entity beans and JDBC for database operations? B. The bean managed entity bean functionally replaces the JDBC API.

4. Suppose that the business logic of an existing application is implemented using a set of CGI programs. Which Java technologies can be used to implement the CGI programs as a Java-based solution? C. Enterprise JavaBeans D. Servlets

5. Which of the following is not true about Enterprise JavaBeans (prior to EJB 3.0) session bean objects?

A. A session bean can be defined without an ejbCreate() method.

6. If you try to create an (prior to EJB 3.0) CMP-based entity bean for a table that does not have a primary key, which of the following statements is not true? A. You cannot create CMP entity beans without a database primary key.

7. Can you update the primary key field in a CMP entity bean (prior to EJB 3.0)? A. No; you cannot update the primary key field in a CMP entity bean.

8. In an application with several stateless session Enterprise JavaBeans (prior to EJB 3.0), in terms of performance ramifications of storing the remote reference to a stateless session bean, which of the following statements is least accurate? C. The cost for a remote lookup on a stateless session bean is insignificant and generally does not justify using a handle (SSB reference) to access the bean.

9. Which distributed object technology is most appropriate for systems that consist entirely of Java objects? A. RMI

10. Which distributed object technology is most appropriate for systems that consist of objects written in different languages and that execute on different operating system platforms? B. CORBA

11. Which of the following are used by Java RMI? A. Stubs B. Skeletons

12. Which of the following is not a tier of a three-tier architecture? C. Security

13. Which of the following Java technologies implements transaction management? B. JTS

14. Which of the following is not true when discussing application servers and web servers? C. A web server does not support caching, clustering, and load balancing.

15. Which statement is not true when discussing serialization in EJB? D. Serialization is not possible in EJB.

16. Which of the following are true about EJB components, containers, and application servers? A. Components run in containers. B. Containers are hosted by application servers.

17. Which objects would you find in an Enterprise JavaBeans (prior to EJB 3.0) directory service? A. An EJB home interface

18. Containers and servers have the same function. What is the difference between an Enterprise JavaBeans container and an Enterprise JavaBeans server? A. Containers run within servers.

19. Which of the following Enterprise JavaBeans (prior to EJB 3.0) CMP entity bean methods are used by the container to alert the bean when its state is synchronized with the database? A. ejbLoad() B. ejbStore()

20. What happens if remove() is not invoked on a stateful Enterprise JavaBeans (prior to EJB 3.0) session bean? D. The bean is removed after the session time-out has been reached.

21. With respect to stateful Enterprise JavaBeans (prior to EJB 3.0) session beans, which of the following statements is not true? A. Stateful session beans support instance pooling.

22.Select entity listener life cycle callback method annotations.

A. @PrePersist B. @PostPersist C. @PostRemove

***第八单元Messaging***

1. Which of the following are characteristics of publish/subscribe message model? B. Subject-based addressing. C. Location-independent publishers.

2. Which of the following are valid methods for a TopicSubscriber? A. receive() B. receiveNoWait()

3. What are the types of messaging models supported in JMS? A. Point-to-point D. Publish/subscribe

4. What is a message digest? A. A digital fingerprint value that is computed from a message, file, or byte stream

5. Which of the following scenarios are suitable for publish/subscribe messaging model? D. It is used to authorize a user ID and password.

6. What deliver modes are available in JMS? A. PERSISTENT B. NON\_PERSISTENT

7. Which of the following are valid message acknowledgment types? A. AUTO\_ACKNOWLEDGE B. CLIENT\_ACKNOWLEDGE C. DUPS\_OK\_ACKNOWLEDGE

8. Which of the following are not valid message body formats? D. StringMessage

9. Which of the following are not valid JMS objects? C. MessageViewer

10. Which of the following would not be used in a client application performing point-to-point messaging? A. Topic

11. Which of the following are advantages of asynchronous messaging architectures?

A. Better use of bandwidth B. Supports load balancing D. Scalability

12. Which of the following statements are true for asynchronous messaging? A. It decouples senders and receivers. B. It can increase performance. C. It is better suited to smaller message sizes.

13. Which method must be called to receive messages asynchronously? D. The onMessage method

14. Which of the following are not features of asynchronous messaging? B. A message must be acknowledged before the producer can send another.

15. Which method must be called to receive messages synchronously? A. The receive method

16. Which of the following cases are better suited to synchronous messaging? B. Credit card authorization D. Validation of data entered

17. Which of the following are features of synchronous messaging? A. Both parties must be active to participate. B. Messages must be acknowledged before proceeding.

18. Which of the following are not features of synchronous messaging? A. Both parties must be active to participate. C. Message must be acknowledged before proceeding to the next.

19. Which of the following scenarios are not suitable for publish/subscribe messaging model? A. Sending an instant message B. Sending an order to another system D. Sending a transaction to another system

20. What method must be implemented to receive messages in a message-driven bean (MDB)? B. The onMessage method

**补充知识点**

**EntityManager是负责管理Entity的对象**

实体管理器与Hibernate Session，JDO PersistenceManager等功能类似 控制实体的生命周期

persist（） - 将一个实体插入到数据库中 remove（） - 从数据库中删除一个实体 merge（） - 同步分离的实体的状态 refresh（） - 从数据库重新加载状态

**实体生命周期**

jpa 实体生命周期有四种状态

New：瞬时对象，尚未有id，还未和Persistence Context建立关联的对象。

Managed：持久化受管对象，有id值，已经和Persistence Context建立了关联的对象。

Datached：游离态离线对象，有id值，但没有和Persistence Context建立关联的对象。

Removed：删除的对象，有id值，尚且和Persistence Context有关联，但是已经准备好从数据库中删除

Managed状态下的数据保存，更新以及删除数据下的Removed状态，数据都不会立即更新到数据库，只有当你事务提交或者em.flush()，才会立即更新到数据库。

Datached的状态，可以调用em.merge()方法，这个方法会根据实体类的id来更新数据库数据，这时实体类变成了Managed状态。

|  |  |  |  |
| --- | --- | --- | --- |
| 状态名 | 作为java对象存在 | 在实体管理器中存在 | 在数据库中存在 |
| New | yes | no | no |
| Managed | yes | yes | yes |
| Detached | no | no | no |
| Removed | yes | yes | no |

**EJB使用事务方式**

在EJB中有两种使用事务的方式。第一种方式通过容器管理的事务，叫CMT(Container-Managed Transaction)，另一种通过Bean管理的事务叫BMT(Bean-Managed Transaction)。

CMT的最大优点是简单、方便，但它不够灵活。总是在业务方法的开始、结束处标记事务边界，依靠容器来决定何时开始、提交或者回滚。与CMT相对的是，BMT事务管理可以由开发者来决定业务方法中的事务边界。

**事务传播属性**

Required：用该属性标注的方法或组件总是在事务中运行。如果客户端已经在事务中，则在原事务中运行；如果没有事务，则开启一个新事务，在其中运行。

Requires\_New：方法或组件总是在新开启的事务中运行。如果客户端已经在事务中，则首先将原事务挂起，然后新开启一个事务，在其中运行，新事务结束之后，原来事务从挂起点继续执行；如果没有事务，则开启一个新事务，在其中运行。

Supports：和 Required 属性的不同点是，在没有事务的环境中不会开启一个新事务；如果存在事务的话则加入其中运行，这点和 Reuqired 相同。

Not\_Supported：如果事务已经存在的情况下，则原来的事务要挂起，然后调用标注该属性的方法或组件，调用结束之后，继续原来的事务；无事务环境中调用的时候，不开启新事务，这点和 Supports 相同。

Mandatory：调用标注该属性的方法或组件的客户端，必须已经在事务中，如果不在事务中则会抛出异常；如果已经在事务中，则加入原来事务运行。和 Required 不同的是，该属性不会自动开启新的事务；

Never：用 Never 属性标注的方法或组件，不能在事务中运行。如果调用该方法或组件的客户端已经在事务中，则抛出异常。

**什么是JPA?**

**JPA（Java Persistence API ）**Java EE标准的O/R映射框架 用于启用透明POJO持久性的对象/关系映射框架

**ORM是什么？**

对象关系映射（Object Relational Mapping，简称ORM）是一种为了解决面向对象与关系数据库存在的互不匹配的现象的技术。 简单的说，ORM是通过使用描述对象和数据库之间映射的元数据，将java程序中的对象自动持久化到关系数据库中。本质上就是将数据从一种形式转换到另外一种形式。

**# 默认Entity的属性加载方式都是即时加载（EAGER）;**

**# 两种加载方式：LAZY 惰性加载 Set** **EAGER 即时加载（默认）Single** **示例 @Basic(fetch=FetchType.EAGER)**

**关联指定列（@JoinColumn）**

# @JoinColumn用于注释表中的字段，与@Column不同的是它要保存表与表之间关系的字段；# 默认情况下，关联的实体的主键一般用来做外键的。如果不想用主键作为外键，则需要设置referencedColumnName属性，如：

@JoinColumn(name="address\_id",referencedColumnName="ref\_id")

**主键关联（@PrimaryKeyJoinColumn）**

**EJB:Enterprice JavaBeans：**

EJB是一个用于分布式业务应用的标准服务端组件模型。采用EJB架构编写的应用是可伸缩的、事务性的、多用户安全的。采用ejb编写的这些应用，可以部署在任何支持EJB规范的服务器平台，如jboss、weblogic等。

好处：EJB为我们提供了很多在企业开发中需要使用到的服务,如事务管理/安全/持久化/分布式等,因为这些服务由容器提供,无需我们自行开发,这样大大减少了我们的开发工作量.另外EJB提供了强制式分层解耦设计方法

如果你的应用不需要分布式能力，确实没有必要使用ejb，因为ejb设计的初衷是用于分布式场合，而spring一开始就没有打算提供分布式功能。所以两者看似有竞争的关系，但实际上两者的偏重点不一样，像ejb比较适合用于大型企业，因为大型企业一般都会存在多个信息系统，而这些信息系统又相互关联。为了避免业务功能重复开发，实现最大程度的重用，有必要把业务层独立出来，让多个信息系统共享一个业务中心，这样应用就需要具备分布式能力。

**EJB中的三种bean**

•会话bean(session bean)：负责与客户端交互,是编写业务逻辑的地方,在会话bean中可以通过jdbc直接操作数据库,但大多数情况下都是通过实体bean来完成对数据库的操作.

•实体bean(entity bean)：它实际上属于java持久化规范(简称JPA)里的技术, JPA的出现主要是为了简化现有的持久化开发工作和整合ORM技术，结束现在Hibernate、TopLink等ORM框架各自为营的局面。

•消息驱动bean(message-driven bean)：它是专门用于异步处理java消息的组件.具有处理大量并发消息的能力.

**什么是持久化?**

在程序退出后信息可以长期保存的一种数据存储技术。

**JSP和SERVLTE**

•Servlet定义：部署在java的Web服务器上的组件，整个java服务端程序都构建在Servlet之上，以多线程方式提供服务，具有效率高，可扩展，可移植的特点。

•JSP的定义：所有JSP页面在执行过程中，都会被变异成Servlet，由JSP页面生成的Servlet叫做“JSP”页面实现类，可以把JSP理解为嵌入了Java语句的HTML页面文件。

•Servlet与JSP的关系：在页面展现上，Servlet通过输出HTML语句来实现动态页面，如果SUN不退出JSP标准，那么Java的Web页面开发（动态页面的交互过程）将十分痛苦。

**JSP生命周期中所走过的几个阶段**

转译时期:将JSP网页转译为Servlet源代码 .java.

编译时期:将Servlet 源代码 .java编译成 Servlet类 .class.

•编译阶段：servlet容器编译servlet源文件，生成servlet类

•初始化阶段：加载与JSP对应的servlet类，创建其实例，并调用它的初始化方法

•执行阶段：调用与JSP对应的servlet实例的服务方法

•销毁阶段：调用与JSP对应的servlet实例的销毁方法，然后销毁servlet实例

**动态包含和静态包含**

＜%@ include file=”relativeURI”%＞ 可以叫作静态include（静态包含），是jsp指令中的一种。

＜jsp:include page=”relativeURI” flush=”true” /＞叫作动态include（动态包含），是jsp动作的一种

区别：执行时间上：

•＜%@ include file=”relativeURI”%＞ 是在翻译阶段执行

•＜jsp:include page=”relativeURI” flush=”true” /＞ 在请求处理阶段执行.

引入内容的不同：

•＜%@ include file=”relativeURI”%＞引入静态文本,在JSP页面被转化成servlet之前和它融和到一起，.先包含，后编译，不会检查所含文件的变化，适用于包含静态页面

•＜jsp:include page=”relativeURI” flush=”true” /＞引入执行页面或servlet所生成的应答文本. 被包含的文件先编译，后包含进来，然后显示。

**表单传值的方式**：传值方式一般通过HTML中form表单的method指定，分为GET方式（**默认**）和POST方式

**servlet生命周期？**

加载和实例化；在第一次请求Servlet时，Servlet容器将会创建Servlet实例；

初始化；Servlet容器加载完成Servlet之后，必须进行初始化，此时，init方法将被调用；

Servlet初始化之后，就处于响应请求的就绪状态，此时如有客户端请求发送，就会调用Servlet实例的service方法，并且根据用户的请求方式，调用doPost或者doGet方法；

最后，Servlet容器负责将Servlet实例进行销毁，调用destroy方法实现；

**jsp九大隐式对象**

JSP中一共预先定义了9个这样的对象，分别为：request、response、session、application、out、pagecontext、config、page、exception

<**jsp:useBean** id="id"scope="page|request|session|application" typeSpec/>

<**jsp:setProperty** name="beanName" propertyDetail />其中,propertyDetail可以是以下中的一个:

1,property="\*"

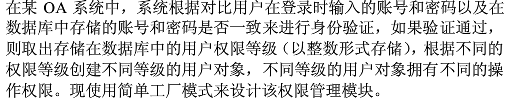
2,property="propertyName" param="parameterName"

3,property="propertyName"

4,property="propertyName" value="property value"

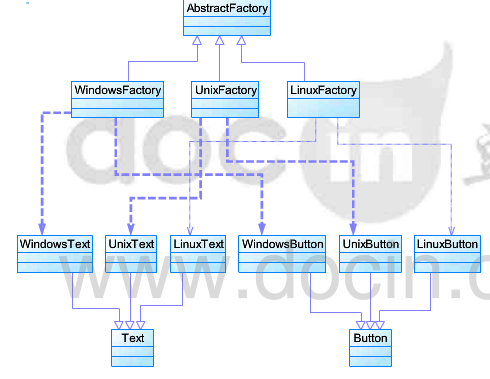
<**jsp:getProperty** name="beanName" property="propertyName"/>

**简单工厂模式例子**





**抽象工厂类图**



**单词对照翻译**

accidental 偶然性

accommodate 兼顾

asynchronous 异步

browser 浏览器

client 客户

cohesion 依附性

components 组件

concept 概念

concurrency 并发

conformity 一致性

container 容器

core 核心

coupling耦合性

decomposition 分解

decorator 装饰

distributed 分布式

eliminated 消除

enterprise 企业

enumerate 列举

essential 必然性

evolvability 可扩展性

exclusively 专门的

filters 过滤器

fundamental 基本的

granularity 粒度

guidelines 指南

individual 个人的

infrastructure 基础设施

initial 最初的

interaction交互

legacy 遗留

maintainability 可维护性

pattern 模式

persistence 持久化

principal 主要的

property属性

properties 需求

purchase 购买

runtime qualities性能

rpc 远端过程调用

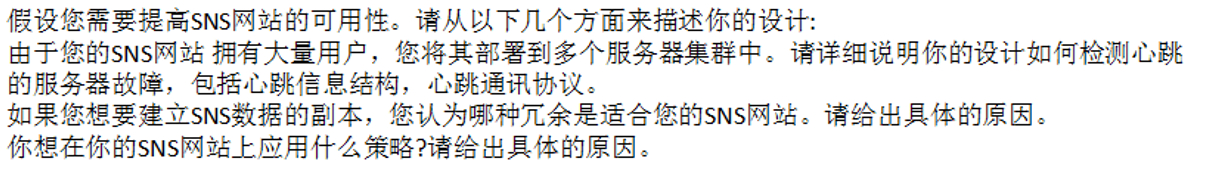
rmi 远端方法调用

surrogate 代理

synchronization 同步

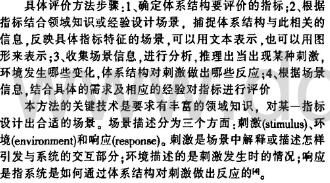
transaction 事物

unique 唯一的



心跳检测：在服务端保存下来所有客户端的状态信息，然后等待客户端定时通过TCP协议发送心跳包访问服务端，更新自己的当前状态，如果客户端超过指定的时间没有来更新状态，则认为客户端已经宕机或者其状态异常。

编码冗余：是指一组数据携带的平均信息量。正因为SNS网站各种多媒体数据中存在着各种各样的冗余，所以多媒体数据是可以被压缩的。



**架构风格：**这些架构风格分别适用于特定领域：

**通信** SOA，消息总线，管道和过滤器

**部署** 客户端/服务器，三层架构，N层架构

**领域** 领域模型，网关

**交互** 分离表现层

**结构** 基于组件的架构，面向对象，分层架构



**经验原则**

