O/R Mapping 是 Object Relational Mapping（对象关系映射）的缩写。通俗点讲，就是将对象与关系数据库绑定，用对象来表示关系数据。在O/R Mapping的世界里，有两个基本的也是重要的东东需要了解，即VO，PO。

VO，值对象(Value Object)，PO，持久对象(Persisent Object)，它们是由一组属性和属性的get和set方法组成。从结构上看，它们并没有什么不同的地方。但从其意义和本质上来看是完全不同的。

1．VO是用new关键字创建，由GC回收的。

PO则是向数据库中添加新数据时创建，删除数据库中数据时削除的。并且它只能存活在一个数据库连接中，断开连接即被销毁。

2．VO是值对象，精确点讲它是业务对象，是存活在业务层的，是业务逻辑使用的，它存活的目的就是为数据提供一个生存的地方。

PO则是有状态的，每个属性代表其当前的状态。它是物理数据的对象表示。使用它，可以使我们的程序与物理数据解耦，并且可以简化对象数据与物理数据之间的转换。

3．VO的属性是根据当前业务的不同而不同的，也就是说，它的每一个属性都一一对应当前业务逻辑所需要的数据的名称。

PO的属性是跟数据库表的字段一一对应的。

PO对象需要实现序列化接口。

-------------------------------------------------

PO是持久化对象，它只是将物理数据实体的一种对象表示，为什么需要它？因为它可以简化我们对于物理实体的了解和耦合，简单地讲，可以简化对象的数据转换为物理数据的编程。VO是什么？它是值对象，准确地讲，它是业务对象，是生活在业务层的，是业务逻辑需要了解，需要使用的，再简单地讲，它是概念模型转换得到的。

首先说PO和VO吧，它们的关系应该是相互独立的，一个VO可以只是PO的部分，也可以是多个PO构成，同样也可以等同于一个PO（当然我是指他们的属性）。正因为这样，PO独立出来，数据持久层也就独立出来了，它不会受到任何业务的干涉。又正因为这样，业务逻辑层也独立开来，它不会受到数据持久层的影响，业务层关心的只是业务逻辑的处理，至于怎么存怎么读交给别人吧！不过，另外一点，如果我们没有使用数据持久层，或者说没有使用hibernate，那么PO和VO也可以是同一个东西，虽然这并不好。

----------------------------------------------------

java的(PO,VO,TO,BO,DAO,POJO)解释

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，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。

TO(Transfer Object)，数据传输对象

在应用程序不同tie(关系)之间传输的对象

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

是一个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...

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

----------------------------------------

PO：全称是

persistant object持久对象

最形象的理解就是一个PO就是数据库中的一条记录。

好处是可以把一条记录作为一个对象处理，可以方便的转为其它对象。

BO：全称是

business object:业务对象

主要作用是把业务逻辑封装为一个对象。这个对象可以包括一个或多个其它的对象。

比如一个简历，有教育经历、工作经历、社会关系等等。

我们可以把教育经历对应一个PO，工作经历对应一个PO，社会关系对应一个PO。

建立一个对应简历的BO对象处理简历，每个BO包含这些PO。

这样处理业务逻辑时，我们就可以针对BO去处理。

VO ：

value object值对象

ViewObject表现层对象

主要对应界面显示的数据对象。对于一个WEB页面，或者SWT、SWING的一个界面，用一个VO对象对应整个界面的值。

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

POJO ：

plain ordinary java object 简单java对象

个人感觉POJO是最常见最多变的对象，是一个中间对象，也是我们最常打交道的对象。

一个POJO持久化以后就是PO

直接用它传递、传递过程中就是DTO

直接用来对应表示层就是VO

DAO：

data access object数据访问对象

这个大家最熟悉，和上面几个O区别最大，基本没有互相转化的可能性和必要.

主要用来封装对数据库的访问。通过它可以把POJO持久化为PO，用PO组装出来VO、DTO

-----------------------------------------------------------------

PO:persistant object持久对象,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.

DAO:data access object数据访问对象，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.

BO:business object业务对象,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,我个人觉得它和其他不是一个层面上的东西,VO和PO应该都属于它.

---------------------------------------------

VO：值对象、视图对象

PO：持久对象

QO：查询对象

DAO：数据访问对象

DTO：数据传输对象

----------------------------------------

struts 里的 ActionForm 就是个VO;

hibernate里的 实体bean就是个PO,也叫POJO;

hibernate里的Criteria 就相当于一个QO;

在使用hibernate的时候我们会定义一些查询的方法,这些方法写在接口里,可以有不同的实现类.而这个接口就可以说是个DAO.

个人认为QO和DTO差不多.

----------------------------------------

PO或叫BO，与数据库最接近的一层，是ORM中的O，基本上是数据库字段对应BO中的一个属性，为了同步与安全性考虑，最好只给DAO或者Service调用，而不要用packcode,backingBean,或者BO调。

DAO，数据访问层，把VO，backingBean中的对象可以放入。。。。

DTO，很少用，基本放入到DAO中，只是起到过渡的作用。

QO，是把一些与持久性查询操作与语句放入。。

VO，V层中用到的基本元素与方法等放其中。如果要其调用BO，则要做BO转换VO，VO转换BO操作。VO的好处是其页面的元素属性多于BO，可起到很好的作用。。。。

-----------------------------------------

楼上的不对吧，PO是持久化对象。BO＝business object—业务对象。

PO可以严格对应数据库表，一张表对映一个PO。

BO则是业务逻辑处理对象，我的理解是它装满了业务逻辑的处理，在业务逻辑复杂的应用中有用。

VO：value object值对象、view object视图对象

PO：持久对象

QO：查询对象

DAO：数据访问对象——同时还有DAO模式

DTO：数据传输对象——同时还有DTO模式

参考资料：http://blog.csdn.net/fableking/archive/2008/02/11/2088514.aspx