链滴

领域驱动设计 DDD 之实体

作者：valarchie

原文链接：https://ld246.com/article/1573228384568

来源网站：链滴

什么是实体?

实体最主要有两点特征，一是唯一标识，二是连续性。

● 唯一标志：

当一些对象不是由属性定义，而是由一个唯一标志定义的话，我们就可以认为它是一个实体。好比我不能通过一个人的外在特征去唯一定位一个人，因为人从小到大，从年轻到衰老其外在特征都是在改的。而身份证号码可以贯穿一个人的一生而不发生变化。而且唯一标识不一定仅有一个属性表示，有能通过多个属性标识某一个唯一对象，就好比数据库中的联合外键（可能有根据电话以及姓名唯一确一个实体的情况）。

● 连续性：

对象的连续性体现在对象是有生命周期的。在这个生命周期内，对象内的属性可能是变化着的。好比行账户表，它就属于一个实体，用户的银行卡号可以唯一的确定一个人的账户，而账户的内的余额随时间变化（利息）或者随着交易变化。

但实体并非一定是映射到我们现实世界的某个具体事物。实体可以是一个人，一辆车，也可以是一次易或者是一场比赛。只要它们是满足两个条件的：一：它在整个生命周期是连续的，即变化的。可以俗的理解为数据库中的一个用户信息表，用户的信息会发生一些改变。而系统中的操作日志表，我们会再对其进行修改，我们也没有必要对某个操作日志做唯一的标识。二：它的区别不是由除了唯一标之外的属性所区分开的。比如数据库中有一张银行账户表，有卡号、余额、账户类别三个字段。我们分每个账户并不是通过余额和账户类别来区分一个银行账户，而是通过唯一的银行卡号来唯一区分。

实体不单单出现在同一张表或者同一个库，甚至实体会出现在不同的系统当中。而当我们需要区分对的时候，实体模型应该要定义出当符合什么条件才能是算相同的事务。好比支付宝系统里的用户和银系统里的用户想要打通，并对用户进行信用评分，那么他们如何知道支付宝里的某些流水和银行里的些流水同属于某一个人呢？这个时候可能就需要通过银行开户的身份证和支付宝当时实名认证的身份来进行甄别。

实体建模

```
public class Customer {

    private Long customerID;

    private String name;

    private String phone;
    // 余额
    private String balance;
    // 会员等级
    private String vipLevel;

}
```

以上对Customer的建模我们可以适当地做一下改变，想想我们对实体最根本的理解，实体最主要的是为了标识某个领域内的事物。在以上这个例子当中，余额和会员等级其实都不能去标识一个用户，name字段虽然不唯一，但我们也经常用来识别账户。而电话则是唯一的，因为注册电话是唯一的。以其实我们不仅可以通过CustomerID来识别某个用户，还可以通过name和phone来查找用户。所我们的准则是有助于识别某个领域对象的属性我们放到实体当中去。

所以在上面的例子当中我们可以将余额balance和vipLevel两个字段从Customer实体类中移出转移到

他关联对象上，并通过这些关联对象履行职责。

```
public class Customer {

    private Long customerID;

    private String name;

    private String phone;

}
```

生成实体唯一标识的四种方式

1. 用户提供一个或者多个初始唯一值作为输入时，例如注册账户时填入自定义用户名或者邮箱或者电等等。我们的系统必须保证这个值的唯一。
2. 程序内部通过某种算法自动生成身份标识，例如UUID、雪花ID等机制去生成唯一算法。
3. 程序依赖于持久化存储，比如数据库生成的自增主键
4. 通过其他的限界上下文决定出的唯一标识，作为程序的输入。

在某种特殊情况下，我们的实体是利用两个属性来进行标识对象，类似于数据库表中的联合主键，但我们的数据库框架如Hibernate或者Mybatis实际修改表中的数据可能用到的是表中的id字段如updatByPrimaryKey()此类方法，这个时候我们可以让实体继承一个拥有id属性的抽象类，以解决上述问题。

```
public abstract class IdentifiedDomainObject implements Serializable {

    private long id;

    protected long getId() {
        return id;
    }

    protected void setId(long id) {
        this.id = id;
    }

}
```

实体的不变性。

有时候一个实体维护了一个或者多个不变条件，可以通俗的理解为我们的业务逻辑规则，比如一条订的支付金额不可能大于商品总额，也不可能为负数。实体必须在整个生命周期中都保持这种一致性否就是错误的。不变条件主要是由聚合所关注（后文会谈到）。我们可能还需要去验证实体内的属性是符合我们的要求，如不能为null，不能为空串，长度不能大于100，需满足一定格式等等。我们可以为实体填入参数的时候进行判别。

关于DDD的理解各有不同，欢迎网友评论一起探讨。