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...

pojo:不按mvc分层,只是java bean有一些属性,还有get set方法

domain:不按mvc分层,只是java bean有一些属性,还有get set方法

po:用在持久层,还可以再增加或者修改的时候,从页面直接传入action中,它里面的java bean 类名等于表名,

属性名等于表的字段名,还有对应的get set方法

vo: value object表现层对象,主要用于在高级查询中从页面接收传过来的各种参数.好处是扩展性强

bo: 用在servie层,现在企业基本不用.

这些po,vo, bo,pojo可以用在各种层面吗

可以,也就是po用在表现层,vo用在持久层不报错,因为都是普通的java bean没有语法错误.

但是在企业最好不要混着用,因为这些都是设计的原则,混着用比较乱.不利于代码维护.

自学方法论: 理论 -> 实践 -> 理论 -> 实践 反复迭代三遍

总结:

1. mybatis是一个持久层框架, 作用是跟数据库交互完成增删改查

2.原生Dao实现(需要接口和实现类)

4.动态代理方式(只需要接口)

mapper接口代理实现编写规则:

1) 映射文件中namespace要等于接口的全路径名称

2) 映射文件中sql语句id要等于接口的方法名称

3) 映射文件中传入参数类型要等于接口方法的传入参数类型

4) 映射文件中返回结果集类型要等于接口方法的返回值类型

5. #{}占位符:占位

如果传入的是基本类型,那么#{}中的变量名称可以随意写

如果传入的参数是pojo类型,那么#{}中的变量名称必须是pojo中的属性.属性.属性...

6. ${}拼接符:字符串原样拼接

如果传入的是基本类型,那么${}中的变量名必须是value

如果传入的参数是pojo类型,那么${}中的变量名称必须是pojo中的属性.属性.属性...

注意:使用拼接符有可能造成sql注入,在页面输入的时候可以加入校验,不可输入sql关键字,不可输入空格

7. 映射文件:

1)传入参数类型通过parameterType属性指定

2)返回结果集类型通过resultType属性指定

8. hibernate和mybatis区别:

hibernate:它是一个标准的orm框架,比较重量级,学习成本高.

优点:高度封装,使用起来不用写sql,开发的时候,会减低开发周期.

缺点:sql语句无法优化

应用场景:oa(办公自动化系统), erp(企业的流程系统)等,还有一些政府项目,

总的来说,在用于量不大,并发量小的时候使用.

mybatis:它不是一个orm框架, 它是对jdbc的轻量级封装, 学习成本低,比较简单

有点:学习成本低, sql语句可以优化, 执行效率高,速度快

缺点:编码量较大,会拖慢开发周期

应用场景: 互联网项目,比如电商,P2p等

总的来说是用户量较大,并发高的项目.