链滴

领域驱动设计 DDD 之资源库

作者：valarchie

原文链接：https://ld246.com/article/1574311683761

来源网站：链滴

为什么使用资源库?

如果完全按照领域模型的角度，完全通过遍历对象的方法来获取所有关联的对象。这种模型会过于错复杂。对象嵌套的层级或者关联的层级非常深。例如通过Customer.order.product.price层层遍历来取当时客户订单的商品的价格。

那如果完全按照数据库模型的角度，模型中的对象不需要完全连接起来，对象关系网就能保持在一个控范围。但是这又会回到之前传统开发模式中，零散的使用各个DAO从各个表抽取数据自行拼凑出我想要的模型。

这里就会出现一个问题，Customer类需要保持客户所有已订的Order，还是通过CustomerID在数据中查找Order列表呢?

对象关联还是纯数据库?

客户需要一种有效的方式来获取已存在的领域对象的引用。如果基础设施提供了这方面的遍历，那么发人员可能会增加很多可遍历的关联，这会使模型变得非常混乱。另一方面，开发人员可能使用查询数据库中提取他们所需的数据，或是直接提取具体的对象，而不是通过聚合的根来得到这些对象。这就导致领域逻辑进入查询和客户代码中，而实体和值对象变成了单纯的数据容器。采用大多数处理数库访问的技术复杂性很快就会使客户代码变得混乱，这将导致开发人员简化领域层，最终使模型变得关紧要。

我们需要找到一个恰当的方式。

我们不需要对那么很容易通过遍历来找到的持久对象进行查询访问，如不需要单独的去获取地址，而以通过Person对象来获取地址，意味着当我们获取Person对象的时候，地址值对象已经是填充好的我们不需要特地利用另外一个资源库的方法，获取地址再进行填充。但当获取Person的所有订单记录，由于订单的数量有时候过于庞大，加载Person的时候其实并不一定需要查看订单记录，我们应该使Repository的另外方法来获取Person的订单记录。

对值对象的全局搜索通常是没有意义的。如果确实需要在数据库中搜索一个已知的值对象，那么值得虑一下，搜索结果可能实际上是一个实体，尚未识别出它的标识。

定义

在所有持久化对象中，有一小部分必须通过基于对象属性的搜索来全局访问。当很难通过遍历的方式访问某些聚合根的时候，就需要这种访问方式，它们通常是实体，有时可能是具有复杂内部结构的值象。而其他对象则不适合使用这种访问方式，因为这会混淆它们之间的重要区别。随意的数据库查询破坏领域对象的封装和聚合。技术基础设施的数据访问机制的暴露会增加客户的复杂度，并妨碍模型动的设计。

如何使用资源库

我们应该将资源库看作一个对象的集合。

客户使用查询方法向资源库请求对象，这些查询方法根据客户所指定的条件来挑选对象。资源库检索请求的对象，并封装数据库查询和元数据映射机制。他们可以返回汇总的信息，如多少个实例满足条。甚至返回汇总计算，如所有匹配对象的某个数值属性的总和。

这样一来客户就只需与一个简单的、易于理解的接口进行对话，并根据模型向这个接口提出它的请求。

为每种需要全局访问的对象类型创建一个资源库，这个资源库相当于该类型的所有对象在内存的一个

合的“替身”。通过一个众所周知的全局接口来提供访问。提供添加和删除对象的方法，用这些方法封装在数据存储中实际插入和删除数据的操作。根据提供具体条件来挑选对象的方法，并返回属性值足条件的对象或者对象集合，从而将实际的存储和查询技术封装起来。只为那些确实需要直接访问的合根提供资源库，让客户始终聚焦于模型，而将所有对象的存储和访问操作交给资源库来完成。

让用户无感知的，以为就在内存中使用一个集合一样。

资源库的方法设计：

方法参数：

1. 唯一标识
2. 复杂的参数组合
3. 值域（日期范围）

返回值：

1. 对象
2. 对象集合
3. 某些类型的汇总。

资源库的实现

1. 对类型进行抽象，比如有奔驰车宝马车，不应有两种资源库，而应该是只有一个抽象车类的资源库。
2. 充分利用与客户进行解耦。
3. 将事务的控制权交给客户。

资源库的优点

1. 为客户提供了一个简单的模型，可用来获取持久化对象并管理他们的生命周期。
2. 他们将应用程序和领域设计与持久化技术进行解耦。
3. 它们体现了有关对象访问的设计决策。
4. 很容易测试，将利用集合直接替换资源库进行测试。

资源库中如何管理事务

对事务的管理绝对不应该放在领域模型和领域层中。通常来说，与领域模型相关的操作都非常的细粒，以致于无法用于管理事务。另外，领域模型也不应该意识到事务的存在。通常我们将事务放在应用，然后为每个主要的用例创建一个门面，一个用例对应一个门面业务方法，如果所有操作安全完成的，门面中的业务方法提交事务，否则失败回滚。

关于DDD的理解各有不同，欢迎网友评论一起探讨。