 在 struts hibernate 这种结构中，是不应该把Hibernate产生的PO直接传递给JSP的，不管他是Iterator，还是List，这是一个设计错误。我来谈谈在J2EE架构中各层的数据表示方法：Web层的数据表示是FormBean，数据来源于HTML Form POST 业务层的数据表示是VO持久层的数据表示是PO，其数据来源于数据库，持久层的数据表示例如CMP在一个规范的J2EE架构中，不同层的数据表示应该被限制在层内，而不应该扩散到其它层，这样可以降低层间的耦合性，提高J2EE架构整体的可维护性和可扩展性。比如说Web层的逻辑进行了修改，那么只需要修改FormBean的结构，而不需要触动业务层和持久层的代码修改。同样滴，当数据库表进行了小的调整，那么也只需要修改持久层数据表示，而不需要触动业务层代码和Web层代码。不过由于Hibernate的强大功能，例如动态生成PO，PO的状态管理可以脱离Session，使得在应用了Hibernate的J2EE框架中，PO完全可以充当VO，因此我们下面把PO和VO合并，统称为PO。先来谈谈ActionFormBean和持久层的PO之间的重大区别:在简单的应用中，ActionFormBean和PO几乎是没有区别，所以很多人干脆就是用ActionFormBean来充当PO，于是ActionFormBean从JSP页面到Servlet控制层再到业务层，然后穿过持久层，最后一直映射到数据库表。真是一竿子捅到了底！但是在复杂的应用中，ActionFormBean和PO是分离的，他们也不可能一样。ActionFormBean是和网页里面的Form表单一一对应的

，Form里面有什么元素，Bean里面就有什么属性。而PO和数据库表对应，因此如果数据库表不修改，那么PO也不会修改，如果页面的流程和数据库表字段对应关系不一致，那么你又如何能够使用ActionFormBean来取代PO呢？比如说吧，用户注册页面要求注册用户的基本信息，因此HTML Form里面包含了基本信息属性，于是你需要一个ActionFormBean来一一对应(注意：是一一对应)，每个Bean属性对应一个文本框或者选择框什么的。而用户这个持久对象呢？他的属性和ActionFormBean有什么明显不同呢？他会有一些ActionFormBean所没有的集合属性，比如说用户的权限属性，用户的组属性，用户的帖子等等。另外还有可能的是在ActionFormBean里面有3个属性，分别是用户的First Name, Middle Name, Last Name，而在我的User这个持久对象中就是一个 Name 对象属性。假设我的注册页面原来只要你提供First Name，那么ActionFormBean就这一个属性，后来我要你提供全名，你要改ActionFormBean，加两个属性。但是这个时候PO是不应该修改滴，因为数据库没有改。那么在一个完整的J2EE系统中应该如何进行合理的设计呢？JSP(View) - Action Form Bean (Module) - Action(Control)Action Form Bean是Web层的数据表示，它和HTML页面Form对应，只要Web页面的操作流程发生改变，它就要相应的进行修改，它不应该也不能被传递到业务层和持久层，否则一旦页面修改，会一直牵连到业务层和持久层的大面积的代码进行修改，对于软件的可维护性和可扩展性而言，是一个灾难，Actiont就是他的边界，到此为止！Action(Web Control) - Business Bean - DAO - ORM -DB而PO则是业务层和持久层的数据表示，它在业务层

和持久层之间进行流动，他不应该也不能被传递到Web层的View中去，而ActionServlet就是他的边界，到此为止！然后来看一看整个架构的流程：当用户通过浏览器访问网页，提交了一个页面。于是Action拿到了这个FormBean，他会把FormBean属性读出来，然后构造一个PO对象，再调用业务层的Bean类，完成了注册操作，重定向到成功页面。而业务层Bean收到这个PO对象之后，调用DAO接口方法，进行持久对象的持久化操作。当用户查询某个会员的信息的时候，他用全名进行查询，于是Action得到一个UserNameFormBean包括了3个属性，分别是first name, middle name, last name，然后Action把UserNameFormBean的3个属性读出来，构造Name对象，再调用业务Bean，把Name对象传递给业务Bean，进行查询。业务Bean取得Name(注意: Name对象只是User的一个属性)对象之后调用DAO接口，返回一个User的PO对象，注意这个User不同于在Web层使用的UserFormBean，他有很多集合属性滴。然后业务Bean把User对象返回给Action。Action拿到User之后，把User的基本属性取出(集合属性如果不需要就免了)，构造UserFormBean，然后把UserFormBean request.setAttribute(...)，然后重定向到查询结果页面。查询页面拿到request对象里面的ActionFormBean，自动调用tag显示之。总结：Form Bean 是Web层的数据表示，他不能被传递到业务层；PO是持久层的数据表示，在特定情况下，例如Hibernate中，他可以取代VO出现在业务层，但是不管PO还是VO都必须限制在业务层内使用，最多到达Web层的Control，绝不能被扩散到View去。Form Bean 和PO之间的数据转化是在Action中进行滴。BTW:JDO1.x还不能

像Hibernate功能这样强大，PO不能脱离持久层，所以必须在业务层使用VO，因此必须在业务层进行大量的VO和PO的转化操作，相对于Hibernate来说，编程比较烦琐。