<https://blog.csdn.net/gaoyunpeng/article/details/2093211>

**PO(persistant object) 持久对象**  
在o/r 映射的时候出现的概念,如果没有o/r映射,就没有这个概念存在了.通常对应数据模型(数据库),本身还有部分业务逻辑的处理.可以看成是与数据库中的表相映射的java对象.最简单的PO就是对应数据库中某个表中的一条记录,多个记录可以用PO的集合.PO中应该不包含任何对数据库的操作.  
  
**VO(value object) 值对象**  
通常用于业务层之间的数据传递,和PO一样也是仅仅包含数据而已.但应是抽象出的业务对象,可以和表对应,也可以不,这根据业务的需要.个人觉得同DTO(数据传输对象),在web上传递.  
  
**BO(business object) 业务对象**  
从业务模型的角度看,见UML元件领域模型中的领域对象.封装业务逻辑的java对象,通过调用DAO方法,结合PO,VO进行业务操作.  
  
**POJO(plain ordinary java object) 简单无规则java对象**  
纯 的传统意义的java对象.就是说在一些Object/Relation Mapping工具中,能够做到维护数据库表记录的persisent object完全是一个符合Java Bean规范的纯Java对象,没有增加别的属性和方法.我的理解就是最基本的Java Bean,只有属性字段及setter和getter方法!.  
  
**DAO(data access object) 数据访问对象**  
是sun的一个标准j2ee设计模式,这个模式中有个接口就是DAO,它负责持久层的操作.为业务层提供接口.此对象用于访问数据库.通常和PO结合使用,DAO中包含了各种数据库的操作方法.通过它的方法,结合PO对数据库进行相关的操作.夹在业务逻辑与数据库资源中间.配合VO, 提供数据库的CRUD操作...  
  
**DTO (Data Transfer Object)数据传输对象**  
主要用于远程调用等需要大量传输对象的地方。  
比如我们一张表有100个字段，那么对应的PO就有100个属性。  
但是我们界面上只要显示10个字段，  
客户端用WEB service来获取数据，没有必要把整个PO对象传递到客户端，  
这时我们就可以用只有这10个属性的DTO来传递结果到客户端，这样也不会暴露服务端表结构.到达客户端以后，如果用这个对象来对应界面显示，那此时它的身份就转为VO  
  
**O/R Mapper 对象/关系 映射**  
定义好所有的mapping之后,这个O/R Mapper可以帮我们做很多的工作.通过这些mappings,这个O/R Mapper可以生成所有的关于对象保存,删除,读取的SQL语句,我们不再需要写那么多行的DAL代码了.  
实体Model(实体模式)  
DAL(数据访问层)  
IDAL(接口层)  
DALFactory(类工厂)  
BLL(业务逻辑层)  
BOF    Business Object Framework      业务对象框架  
SOA    Service Orient Architecture    面向服务的设计  
EMF    Eclipse Model Framework        Eclipse建模框架