# 软件构件与中间件技术

## 目录



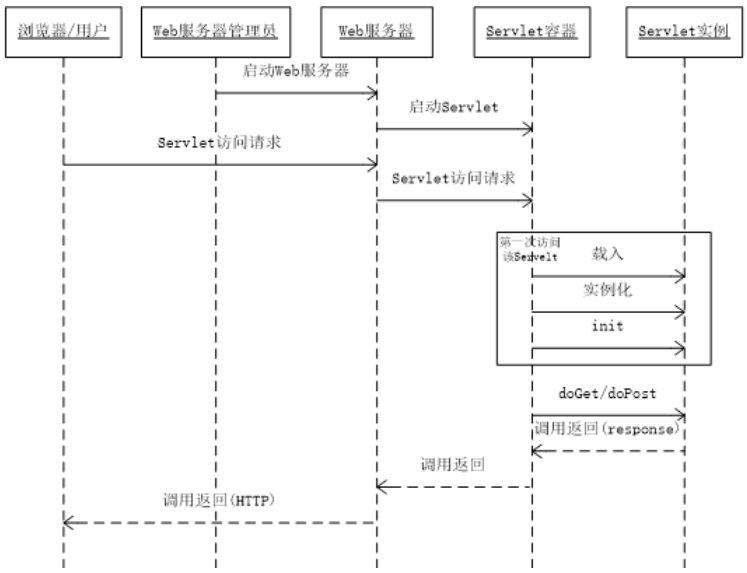
## 知识点

### servlet

#### 什么是servlet

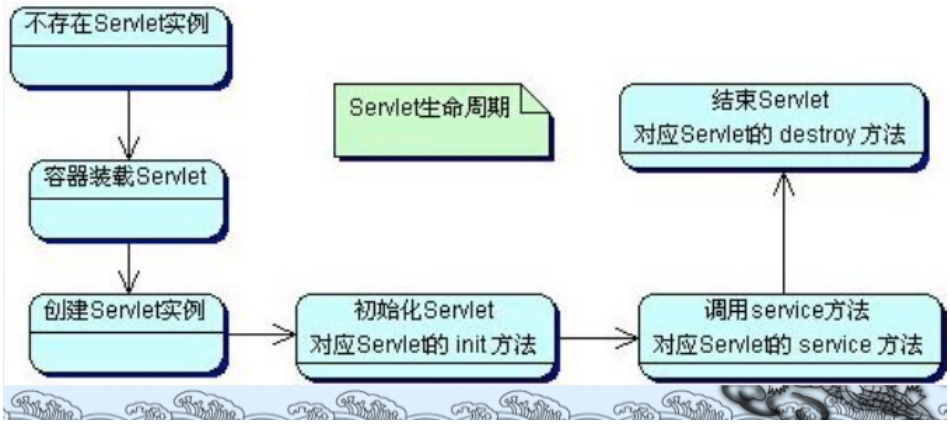
servlet 是一种 Java 程序类，是一种用来扩展以请求-应答模式运行应用程序的服务器的能力的一种 Java 程序。

#### 工作原理




#### 生命周期

Servlet的运行状态完全由Servlet容器维护和管理，一个Servlet的生命周期一般有初始化、提供服务和销毁三个阶段。




## 主要接口和类

- 主要接口
  - Servlet接口
  - ServletRequest、ServletResponse接口
- 主要类
  - GenericServlet抽象类
  - HttpServlet抽象类
  - HttpServletRequest、HttpServletResponse接口

## 共享变量

在servlet中进行变量的共享可以通过Servlet容器中存在的ServletContext 、HttpSession 和HttpServletRequest 的实例来实现。

- ServletContext
  范围最大，应用程序级别的，整个应用程序都能访问。
- HttpSession
  会话级别的，在当前的浏览器中都能访问。
- HttpServletRequest
  范围最小，请求级别，请求结束，变量的作用域也结束。

## 读写文件

在Servlet类中可以使用Java提供的输入/输出流来完成对服务器上文件的读取和写入。

- 读文件
  一般利用File、FileReader和BufferedReader类的组合来完成。
- 写文件
  通常使用File、FileWriter和BufferedWriter的组合来完成。

## 访问数据库

使用Servlet访问数据库也和普通的Java程序访问数据库的方法相同，都是利用JDBC提供的接口来完成数据库的连接以及数据库的相关操作。
其中常用到的类是Connection类、Statement类和ResultSet。

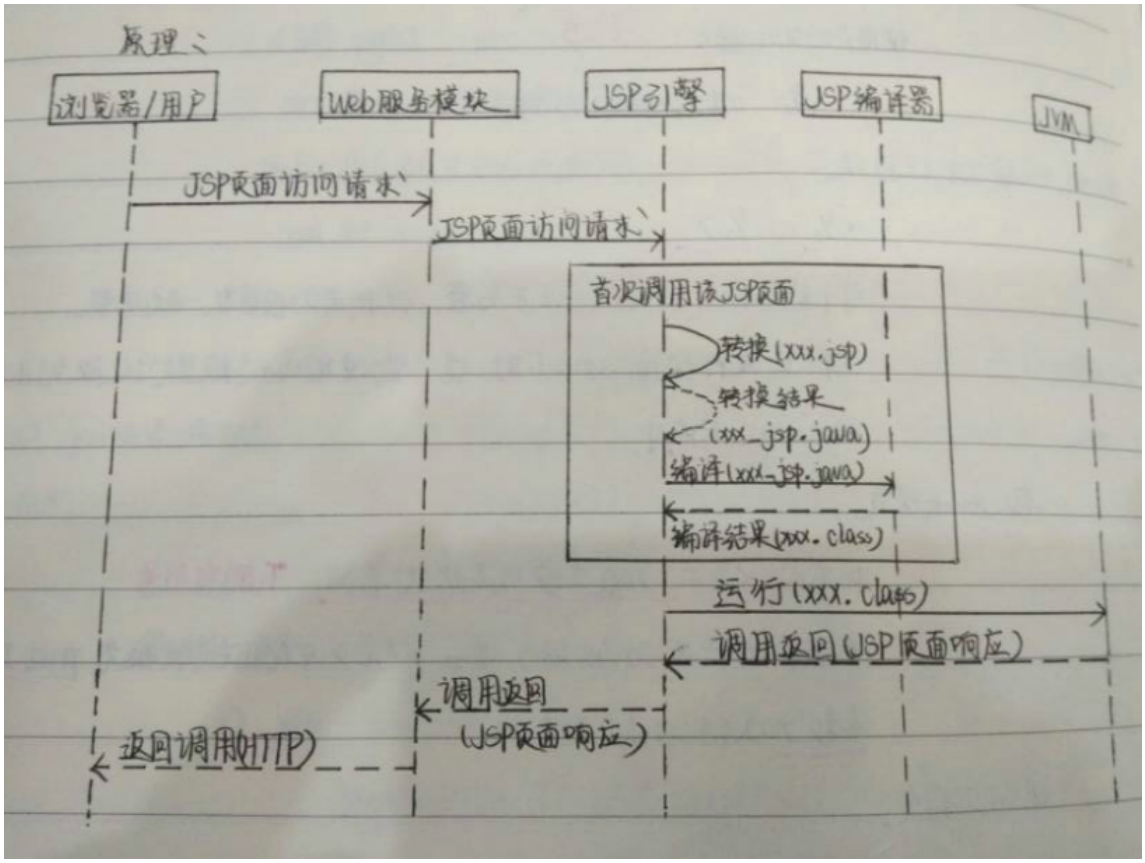
## ServletConfig

每一个ServletConfig对象对应着一个唯一的Servlet。Servlet容器在调用init方法时，把servletConfig对象当做参数传递给servlet对象。 servletConfig 包含这个Servlet所有配置信息。

# JSP

## JSP 的概念和原理

- JSP 的概念
  JSP是Java Server Page的缩写，是Sun公司倡导、许多公司参与一起建立的一种动态网页技术标准。
- JSP 的原理




## JSP的基本构成

JSP声明、JSP程序块、JSP表达式、JSP指令、JSP动作、JSP注释。

## 内置对象

内置对象是JSP规范所定义的由Web容器实现和管理的一些在JSP页面中都能使用的公共对象。

## JavaBean

- 什么是 JavaBean
  JavaBean将功能、处理、值、数据库访问和其他任何可以用java代码创造的对象进行打包。
- JSP 如何调用和使用 JavaBean
  在JSP中可以使用jsp:useBean、 jsp:setProperty、jsp:getProperty这三个动作来完成对JavaBean的调用。

## Servlet与JSP的比较

- 相似之处

都可以生成动态网页。

- 不同之处
  JSP的优点是擅长于网页制作，生成动态页面比较直观，缺点是不容易跟踪与排错。
  Servlet是纯Java语言，擅长于处理流程和业务逻辑，缺点是生成动态网页不直观。

# JSF

## JSF 概述

JSF是一种基于组件的框架。

- 特点
  严格遵循MVC的框架，有丰富的可重用的服务端组件。

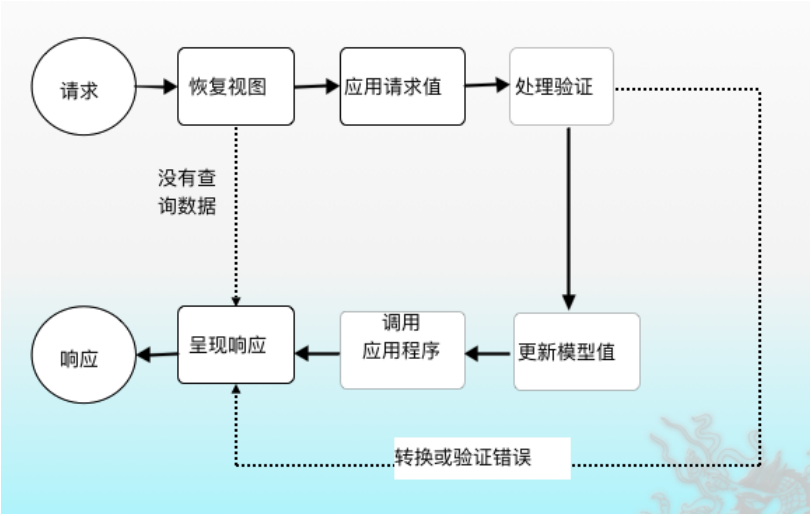
## 托管 Bean

- 定义
  JSF使用JavaBean 来达到业务逻辑与视图分离的目的，称为托管Bean，其作用是在真正的业务逻辑Bean及UI组件之间搭起桥梁。
- 配置托管 Bean
  - 使用标注配置托管Bean
  - 使用XML配置托管Bean
- 生命周期
  使用@PostConstruct和@PreDestroy标注，可以指定在创建bean之后以及bean离开作用域之前被自动调用的方法。由于托管Bean是被容器创建和销毁的，因此，这两个标注为我们监控Bean的生命周期提供了便利。
- EL表达式
  JSF提供了一种在 Web 应用程序页面中使用的表达式语言 (JSF EL)，用来访问位于页面 Bean 以及其他与 Web 应用程序关联的 Bean（如会话 Bean 和应用程序 Bean）中的属性或者方法。

## 导航

- 静态导航和动态导航
  静态导航：当用户单击按钮提交表单数据时，往往会转向某个页面，JSF将用户填写的表单内容将传输到服务器。
  动态导航：页面流不仅取决于用户单击了哪个按钮, 也取决于用户的输入。
- 重定向和非重定向
  重定向：重定向响应告诉客户端下一页面使用哪个URL。客户端然后生成到该URL的GET请求。
  非重定向：如果不使用重定向，原始的URL保持不变。

## JSF生命周期




### 转换和验证

JSF提供了转换和验证机制，可以对用户在页面输入的数据进行预先处理，从而可以确保传递到后台的数据的正确性。

### 事件处理

通常，可以在组件中注册事件处理程序（事件监听器）。

- JSF支持4种类型的事件
  - 值更改事件
  - 动作事件
  - 阶段事件
  - 系统事件

### JSF标签

JSF 2.0一共有6个标签库。

### 上下文和依赖注入

- 上下文
  上下文是指能够将无状态组件的生命周期和交互与定义良好的、可扩展的生命周期上下文绑定在一起的能力。
- 依赖注入
  在依赖注入的模式下，创建被调用者的工作不再由调用者来完成，而是由容器来完成，然后再注入调用者。因此依赖注入是指能够以类型安全的方式将组件注入到应用程序的能力。

## EJB

### 定义

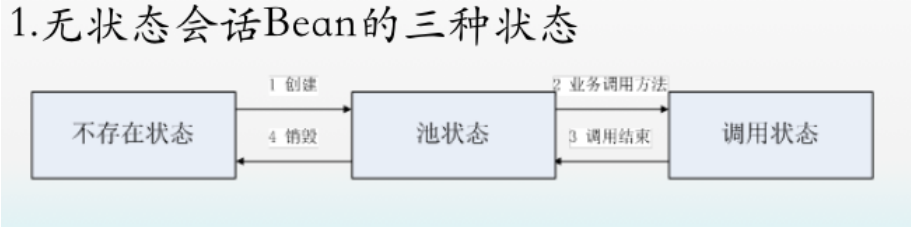
EJB 是Enterprise JavaBean的缩写，又称为企业Bean，是Sun公司提出的服务器端组件规范，它描述了在构建组件的时候所需要解决的一系列问题，如可扩展(Scalable)、分布式(distributed)、事务处理(Transcational)、数据存储(Persistent)、安全性(security)等。

### EJB 3.1组件类型

会话 Bean 和消息驱动 Bean。

### 会话 Bean

- 包括
  有状态会话 Bean、无状态会话 Bean、单例会话 Bean。
- 单例会话Bean的并发控制
  - 容器管理并发CMC
  - Bean管理并发BMC
- 单例会话Bean生命周期
  单例会话bean与无状态会话bean在大体上是一样的，只不过对于单例会话bean在实例池中只有一个实例存在，而对于无状态会话Bean来说，在实例池中存在多个实例。
- 无状态会话 Bean 生命周期

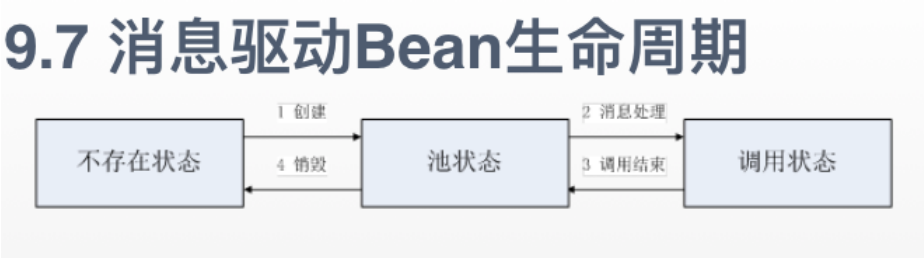



## JMS 和消息驱动 Bean

### MDB

消息驱动Bean是一种通过消息方式为外界提供服务的组件。

### MDB的生命周期




### MDB的生命事件

- PostConstruction事件
- PreDestroy事件

### JMS 消息接收

接收消息可以分为同步和异步接收。

- 同步接收
  同步接收使用MessageConsumer.receive()方法实现，如果消息可用，JMS服务器将这个消息返回，否则接收者一直等待消息的到来。
- 异步接收
  异步接收通过向MessageConsumer注册消息监听器对象javax.jms.MessageListener，实现队列中消息的异步接收，消息的处理在onMessage()方法中实现。

## JPA

### 概念

JPA全称为Java Persistence API，即Java 持久化API。基于Java 持久化的解决方案，旨在规范、简化Java对象的持久化工作。

### 如何获得实体管理器

- 容器托管
- 应用托管

# 填空&选择

## 填空

### Servlet 程序开发

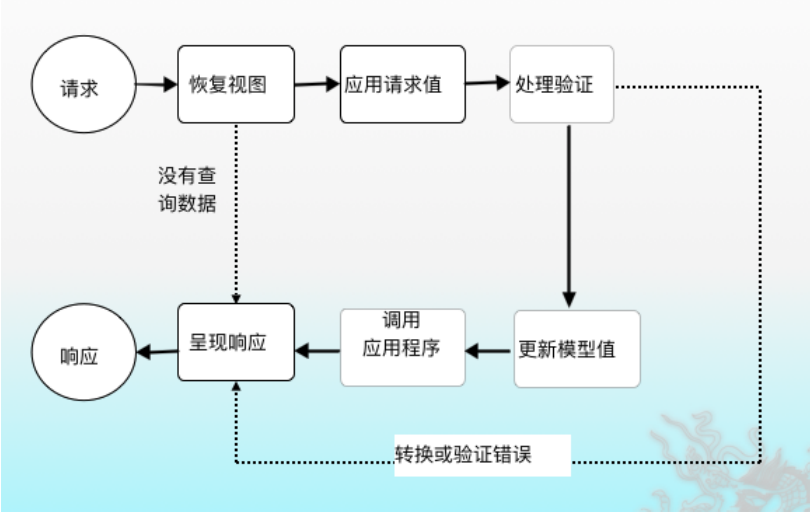
- http://localhost:8080/ServletTrain/HelloWorld 的路径构成
  **localhost:8080：服务器路径**
  **ServletTrain：项目路径**
  **HelloWorld：Servlet路径**
- HttpServlet 处理客户端请求的两种方法是 **doGet 和 doPost**。

### JSP 程序开发

- 在 JSP 中可以使用 jsp:useBean、jsp:setProperty、jsp:getProperty 这三个动作来完成对 JavaBean 的调用

### JSF

- **导航**设定JSF页面之间的跳转关系，可以为静态导航与动态导航两类。
- 每个组件对提交值执行数据类型转换和验证操作是在**处理验证阶段**进行的；调用注册的动作监听器和动作方法是在**调用应用程序阶段**进行的。




- JSF EL是**以#开始，将变量或运算式放置在｛与｝之间。**

### JDBC

- 使用**本地Java驱动程序**，对于独立的Java客户端程序是一个好的选择。不仅使得Java客户端不依赖于特定的环境，而且由于不需要经过中间环境的传递，可以获得更好的SQL语句处理速度。

## EJB 概述

- EJB 3.1中将会话Bean分成**有状态会话Bean、无状态会话Bean、单例会话Bean。**

## 会话 Bean

- 有状态会话 Bean 的生命周期包括：**不存在、准备就绪、挂起**三个状态。
- 无状态会话 Bean 中，创建本地接口使用 **@Local** 标注。

## JMS 与消息驱动 Bean

- JMS 提供**点对点方式和订阅/发布**两种类型的消息传递模型。

| | Java EE环境 | | Java SE环境 |
|---|---|---|---|
| | EJB容器 | Web容器 | |
| 应用托管的EntityManager | JTA(容器管理)，RESOURCE_LOCAL | JTA(非容器管理)，RESOURCE_LOCAL | RESOURCE_LOCAL |
| 容器托管的EntityManager | JTA(容器管理) | 不支持 | 不支持 |

## JPA

- **实体管理器**是 Java 实体对象与数据库交互的中介，它负责管理一组对应的实体，包括这组实体的CRUD ( 即 Create 创建、Read 读取、Update 更新和 Delete 删除的缩写)操作等。
- 在Java SE环境中，只能使用应用托管的EntityManager并且只能采 **RESOURCE_LOCAL(即本地的事务管理)**的方式管理事务。
- 若为双向关联，在保存实体关系的实体中(多的一方，主动的一方)，要配合使**用@JoinColumn或@JoinTable标注**；在没有保存实体关系的实体中(一的一方，被动的一方)，要使 **mappedBy**属性明确所关联的实体。
- EntityManager对象的事务管理方式有两种，分别为**JTA和RESOURCE_LOCAL，即Java Transaction AI方法和本地的事务管理。**
- 在 JPA 中，通常将数据库中的表映射为**类**，表中的字段映射为**类中的属性。**
- 如果想要指定查询语句中实体类名称是 Student，则对应的 JPA 实体映射代码是 **@Entity（name=“Student”）**。
-

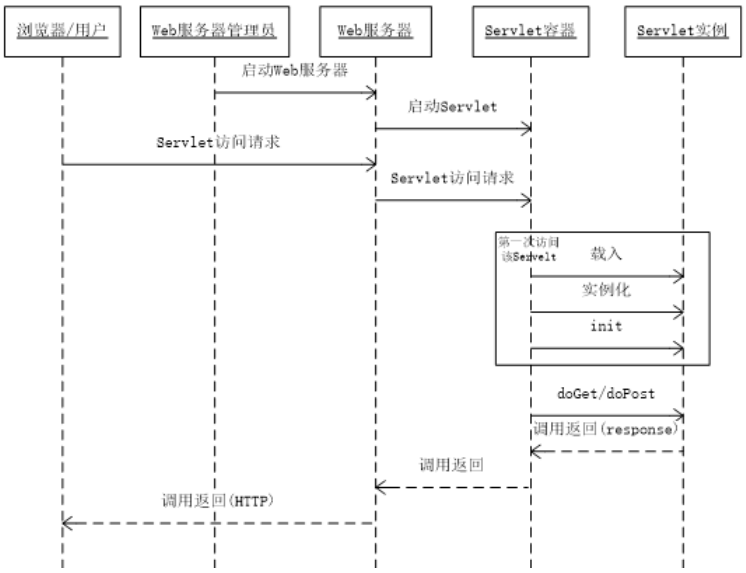
# 选择

## 软构件与中间件技术

- **中间件是一类软件**而不是某一种软件，**其作用是在系统软件和应用软件之间实现连接，实现通过不同的接口共享资源。**
- 通常按照中间件的作用，大致可以将中间件分为两大类：把支持单个的应用系统或解决一类问题的中间件称为**底层中间件**，把用于与各种应用系统关联，完成系统整合的中间件称为**高层中间件。**
- **中间件的特征：独立于系统、用于分布式环境、支持标准的协议和接口、可以实现应用之间的交互操作、具有网络通信功能。**

## Servlet 程序开发

- 在一次客户端访问服务器端 Servlet 的访问请求过程中，一定会被调用的函数是 **service()。**



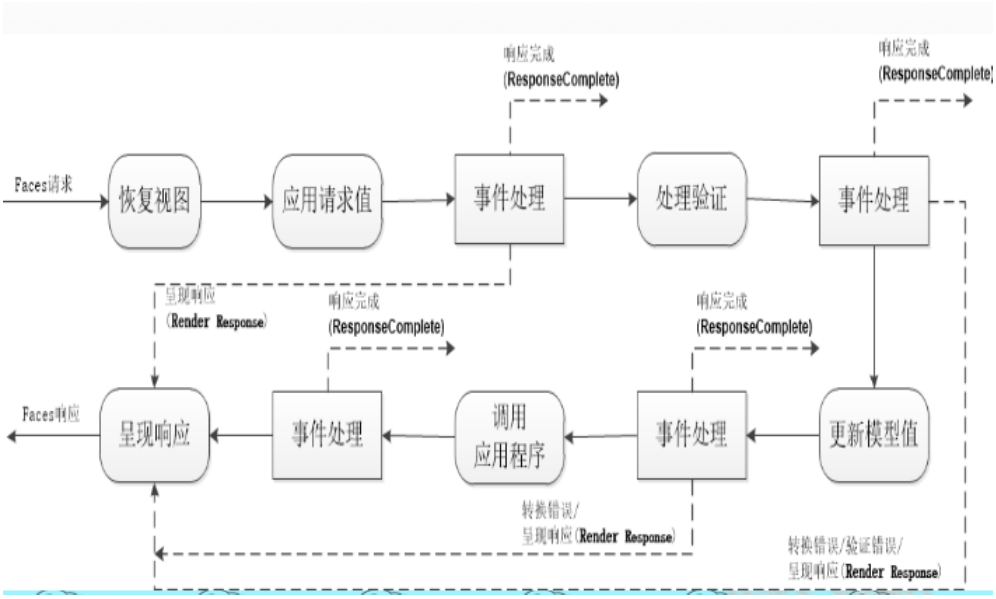

- Servlet 程序的入口是 **init()**。

## JSP 程序开发

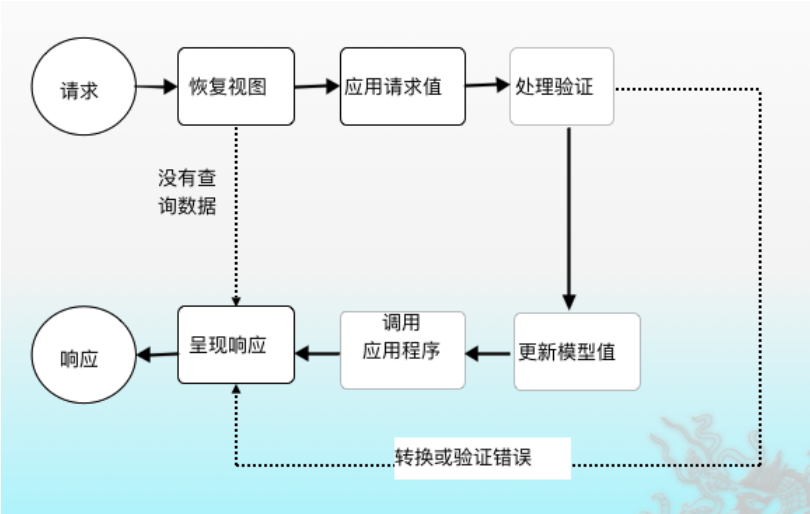
- 用来获取已被创建 JavaBean 组件属性值的语句是**jsp:getProperty**。

## JSF

- UserBean 是一个托管 Bean，在 JSF 页面中，要访问该类型的实例的 name 属性，**#{**UserBean.name**}**。
- 事件处理




- 当对 JSF 页面进行刷新时，**请求作用域**中的托管 Bean 不会持续存在。
- JSF 请求生命周期




## JNDI

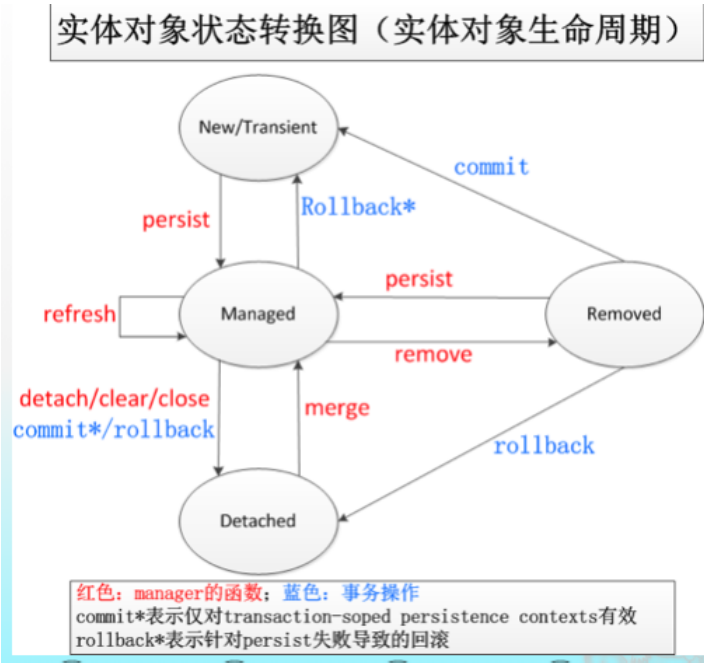
- 在 Jboss 服务器上，如果想要通过 JNDI 远程访问的方式获得 JNDI 对象，那么应该将该 JNDI 对象绑定到 **java:jboss/exported** 命名空间中。

## EJB 概述

- EJB 3.1中将会话Bean分成**有状态会话Bean、无状态会话Bean、单例会话Bean。**
- EJB3.1的bean**可以不需要接口。**
- 单例会话 Bean 提供了和无状态会话 Bean 类似的功能，区别是单例会话 Bean 在每个应用程序中只有一个实例，而无状态会话 Bean 会有一个实例池。
- 判断访问 EJB 方式是本地访问还是远程访问取决于 EJB 访问客户端和 EJB 是否位于**同一个 JVM。**

## JPA

- 瞬时状态(New/Transient)，托管状态(Attatched/Managed)，游离状态(Detached)，销毁状态( Removed )




- **一对一关联通常使用外键关联，且需要配合@JoinColumn注释使用。**
- 对于多对多的实体关系，在设计表结构时**只能采用表关联的方式。**
- 对一对多关联，在关系数据库中**既可采用外键的方式也可采用关联表的方式进行存储。**
- **“多”方实体类为关系主动方，即负责外键记录的更新，“一”方为被动方，即没有权力更新外键记录。**
- 在 JPA 中，新生成的实体对象 a 处于**瞬时状态**，在实体管理器对象执行了 persist() 操作之后，实体对象 a 将变为**托管状态。**

### JPQL

- JPQL是一种简单的类似于SQL的基于字符串的语言，用于查询实体，以及实体间的关系。
- JPQL操作的是实体类，不是数据库定义的物理模型。

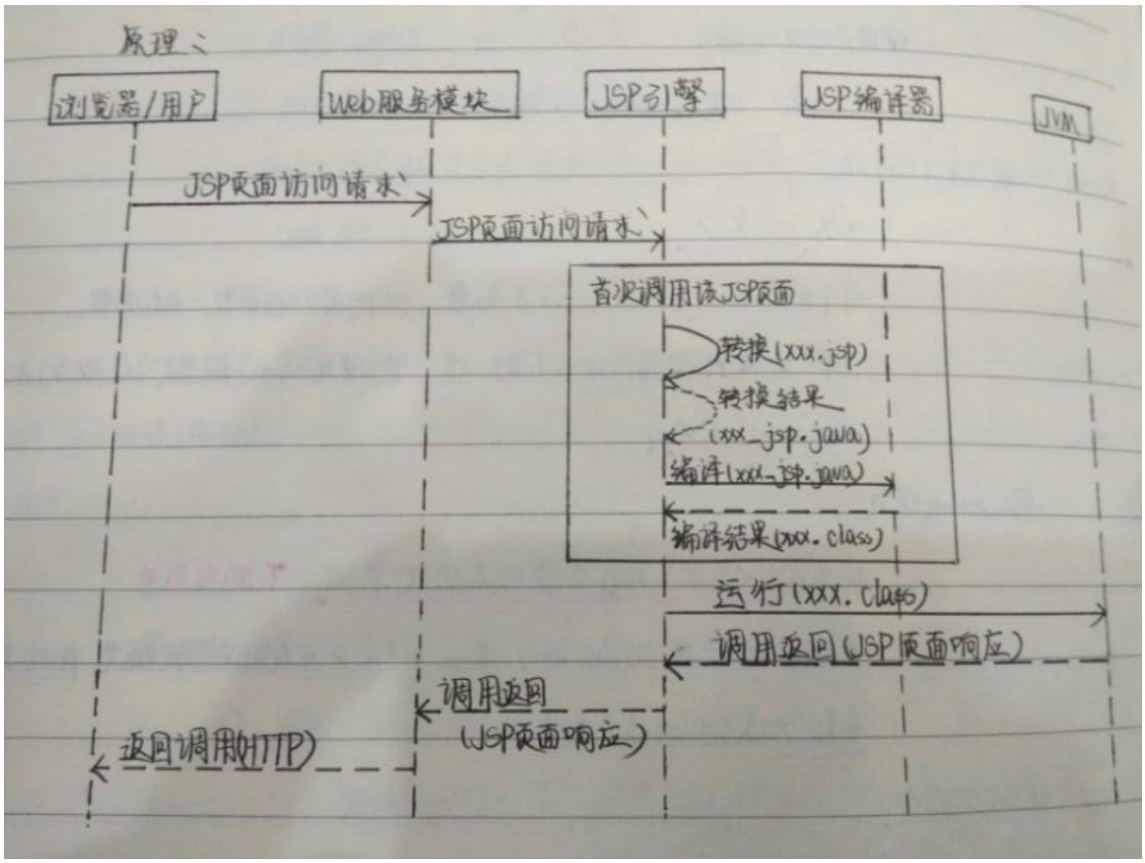
### Web Service

- SOAP（Simple Object Access Protocol，简单对象访问协议）是一种轻量的、简单的、基于 XML 的协议,它被设计成在 Web 上交换结构化的信息。
- WSDL（Web Service Description Language，Web Service描述语言）是一种基于XML的服务描述文档，用于描述Web Service及其函数、参数和返回值。
- UDDI （Universal Description, Discovery, and Integration，统一描述、发现和集成）。UDDI是一种目录服务，提供了Web Service的注册和搜索机制。

## 简答题

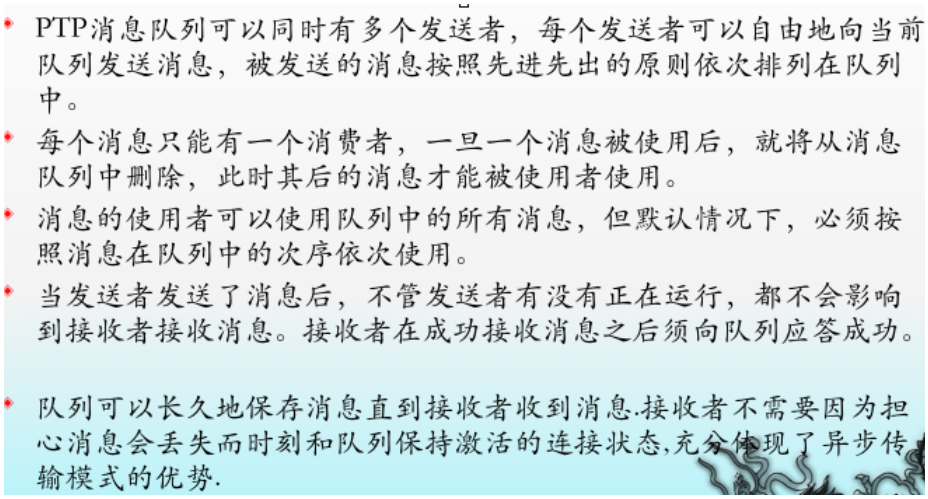
## 题目&答案

- 请画图描述 JSP 工作原理

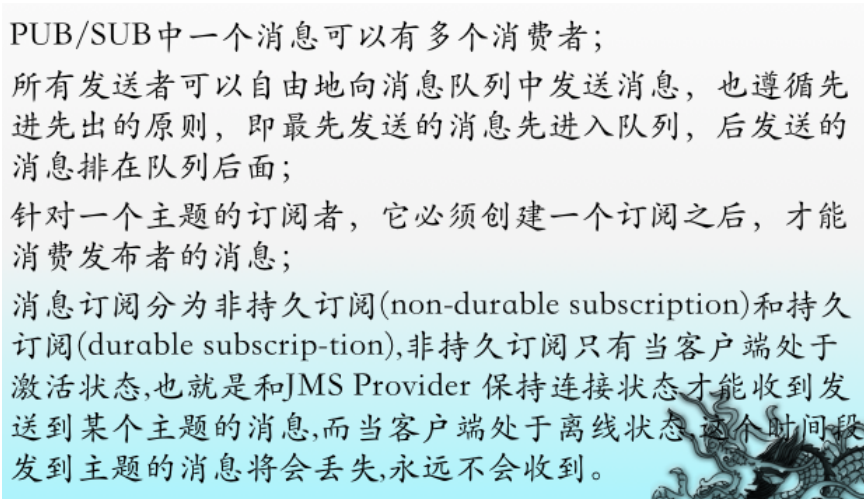



- 请简述 JNDI 的作用，并说明什么情况下可以使用本地访问，什么情况下需要使用远程访问。
  JNDI的作用：为开发人员提供了查找和访问各种命名和目录服务的通用、统一的接口。
  客户端与服务端的EJB对象在同一个JVM进程中使用本地访问。
  客户端与服务端的EJB对象不在同一个JVM进程中使用远程访问。

- 请简述 JMS 的两种消息传递模型
  点对点



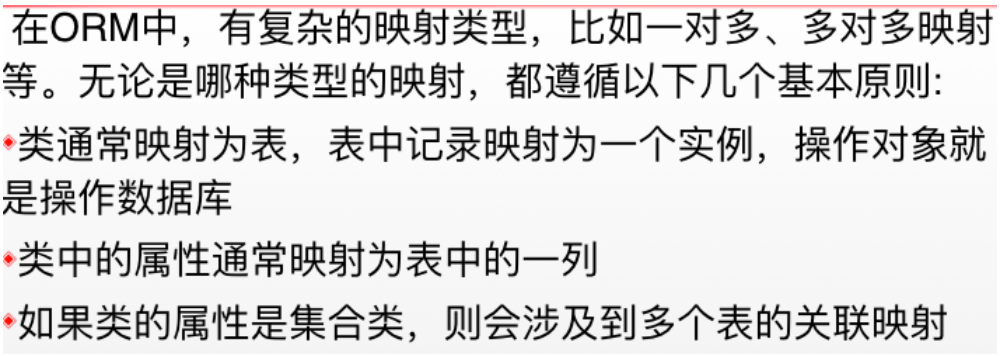

发布/订阅



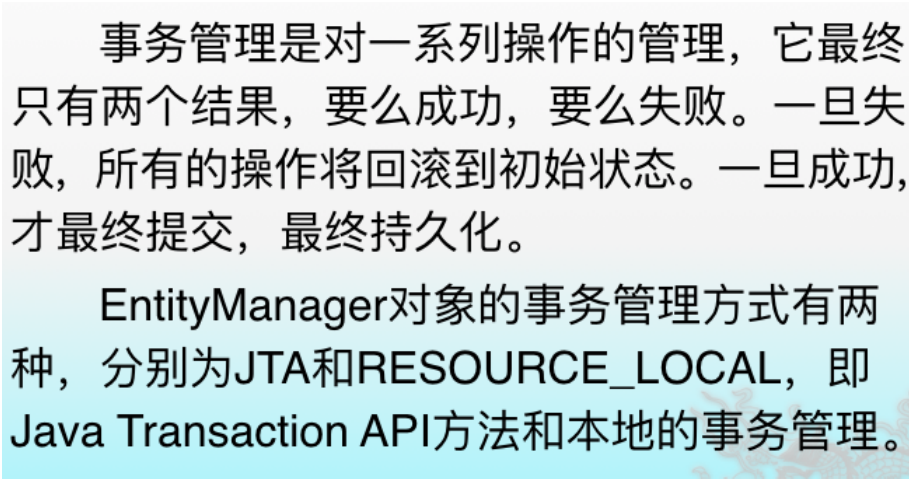

- 请简述什么是 ORM 及映射的基本原则

  ORM：对象关系映射（Object Relational Mapping）

  映射的基本原则：




- 请简述 JPA 中的事务管理

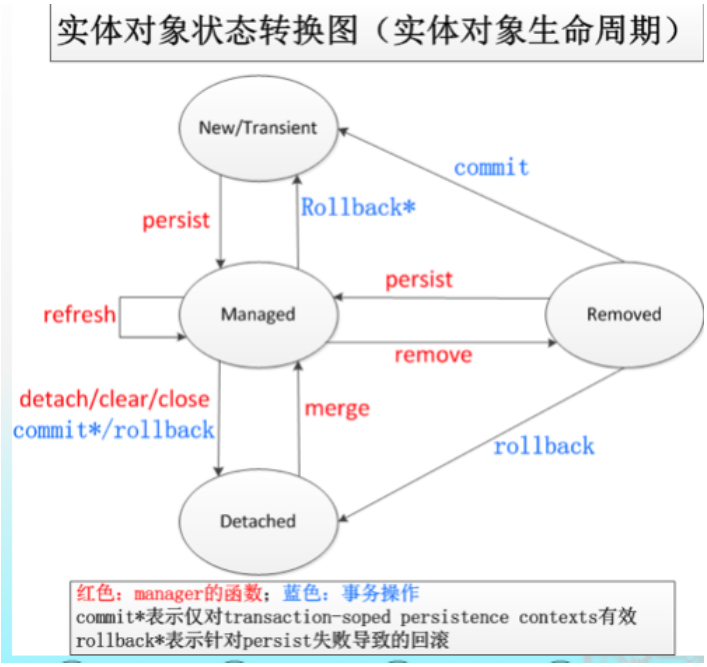



- 请画图描述有状态会话 Bean 生命周期

在ORM中，有复杂的映射类型，比如一对多、多对多映射等。无论是哪种类型的映射，都遵循以下几个基本原则:

- 类通常映射为表，表中记录映射为一个实例，操作对象就是操作数据库
- 类中的属性通常映射为表中的一列
- 如果类的属性是集合类，则会涉及到多个表的关联映射

- 请简要说明单例会话 Bean 中并发管理的两种方法及每种方法的特点
  单例会话 Bean 并发控制的两种方法
  - 容器管理并发CMC
    - EJB容器控制客户端访问单例会话Bean的业务方法。
    - 它是默认管理方式，相当于在所有业务方法之上的写锁定。
    - 只能放在方法之前，控制整个方法。
  - Bean管理并发BMC
    - 可以进行细粒度控制。
    - 利用Java编程语言的同步原语进行同步控制。
- 请画图描述 JPA 中实体的生命周期



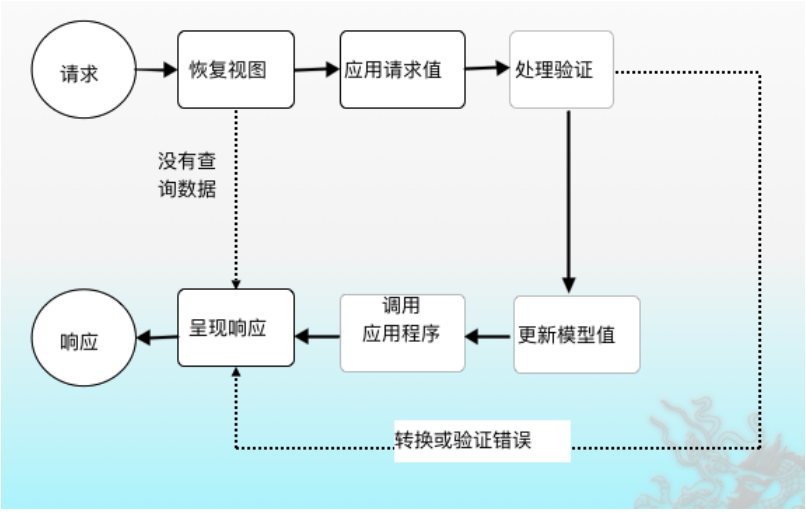

- JPA 实体之间的关联关系分为7种，请详细说明一对一单向和多对多双向关系中该如何使用标注和设计数据库
  - 一对一单向

- @JoinColumn标注用来指定实体关系的保存情况，@OneToOne标注只能确定实体之间的关系是一对一的关系，不能指定数据库表中用来表示关联关系的字段。所以此时要结合@JoinColumn标注来进行说明。
- @JoinColumn的name属性，指定外键字段的名字，在默认情况下，name=关联表的名称+“_”+ 关联表主键的字段名；
- @JoinColumn的referencedColumnName属性，指定了这个外键的引用字段。在默认情况下，为外键所指向的表的主键字段名称。

- 多对多双向

- 多对多双向可以**分解为两个多对多单向**，仍然需要使用关联表来维护其关系。
- 配置多对多双向关联的方法是，在关联关系的两端都增加@ManyToMany标注，并在关联关系拥有者一端**(主动端)配置@JoinTable**标注，而在另一端配置**(被动端)mappedBy属性**。

- 请画图描述 JSF 生命周期




- 简要叙述多层应用体系结构的构成和各个部分的功能。
  根据企业信息系统各个组成部分在功能上的区别，将应用系统划分为表示层、业务逻辑层和数据层。

  - 表示层
    也被称为客户层或客户端，用来提供呈现在客户端的人机交互界面，完成用户信息的输入工作和用户的需求结果的呈现工作。

- 业务逻辑层
  多层应用体系结构的核心部分，其主要工作包括：
  - 处理应用程序与具体业务内容相关的逻辑计算。
  - 数据库的访问和数据提取工作。
  - 用户所提交数据的解析、映射工作。
  - 用户所需要信息的界面组织构造工作。
  - 分布式系统的管理工作。
- 数据层
  提供数据的存储服务，一般就是数据库管理系统。

- 什么是 Servlet？简要叙述 Servlet 的生命周期。
  servlet：servlet 是一种 Java 程序类，是一种用来扩展以请求-应答模式运行应用程序的服务器的能力的一种 Java 程序。
  Servlet的生命周期
  - 装载初始化阶段
    如果这个 servlet 的实例不存在于容器中，容器将调用 servlet 类，生成这个 servlet 类的实例，并通过运行 init 方法初始化实例。
  - 执行时期
    借助 service 方法完成请求和应答的处理，通常情况下，Web 服务器会在比较长的时间里不停的处理请求和应答。
  - 结束时期
    如果有必要的话，容器将通过运行 servlet 类的 destroy 方法将 servlet 类的实例从容器中去除，从而能结束 servlet。
- 什么是有状态会话 Bean？简要叙述有状态会话 Bean 的生命周期。
  有状态会话 Bean：有状态会话 Bean 的实现类通常会包含类成员变量，当一个有状态会话 bean 实例存在时，其成员变量通常用来保存一个特定的客户端与有状态会话 Bean 实例会话的状态，这种状态通常被称为会话状态。
  有状态会话 Bean 的生命周期：在有状态会话 Bean 的生命周期中，分为不存在态、就绪态、钝化态等三个状态。容器执行依赖注入并调用标示了 @PostConstruct 元注释的方法。EJB 容器可以通过将 Bean 从内存移出到二级存储使 Bean 失效，容器立即调用标示了 @PrePassivate 元注释的方法。如果客户端调用了 Bean 的业务方法，EJB容器调用标示了 @PostActivate 元注释的方法。客户端调用标示了 @Remove 元注释的方法，EJB容器调用标示了 @PreDestroy 元注释的方法。

- 什么是消息驱动 Bean？简要叙述消息驱动 Bean 的生命周期。
  消息驱动 Bean：消息驱动 Bean 是允许 JavaEE 应用异步地处理消息的企业 Bean，这种类型的 Bean 通常担当一个消息监听器的作用，消息监听器类似于事件监听器，但是只接收消息。
  消息驱动 Bean 的生命周期：消息驱动 Bean 也从不被钝化，其生命周期也只有不存在态和就绪态。 EJB 容器通常生成一个消息驱动 Bean 实例池。容器执行依赖注入并调用标示了 @PostConstruct 元注释的方法。EJB 容器调用标示了 @PreDestroy 元注释的方法。

- 程序中，在某个类“Greeting”的对象实例“greeting”前面用“@Inject”修饰，其含义是什么？
  依赖注入：指能够以类型安全的方式将组件注入到应用程序的能力。

- 在两个有关联关系的实体之间，存在几种关联关系？存在几种关联方向？总共构成几种关系？
  存在四种关联关系：一对一、一对多、多对一、多对多。
  存在两种关联方向：单向关联、双向关联。
  总共构成七种关系，没有多对一双向。

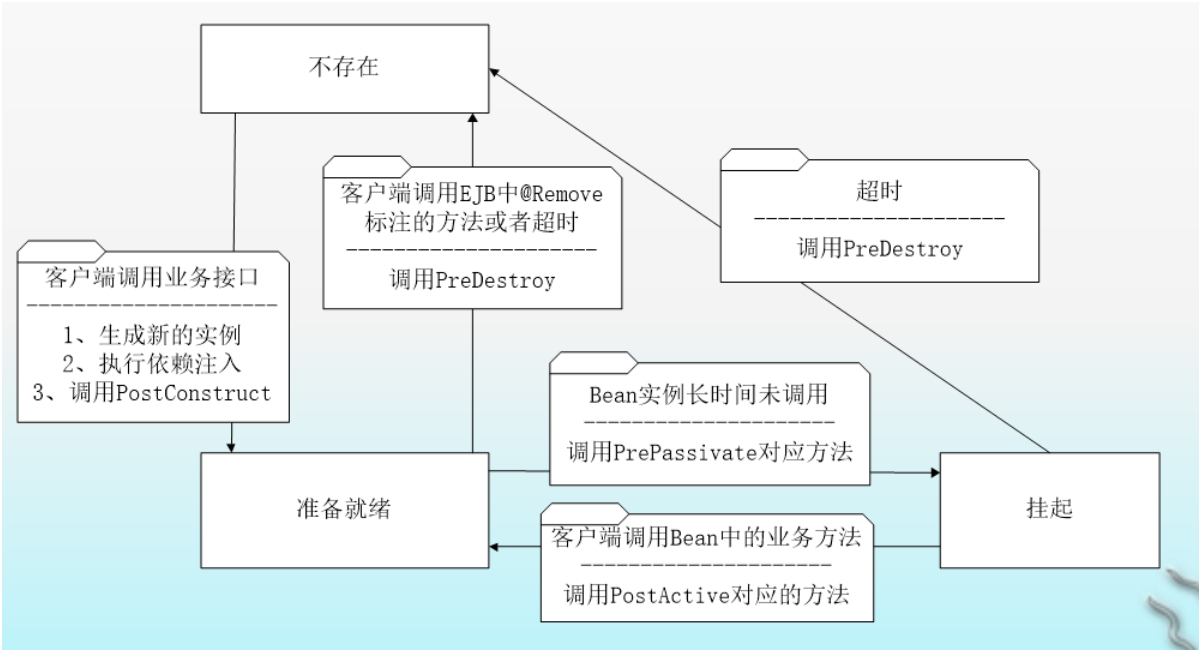
- 企业应用可以采用几种安全策略？

- 什么是消息？什么是消息服务？消息服务有几种类型？
  消息：是软件组件与应用系统之间的一个通讯方法，一个消息客户端可以发送消息到另一个客户端，也可从另一个客户端接收消息。
  消息服务：在分布式应用之间提供消息传递。
  消息服务的两种类型：点对点与发布/订阅。

## 参考

- 大纲知识点
- 课件
- 奔腾试题

有状态会话Bean的生命周期




JDBC驱动程序的使用建议
建议1：尽量不使用JDBC-ODBC桥驱动程序和数据库本地客户端驱动程序
建议2：使用本地Java驱动程序，对于独立的Java客户端程序是一个好的选择。不仅使得Java客户端不依赖于特定的环境，而且由于不需要经过中间环境的传递，可以获得更好的SQL语句处理速度。
建议3：对于在容器中运行的程序，建议使用中间件服务器提供的JDBC Java驱动程序，将数据库访问统一置于中间件服务器的管理之下，具有更好的安全性。对于客户端程序来说，也减少了由于数据库变化造成的代码修改。

无状态会话Bean和有状态会话Bean的区别：
无状态会话Bean不维持和客户端的会话状态，这意味着客户端程序对这类组件的两次方法调用之间是没有联系的。
有状态会话Bean是一种保持会话状态的服务，每个实例都与特定的客户端相关联，在与客户端的方法调用之间维持对话状态
（1）组件对象进入休眠（缓存状态）的时刻
无状态会话Bean在一个方法执行完毕后就会被释放到实例池中。
有状态会话Bean进入休眠状态，通常是因为对应的组件对象长时间不被使用，或服务器负载十分重的情况下才会发生；
（2）组件对象在休眠状态（缓存状态）的差别
无状态会话Bean进入缓存状态后，其对应的对象在服务器上还存在；
有状态会话Bean对象进入休眠状态后，其EJB组件对象被销毁，但其对象状态被保存到了硬盘或数据库中。
（3）组件对象进入休眠或（缓存状态）时，EJB服务器所保留的组件数据
无状态会话Bean组件对象进入缓存状态后，其被保留的是组件上下文环境；
有状态会话Bean组件对象进入休眠状态时被保存的不但是EJB组件的上下文环境，而且还包括EJB组件的各种属性状态。
（4）组件对象在代码形式上的区别
无状态会话Bean中一般没有属性。
有状态会话Bean 每个有状态会话bean应该至少定义一个使用@Remove注解标记的方法，客户端将使用这些方法来结束与有状态会话bean的会话。调用了一个这样的方法之后，服务器将销毁bean实例。