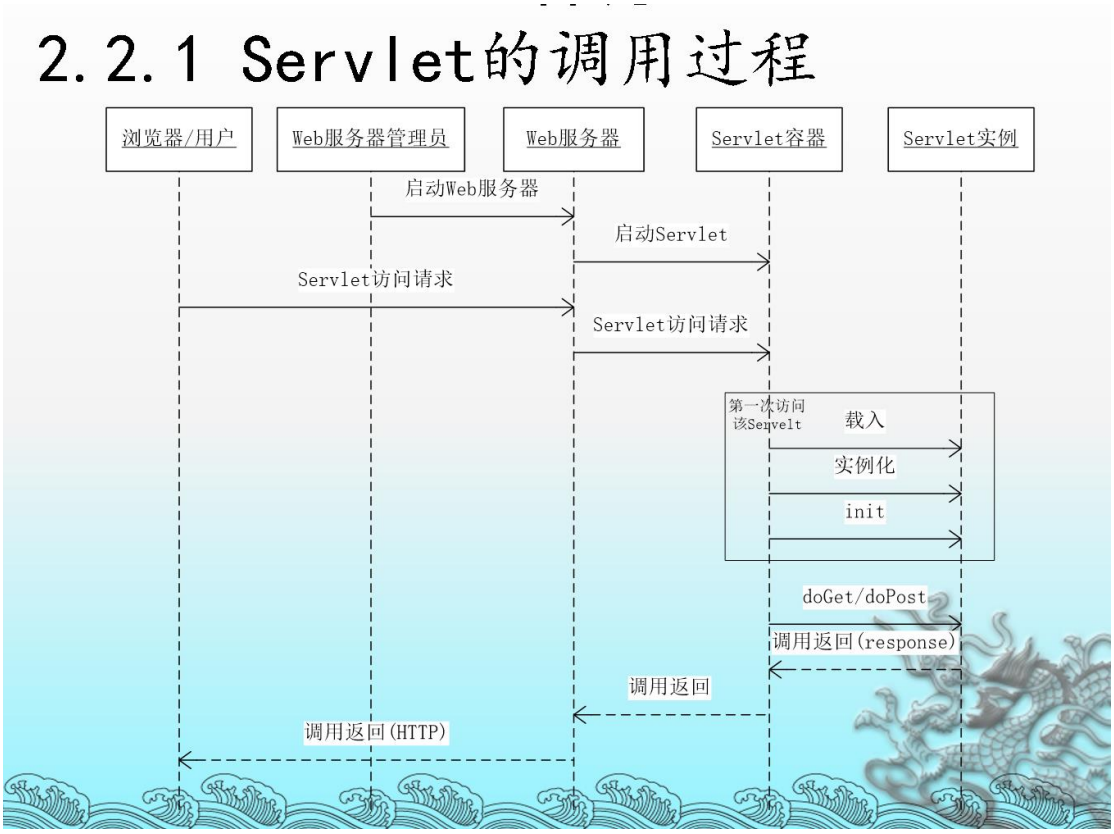
一．概论

1. javaEE 的 12 个规范的定义（干啥的，RMI，JTA，javamail 不是重点）以及缩写。
2. javaEE 架构的四大容器，以及一张图得看懂。

二．Servlet

1. 动态 web 而来，一个 servlet 的例子得会写。标注@WebServlet（）/web.xml 来配置；
2. Servlet 的调用步骤，一张图（注意第一次调用时）

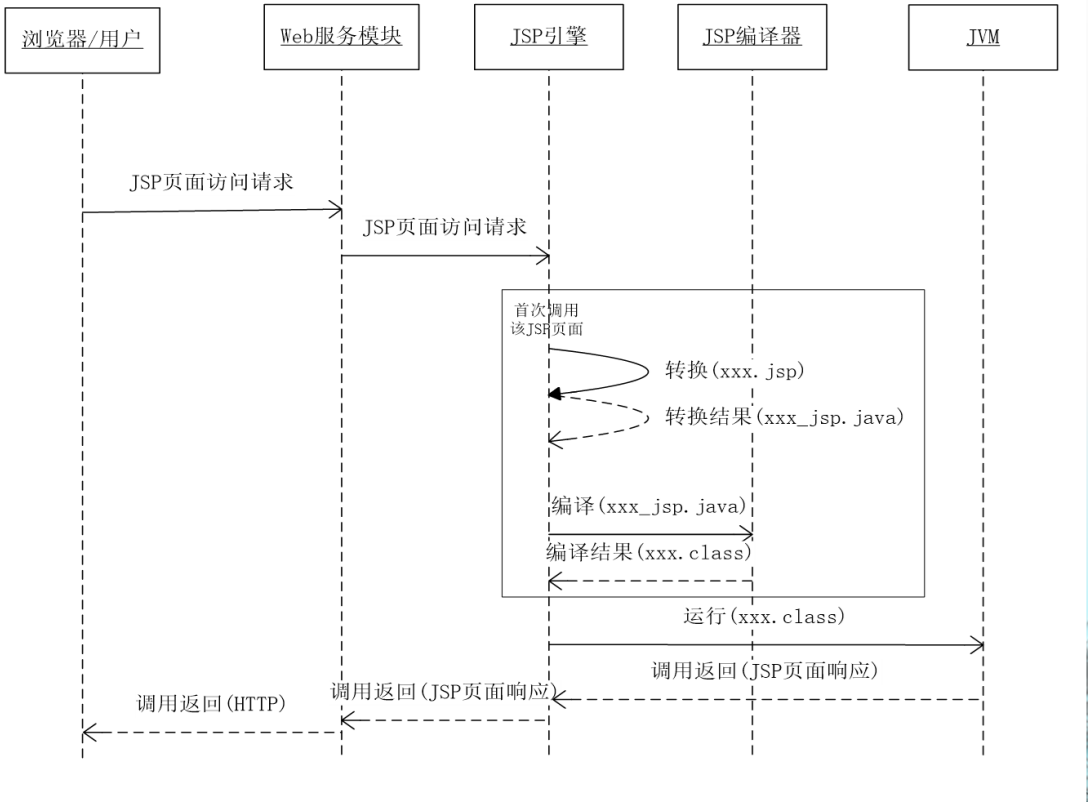


3. Servlet 生命周期：初始化，提供服务，销毁
4. Geneticservlet 和 httpservlet 接口的区别，一个更专一于 HTTP 协议。且都是抽象类
5. ServletConfig 和 ServletContext 接口，每一个 ServletConfig 对象对应着一个唯一的 Servlet。得到 servlet 自己的信息；Servlet 容器在调用 init 方法时，把 servletConfig 对象当做参数传递给 servlet 对象。Servlet 容器在启动一个 Web 应用时，会为它创建一个 servletContext 对象。每个 web 应用有唯一的 servletContext 对象。
6. Httpsession 接口，session 会话什么时候开始什么时候结束(打开浏览器到关闭浏览器)
7. Requestdispatcher 接口，用于请求和转发的，定义了两种方式，forward 和 include。
   servlet A
   RequestDispatcher disp = request.getRequestDispatcher("B");
   disp.forward(request, response);
8. 共享变量：（理解为多个页面之间的全局变量）：ServletContext 、HttpSession 和 HttpServletRequest，都是通过实例来调用 getAttribute（）和 setAttribute（）来实现的。ServletContext 范围最大，应用程序级别的，整个应用程序都能访问。HttpSession 次之，会话级别的，在当前的浏览器中都能访问（当前这个浏览器）。HttpServletRequest 范围最小，请求级别，请求结束，变量的作用域也结束。（当前 request）

（*page|request|session|application*）

三．JSP

1. jsp 的工作原理，一张图要背。



2. jsp 的构成：jsp 声明，程序块，表达式（直接转化为 out.print()），指令，动作（里面有很多很好用的动作，例如 include，forward 怎么写，注意没有求余符号），注释。
3. jsp 三个指令：page、include 和 taglib（自己定义标签并且使用）。<%@ 指令名 {属性名=*"属性值"* } %>
4. 内置对象：（要与 servlet 中的一些接口联系起来）
   1).输出输入对象:request、response、out（输出流对象，作用域是 page）
   2).通信控制对象:pageContext（它代表的是页面上下文，作用域是 page。使用该对象可以访问页面中的共享数据。）、session（会话）、application（是 javax.servlet.ServletContext 类型的一个实例，它表示的是该 JSP 页面所在应用的上下文环境信息，作用域是 application。最大的）
   3).Servlet 对象:page（作用域是 page。它与 Servlet 类中的 this 关键字相对应）、config（配置信息）
   4).错误处理对象:exception
5. javaBean 就是一个 java 程序，但是必须支持以下约束
   如果类的成员变量的名字是 xxx，那么为了更改或获取成员变量的值，在类中可以使用两个方法：getXxx()：用来获取属性 xxx。setXxx()：用来修改属性 xxx。（bool 类型就是 is）；类中方法的访问属性都必须是 public 的；类中如果有构造方法，那么这个构造方法也是 public 的并且没有参数。
6. jsp 调用 javaBean。可以使用三个动作：<jsp:useBean>，<jsp:setProperty>，

<jsp:getProperty>。

7. jsp 的正则表达式${表达式，即输出的内容}

四．JSF（md 都整理完了考试前几天突然说不考这章了…）

1. JSF 是一套基于组件的框架，是一个事件驱动型的组件模型。是基于 MVC 的框架。
2. 托管 Bean，（在真正的业务逻辑 Bean 及 UI 组件之间搭起桥梁），通过一个 EL 表达式，将托管 Bean 和一个 JSF 组件绑定在一起，从而建立关系，使用的例子<h:inputText value="#{userBean.name}"/>。
3. 如何定义（在普通 JavaBean 的基础之上需要进行配置）。配置需要引入标注或者 xml 标注的话需要引入@ManagedBean 和@SessionScoped（这个地方就是作用域，与上面呼应，JSF 有四个作用域应用程序作用域；会话作用域；请求作用域@RequestScoped；视图作用域@ViewScoped）。或者用上下文依赖标注的方法

UserBean. java

2、上下文依赖注入CDI方式

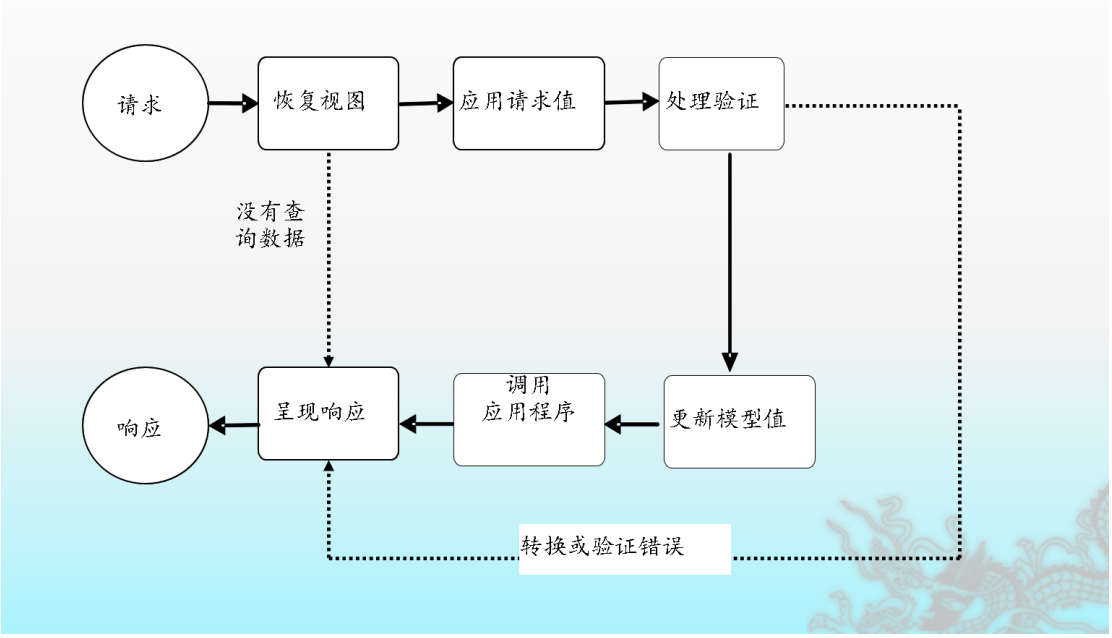
```
package javaee.jsf;
import javax.inject.Named;
import javax.enterprise.context.SessionScoped;
<!-- 定义Bean的作用域为会话 -->
@Named
@SessionScoped
public class UserBean
        implements Serializable {
   <!-- 定义属性 -->
   private String name ;
   private String password;
   <!-- 定义与属性对应的getter 和 setter方法
    ..................
}
```

注意：会话作用域内的CDI bean必须实现Serializable接口。

注意：使用这种方法时，必须在WEB-INF文件夹下包含一个名为beans.xml的文件，该文件可以为空，也可以包括一些bean的配置文件。

或者使用配置 xml 文件的方法在 WEB-INF/faces-config.xml 文件中。

4. 托管 bean 可以使用生命周期标注@PostConstruct 和@PreDestroy 标注，写在类里面。
5. JSF 正则表达式#{变量或者表达式}，并且可以用[]来代替.。和 JSP 中正则有点区别，${…}表示直接计算表达式，当处理页面时，计算并插入表达式的值。 #{…}分隔符表示延迟计算。JSF 实现保存表达式，并在任何需要的时候计算。
6. 导航机制，理解一下，挺形象的，动态和静态导航。
7. JSF 生命周期，很重要



应用请求值：用来自客户端的最新数据更新对应的服务器端组件的值(提交值)。
更新模型值：用组件的新数据更新与组件相关的服务器端模型的值。
调用应用程序：调用注册的动作监听器和动作方法，在动作方法中可以执行应用程序的业务逻辑。
呈现响应：把响应呈现给请求客户端。

8. JSF 的四个事件：动作事件（按钮，链接触发，调用应用程序阶段）；值更改事件（处理验证阶段）；阶段事件；系统事件
9. 转换和验证。标准转换器（基本类型 Date 除外，数字转换，日期转换）。东西很多，没有完全学会。

五．JDBC 和 JNDI 和 EJB

1. 会完整写 JDBC 代码
2. JNDI，提供一个统一的接口来执行名词和目录服务。
3. EJB 企业 Bean，服务端的一种组件规范。EJB 容器和 EJB 组件。

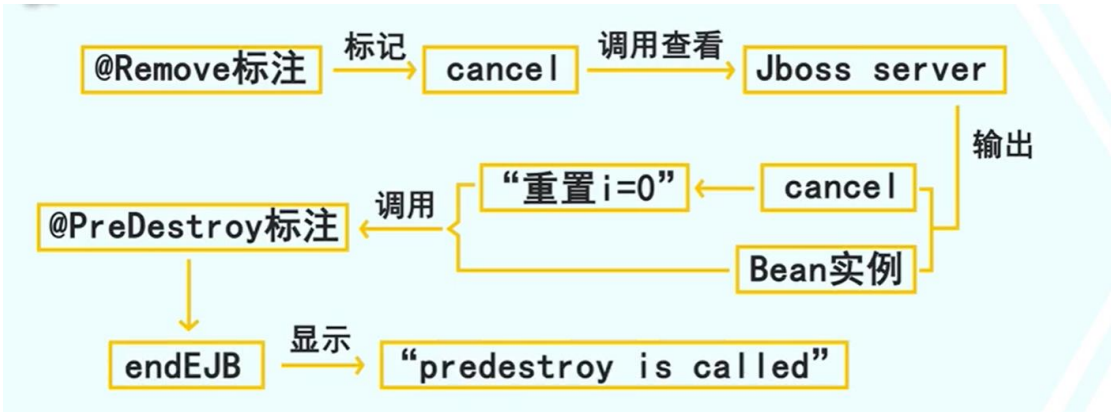
六．会话 Bean

1. 三个 JNDI 名词空间可以用来查找 EJB 对象，java：global；java：app；java：module。远程访问的话，必须用 java：jboss/exported。（在客户端调用那块）

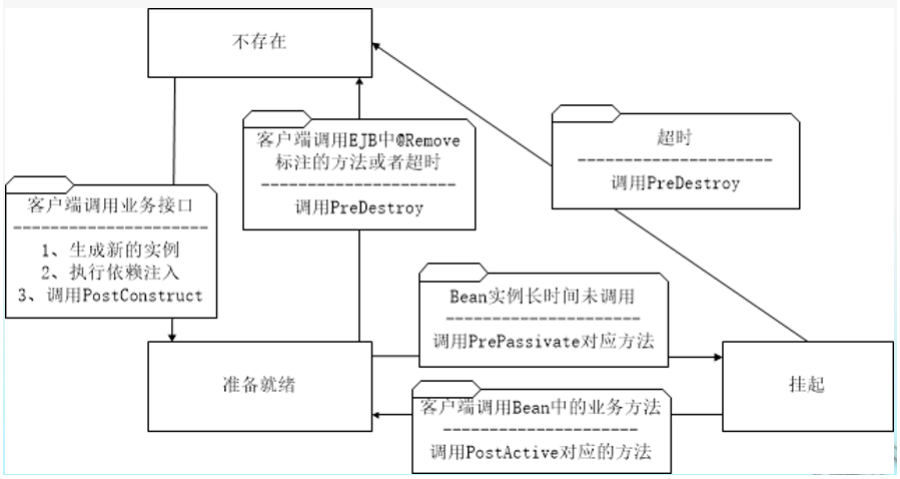
- 发布后在JBoss中给出的JNDI名字
- java:global/SessionEJB/HelloBean!jee.ejb.stateless.remote.HelloBeanRemote
- java:app/SessionEJB/HelloBean!jee.ejb.stateless.remote.HelloBeanRemote
- java:module/HelloBean!jee.ejb.stateless.remote.HelloBeanRemote
- java:jboss/exported/SessionEJB/HelloBean!jee.ejb.stateless.remote.HelloBeanRemote
- java:global/SessionEJB/HelloBean!jee.ejb.stateless.remote.HelloBean
- java:app/SessionEJB/HelloBean!jee.ejb.stateless.remote.HelloBean
- java:module/HelloBean!jee.ejb.stateless.remote.HelloBean

2. 有状态，无状态，单例的定义和区别。
3. 会话 Bean 的构成，就是在普通 javaBean 的基础之上加了点标注。接口（@Remote 和 @Local）和实现类（@Stateful 和@Stateless 和@singleton）。
4. 会话 Bean 客户端怎么写，怎么调用（一直不明白视频中为啥要客户端调用远程会话 Bean，在远程会话 Bean 里调用本地会话 Bean，这个客户端总不能考吧）

5. 无状态会话 Bean 中的无接口 Bean。没有接口了，只有实现类。但是实现类里也有两个标注一个是@Stateless 和@LocalBean
6. 无状态会话 Bean 的生命周期（也是单例会话 Bean 的生命周期），不存在状态，池状态，调用状态。针对的无状态会话 Bean 组件对象。服务完了就释放到缓冲池中，所有用户可能几次调用的时候都不是同一个对象。
7. 无状态会话 Bean 的生命事件 PostConstruct 和 PreDestroy（很眼熟，对了就是类似托管 Bean 中的）。
8. 配置无状态会话 Bean 生命事件。两种方式，一种是标注写在方法前即可，@PostConstruct 和@PreDestroy（注意方法参数列表必须为空，返回类型为 void，且方法不能抛出异常（要是我抛异常你能把我咋地）），另外一种是配置 ejb-jar.xml 文件。、
9. 有状态会话 Bean 和无状态会话 Bean 挺像，但是会多一个@Remove 标注



10. 有状态会话 Bean 的生命周期（理解一下他独特的生命周期是由于他的特性，一直得存在一个）



不存在状态，准备就绪状态，挂起状态。

四个生命周期事件，PostConstruct，PreDestroy，PostActive，PrePassivate。

11. 有状态和无状态会话 Bean 的区别（理解）。组件对象进入休眠（缓存状态）的时刻；组件对象在休眠状态或缓存状态的差别（有状态的 EJB 组件对象已经被销毁）；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（有状态会话 Bean 多保留了 EJB 属性状态）
12. 单例会话 Bean 并发访问，（由于单例会话 Bean 被所有客户端共享，所以必定产生并发问题）

    CMC 和 BMC（容器管理并发性和 Bean 管理并发性）。

    CMC，@Lock 标注，@Lock(LockType.WRITE)：这是一个排它锁，对其他客户锁定正在调用的方法，直至这个方法被调用完毕。@Lock(LockType.READ)：这是共享锁，允许多个客户并发访问或共享。

    BMC（可以更加细粒度的控制是对哪个部分来进行控制并发，不仅仅完全是方法）使用标注@ConcurrentcyManagement(ConcurrencyManagementType.BEAN)和 java 中自带的 **synchronized**
13. 多接口本质就是同时继承多个接口
14. 异步调用用一个标注@Asynchronous

七．JMS 和 MDB（消息驱动 Bean）

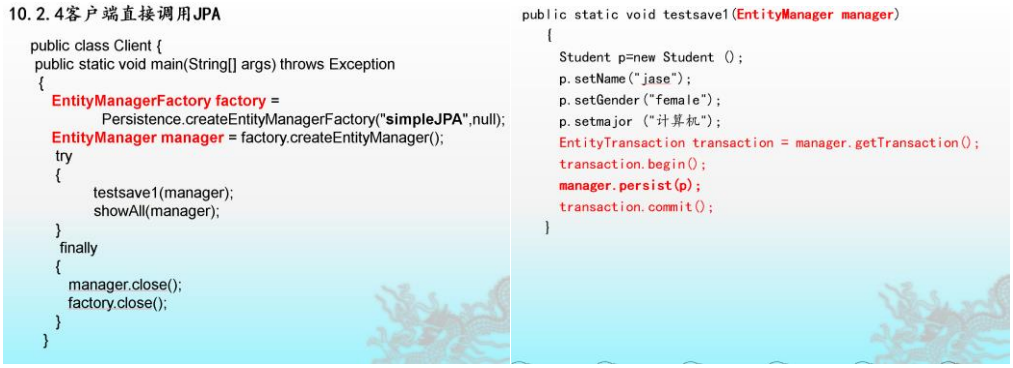
1. JMS 两种消息传递模型点对点（PTP）和发布/订阅（Pub/Sub）。注意是否有时间依赖性（同步接收和异步接收）。
2. JMS 程序的开发得看懂代码。同步是 MessageConsumer.receive(), 异步是 onMessage()。
3. MDB 是 JMS 异步消息的一种处理方式。也是能看懂代码即可
4. MDB 的生命周期类似于无状态会话 Bean。然后生命事件 PostConstruction 事件和 PreDestroy 事件

八．JPA（重点）

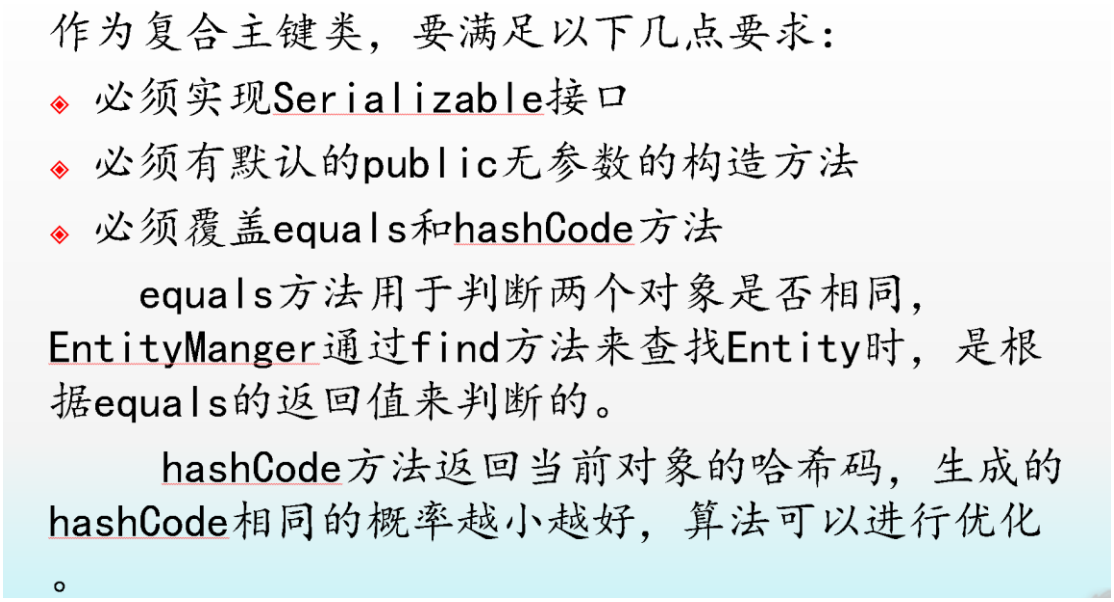
1. ORM 的定义（对象关系映射），类映射为表；属性映射表中属性；如果类的属性是集合类，则会涉及到多个表的关联映射。
2. （实体与表的映射）使用@Entity 标注对一个类进行标注，这个类就是实体类，（实体类

至少要有一个无参的构造方法；实体类至少要有一个主键。）类名默认和表名一致，不一致用@Table（name=）。映射方法和属性@Column（可以标注在类的方法和属性之前）来标志是数据库的哪个属性。@id 用来标注主键，常常与@GeneratedValue(strategy =GenerationType.IDENTITY)相结合使用，JPA 默认四种主键生成策略，auto，identity，sequence，table。

3. 客户端调用 JPA 的过程（也是通过啥工厂）

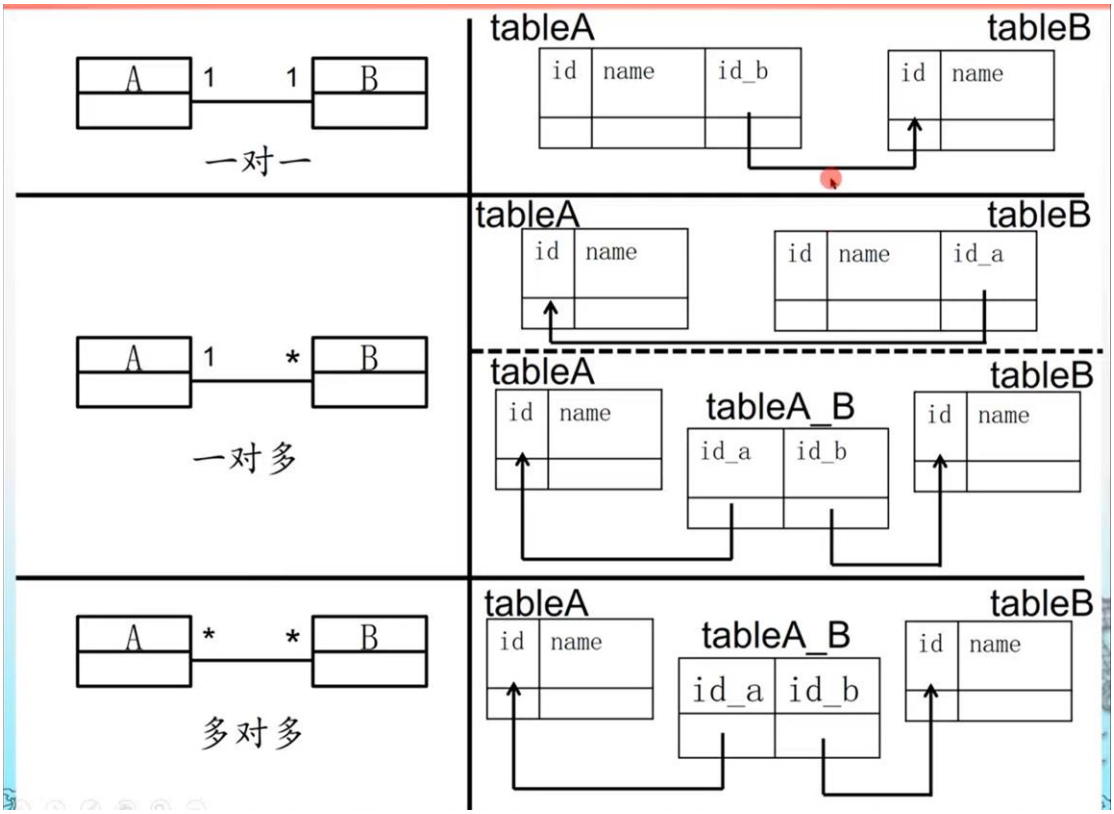


4. 复合主键。首先得另外写一个类，类还必须有一些要求

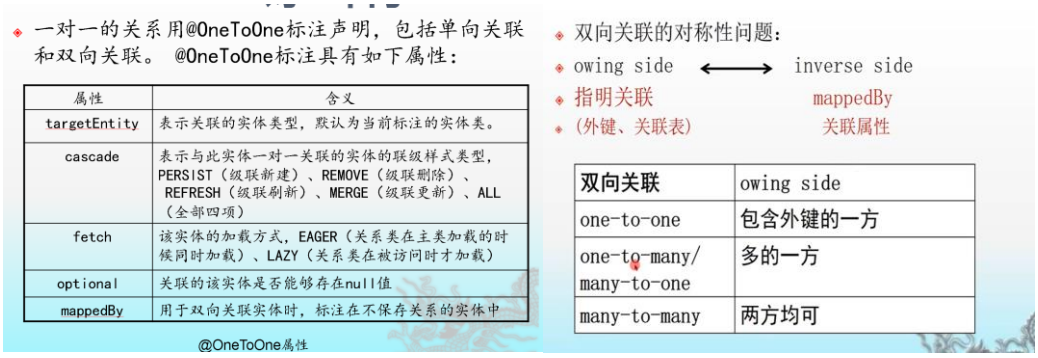


然后在实体类中通过@IdClass 标注在实体中标注复合主键，最后客户端生成对象时也得注意。

5. （实体与实体间的映射）单向关联，双向关联，一对一，一对多，多对一，多对多。（注意外键的位置，使用外键或者关联表的方法，）



6. 以@OnetoOne 为例等一些标注里面属性。其中 cascade 属性（级联）和 mappedBy（双向关联被动一方）属性非常重要。(注意右边这个主动和被动问题是针对双向关联的，单向关联主动方就是发出箭头的那一方)



一对一的关系用@OneToOne标注声明，包括单向关联和双向关联。 @OneToOne标注具有如下属性：

属性	含义
targetEntity	表示关联的实体类型，默认为当前标注的实体类。
cascade	表示与此实体一对一关联的实体的联级样式类型，PERSIST（级联新建）、REMOVE（级联删除）、REFRESH（级联刷新）、MERGE（级联更新）、ALL（全部四项）
fetch	该实体的加载方式，EAGER（关系类在主类加载的时候同时加载）、LAZY（关系类在被访问时才加载）
optional	关联的该实体是否能够存在null值
mappedBy	用于双向关联实体时，标注在不保存关系的实体中

@OneToOne属性

双向关联的对称性问题：

owing side ⟷ inverse side

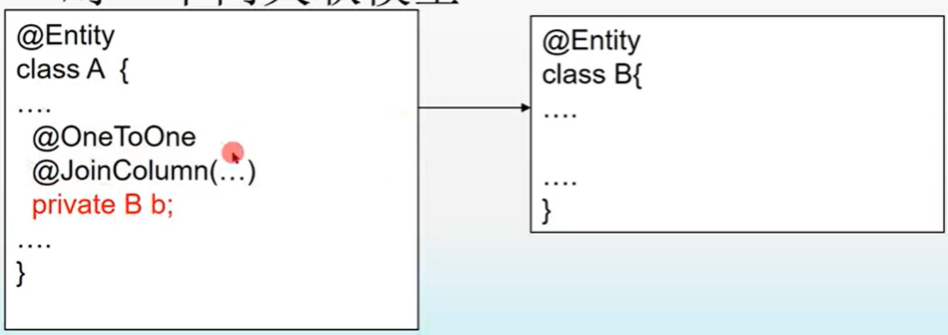
指明关联 mappedBy

（外键、关联表） 关联属性

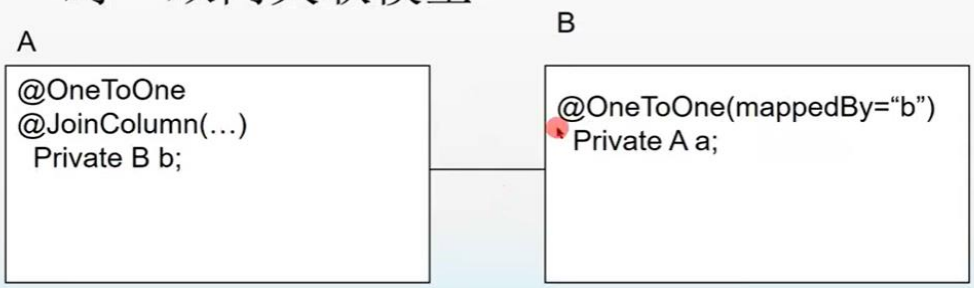
双向关联	owing side
one-to-one	包含外键的一方
one-to-many/ many-to-one	多的一方
many-to-many	两方均可

7. 所有映射关系对应的结构（important）
@JoinColumn 就是关联外键@JoinTable 就是关联表。@JoinColumn 中属性 name=关联表的名称+“_”+ 关联表主键的字段名；

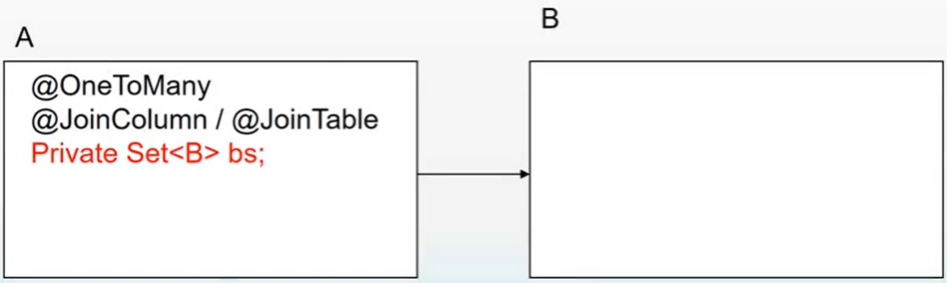
一对一单向关联模型



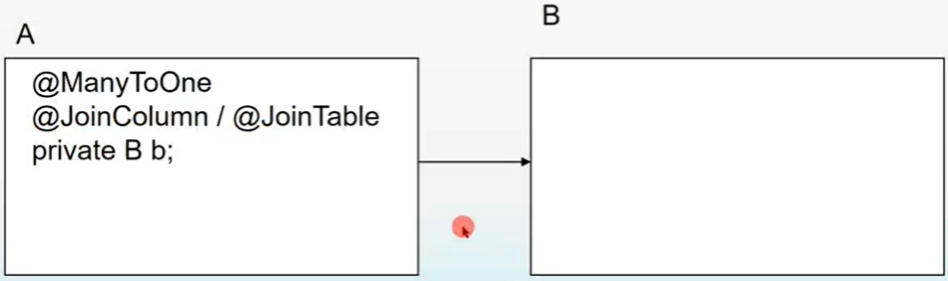
一对一双向关联模型

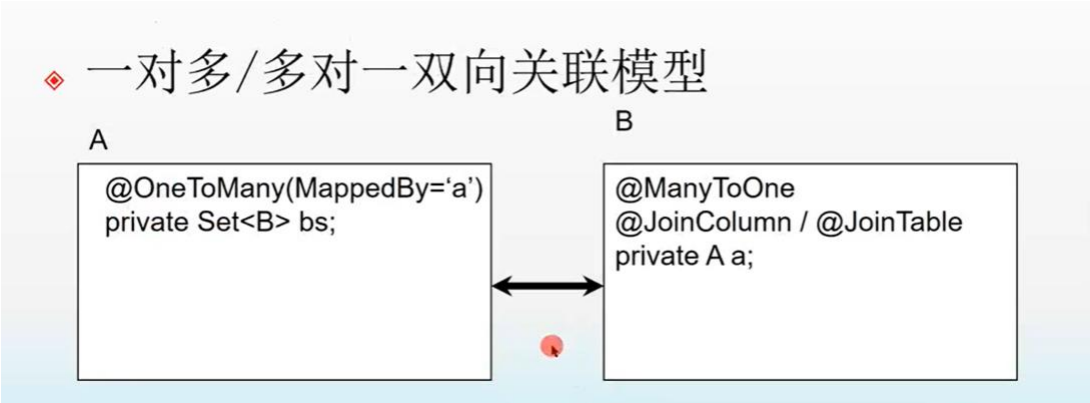


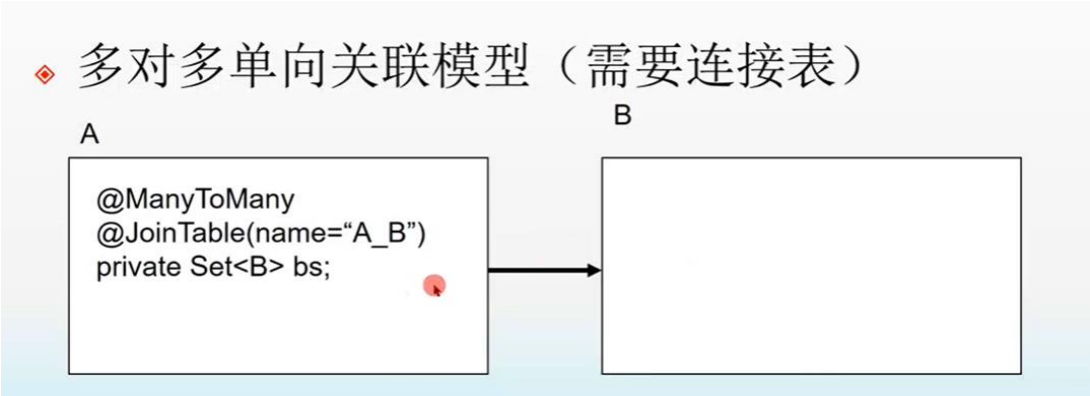
一对多(1:n)单向关联模型：

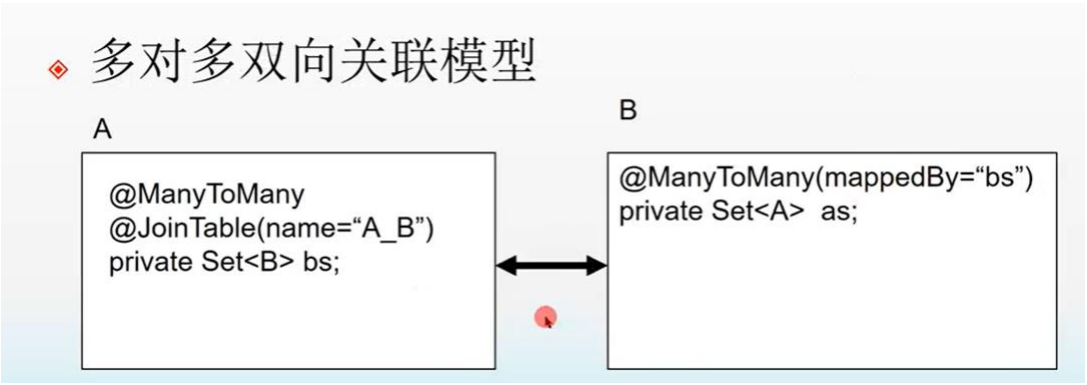


多对一单向关联模型

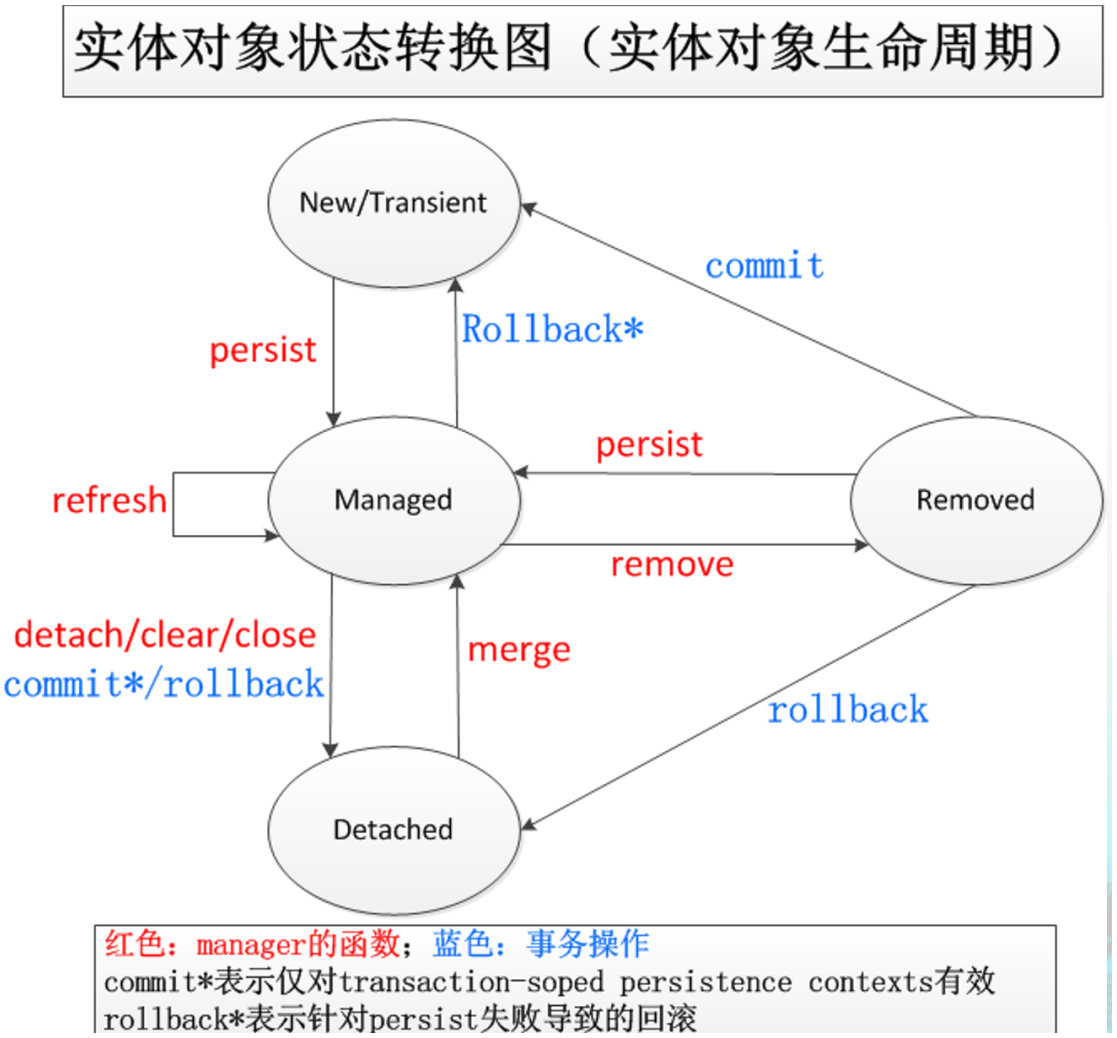








8. 实体管理器（具体实现就是在客户端中怎么调用 JPA，生成对象，调用方法）
   根据 EntityManager 对象的管理方式，可以有以下两种类型：
   容器托管的 EntityManager 对象和应用托管的 EntityManager 对象
   容器托管的 EntityManager 对象最简单，程序员不需要考虑 EntityManager 连接的释放，以及事务等复杂的问题，所有这些都交给容器去管理。容器托管的 EntityManager 对象必须在 EJB 容器中运行，而不能在 Web 容器和 JAVA SE 的环境中运行。应用托管的 EntityManager 反之。对象程序员需要手动地控制它的释放和连接、手动地控制事务等。且可与任何的 Java 环境集成。
9. 获取实体管理器对象的方式（这部分要是考写代码就很难了）
   容器托管：1、@PersistenceContext 标注注入 ； 2、JNDI 方式。
   应用托管：应用托管的 EntityManager 对象，都是通过实体管理器工厂(EntityManagerFactory)对象创建的。（根据不同的应用有不同的方法）
10. 实体的生命周期（图得背熟）
    四个状态：瞬时状态，托管状态，游离状态，以及销毁状态。



11. JPQL 是基于实体的语言。面向对象的。（熟悉 JPQL 查询语句怎么写就行）
    Select 语句返回的类型可以是实体/实体的属性/计算结果/新对象。
    怎么用 JPQL（跟 EJB 实体管理器怎么用，搭上关系了），查询参数得会

执行JPQL查询的步骤

1. 使用注入或通过 EntityManagerFactory 实例获取一个 EntityManager 实例。
2. 通过调用相应 EntityManager 的 createQuery 方法，创建一个 Query 实例。
3. 如果有查询参数，使用相应 Query 的 setParameter 方法进行设置。

4. 使用相应 Query 的方法 getSingleResult 或 getResultList 执行查询。如果进行更新或删除操作，必须使用 executeUpdate 方法，它返回已更新或删除的实体实例的数量。

查询参数
- 位置参数：通过问号（？）加位置来定义
- 命名参数：通过冒号（：）加参数名来定义。

九．JAVAEE 事务

1. Java 事务
   Bean 管理的事务（JDBC 事务和 JTA 事务）和（容器管理的事务更方便）
2. EJB 事务
   容器管理的事务（CMT）
   Bean 管理的事务（BMT），分为 JDBC 和 JTA（更灵活）
3. JPA 管理的事务（和实体管理器联系起来，注意区分和上面不太一样）

14.3 JPA的事务管理

不同的运行环境、不同的 EntityManager对象所支持的事务类型,如下表所示:

	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

十．Webservice（本质是一个超大的 servlet）

1. 四大技术（XML（封装，把功能变成方法），SOAP（访问，一种协议），WSDL（描述，一个 xml 文档），UDDI（目录服务））
2. Webservice 中的角色（服务提供者，服务请求者，服务代理）。
3. Bigwebservice（soap 和 wsdl，面向活动的）和 restestfulwebservice（面向资源的服务，url，核心操作 get，put，post，delate）