链滴

领域驱动设计 DDD 之值对象

作者：valarchie

原文链接：https://ld246.com/article/1573313720803

来源网站：链滴

值对象是什么?

之前我们讲到实体，它最主要的特征在于概念上的标识。而领域中存在一些不需要进行标识的对象，们主要是对事物的描述。如何理解这段话呢？可以通俗的理解，之前我们认为实体是数据库中的一条录，这条记录会发生更改，例如学生的信息表。而值对象对应到数据库中的什么呢？它们不会单独地射一张表而是可能对应表中的几个字段。比如学生信息表中拥有省、市、区、地址四个字段构成学生住址，这四个字段构成的是一个整体用来描述学生的详细地址。

为什么我们将学生详细地址建模成值对象呢?

因为我们并不需要追踪学生详细地址的变化，我们并不关系它的连续性。

在很多DDD的博客文章中讲解到值对象都会用到了地址这一概念，可能有的人会误解地址天然就是值象。并非如此，它有时可能作为值对象，有时可能作为实体。这需要看我们的业务场景。

值对象的情形:

1. 当你和你的舍友在淘宝上购买商品，你们在订单上填的地址是一样的，并不需要严格区分这两个地，快递直接送到这个地址即可。（地址映射到订单表中的几个字段上，如：省市区地址）

实体的情形:

2. 当你们宿舍的宽带坏了，需要报修，这时候你和你的舍友都打了电话报修。但是宽带人员并不会派个人来抢修宽带。因为宽带人员知道你们是同一个地址。（宿舍地址映射到宽带公司系统中的一条地用户表，以地址来区别用户）

定义:

当我们只关心一个模型元素的属性时，应把它归类为值对象。我们应该使这个模型元素能够表示出其性的意义，并为它提供相关功能。值对象应该是不可变的，不要为它分配任何标识，不要将它设计得实体一样复杂。

上面的定义，关于值对象的用来描述的事物的用途应该不难理解。那为什么规定值对象应该要不可变？我举一个很简单的例子。假设我们有三个历史订单的地址都是A市B区C街道。而我们将A市B区C街这一地址建模成实体对应数据库里的address表。于是在订单表order当中就会存在address_id用来指地址表中的A市B区C街道。那如果我们将地址表中的A市B区C街道改成X市Y区Z街道的话，其实历史单就乱套了。原本历史订单记录的是送到A市B区C街道，结果也变成了X市Y区Z街道。

例子:

如何发现领域中天然适合建模为值对象的模型呢?

```
public class Customer {

    private Long customerID;

    private String name;

    private String province;
```

```
    private String city;

    private String street;

}
```

在以上这个例子中，省、市、街道天然是一个整体，并描述了客户的地址，此时可以将这三个属性合起来成为一个值对象。

```
public class Customer {

    private Long customerID;

    private String name;

    private Address address;

}

/**
 * 地址值对象
 */
class Address{

    private String province;

    private String city;

    private String street;

}
```

值对象有助于数据库中的优化，比如在数据库中我们经常要作连表查询，而值对象的不变性使得我们以安全的将B表的信息冗余到A表当中去。举个例子，比如学生分数表含有student_id,score两个字段由于学生姓名不会更改，此时就可以将student_name作为学生分数表的冗余字段，避免学生分数表学生信息表进行连表查询的动作。这也是一种反范式的例子。

如果值对象不变的话，如何去更换值对象呢?

此时我们可以利用整体性替换去更改实体持有的值对象。如下：

```
Address currentAddress = new Address("福建", "厦门", "前埔东路");
currentAddress = new Address("广东", "深圳", "白云南路");
```

在IDDD书中，作者还建议值对象应该具有无副作用性。如何理解无副作用性呢？我举一个非常简单例子，大家都知道Java中的String字符串是不可变的，那如果对String进行replace操作的话，会影响来的字符串吗？其实是不会的。

```
String alpha = "a";
alpha.replace("a", "b");
System.out.println(alpha);
```

上面的例子依然会输出字母a，这就是一种无副作用的设计方式。在BigDecimal类的设计当中也是，BgDecimal的数学运算都不会直接修改原值，而是产生一个新值进行返回。

以上我对值对象映射到表中的某几个字段的只是了便于通俗理解，值对象也可以持久化到数据库的单表中。例子淘宝的用户地址栏，我们可以新增或者删除我们的个人地址。其实这已经将这些地址值对进行持久化了。但是订单可能并非直接引用这些地址的id，而是直接copy这些地址到订单中。

我们最后总结一下值对象的主要特征：

1. 它度量或者描述了领域中的一件东西。
2. 它可以作为不变量。
3. 它将不同的相关属性组合成了一个概念整体。
4. 它可以和其他值对象进行相等性比较。
5. 它不会对协作对象造成副作用。

关于DDD的理解各有不同，欢迎网友评论一起探讨。