实体类（VO，DO，DTO）的划分

经常会接触到VO，DO，DTO的概念，本文从领域建模中的实体划分和项目中的实际应用情况两个角度，对这几个概念进行简析。

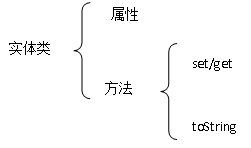
得出的主要结论是：在项目应用中，VO对应于页面上需要显示的数据（表单），DO对应于数据库中存储的数据（数据表），DTO对应于除二者之外需要进行传递的数据。

一、实体类

百度百科中对于实体类的定义如下：

实体类的主要职责是存储和管理系统内部的信息，它也可以有行为，甚至很复杂的行为，但这些行为必须与它所代表的实体对象密切相关。

根据以上定义，我们可以了解到，实体类有两方面内容，存储数据和执行数据本身相关的操作。这两方面内容对应到实现上，最简单的实体类是POJO类，含有属性及属性对应的set和get方法，实体类常见的方法还有用于输出自身数据的toString方法。

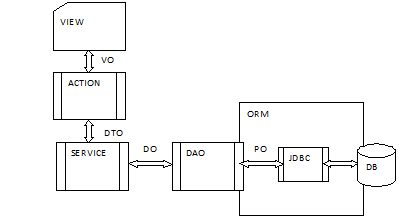
[](http://photo.blog.sina.com.cn/showpic.html#blogid=7a4cdec80100wkyh&url=http://s8.sinaimg.cn/orignal/7a4cdec8gb4957a56a3b7)

二、领域模型中的实体类

领域模型中的实体类分为四种类型：VO、DTO、DO、PO，各种实体类用于不同业务层次间的交互，并会在层次内实现实体类之间的转化。

业务分层为：视图层（VIEW+ACTION），服务层（SERVICE），持久层（DAO）

相应各层间实体的传递如下图：

[](http://photo.blog.sina.com.cn/showpic.html#blogid=7a4cdec80100wkyh&url=http://s14.sinaimg.cn/orignal/7a4cdec8gb4957ba7c4fd)

项目中我们并没有严格遵循这种传递关系，但这种和业务层次的关联对我们理解各实体类的作用是有帮助的。（我们没有接触到PO的原因，我理解为ORM对PO进行了封装）

以下是资料的原文，上图是基于此绘制的：

**概念：**

**VO（View Object）：**视图对象，用于展示层，它的作用是把某个指定页面（或组件）的所有数据封装起来。

**DTO（Data Transfer Object）：**数据传输对象，这个概念来源于J2EE的设计模式，原来的目的是为了EJB的分布式应用提供粗粒度的数据实体，以减少分布式调用的次数，从而提高分布式调用的性能和降低网络负载，但在这里，我泛指用于展示层与服务层之间的数据传输对象。

**DO（Domain Object）：**领域对象，就是从现实世界中抽象出来的有形或无形的业务实体。

**PO（Persistent Object）：**持久化对象，它跟持久层（通常是关系型数据库）的数据结构形成一一对应的映射关系，如果持久层是关系型数据库，那么，数据表中的每个字段（或若干个）就对应PO的一个（或若干个）属性。

**模型：**

       下面以一个时序图建立简单模型来描述上述对象在三层架构应用中的位置

l        用户发出请求（可能是填写表单），表单的数据在展示层被匹配为VO。

l        展示层把VO转换为服务层对应方法所要求的DTO，传送给服务层。

l        服务层首先根据DTO的数据构造（或重建）一个DO，调用DO的业务方法完成具体业务。

l        服务层把DO转换为持久层对应的PO（可以使用ORM工具，也可以不用），调用持久层的持久化方法，把PO传递给它，完成持久化操作。

l        对于一个逆向操作，如读取数据，也是用类似的方式转换和传递，略。

三、项目中的实体类

项目中常见的实体类有VO，DO和DTO，命名规则也常是以相应字符串结尾，如\*VO.Java。但是DTO不总是遵循这个规则，而通常与他的用途有关，如写成\*Query.java，表示存储了一个查询条件。项目中实体类出现的业务层次也没有这么严格，例如我们可以在视图层就组装一个DO，也可以将一个VO从持久层传出来，所以与业务分层相关联的划分方法显得有些冗余。从项目代码中抽象出的理解是：VO对应于页面上需要显示的数据，DO对应于数据库中存储的数据，DTO对应于除二者之外需要进行传递的数据。