链滴

领域驱动设计 DDD 之工厂

作者：valarchie

原文链接：https://ld246.com/article/1574177012526

来源网站：链滴

为什么需要工厂

1. 当创建一个复杂对象或聚合的过程很复杂并且暴露出了过多的内部结构时，我们则可以使用工厂进封装。一个对象在它的声明周期中要承担大量的职责，如果再让复杂对象负责自身的创建，那么职责载将会导致问题。

2. 我们设计好领域模型供客户方调用，但如果客户方也必须使用如何装配这个对象，则必须知道对象内部结构。好比你去驾校学车，却得先学会发动机的原理。对客户方开发来说这是很不友好的。其次复杂对象或者聚合当中的领域知识（业务规则）需要得到满足，如果让客户方自己装配复杂对象或聚的话，就会将领域知识泄露到客户方代码中去。

3. 对象的创建本身可以是一个主要操作，但被创建的对象并不适合承担复杂的装配操作。将这些职责在一起可能产生难以理解的拙劣设计。让客户直接负责创建对象又会使客户的设计陷入混乱，并且破被装配对象或聚合的封装，而且导致客户与被创建对象的实现之间产生过于紧密的耦合。

最重要的一点就是隐藏创建对象的细节

定义

复杂对象的创建是领域层的职责，但这项任务并不一定属于那些用于表示模型的对象，他们没有对应型中的事物，但又确实承担了领域层的职责。

应该将创建复杂对象和聚合的职责转移给单独的对象，这个对象本身可能没有承担领域模型中的职责但它仍然领域设计的一部分。提供一个封装所有复杂装配操作的接口，而且这个接口不需要客户引用被实例化的对象的具体类。在创建聚合时要把它作为一个整体，并确保它满足固定规则。

工厂的设计要点

1.每个创建方法都应该是原子的，并保证生成的对象处于一致的状态。

2.可以使用独立的工厂或者在聚合根上使用工厂方法。当A对象的创建主要使用了B对象的数据或者规时，那么可以在B对象上创建一个工厂方法来生成A对象。

3.以下情况只需使用构造函数即可。

- 类仅仅是一种类型，没有其他子类，没有实现多态性。
- 客户关心的是实现类。
- 客户可以访问对象的所有属性，因此向客户公开的构造函数中没有嵌套的对象创建。
- 构造过程很简单。
- 公共构造函数必须遵守与工厂相同的规则，必须是原子操作且满足所有固定规则。
- 不要在构造函数中调用其他构造函数，应保持构造函数的简单。

4.工厂方法的参数应该是较低层的对象。比如装配一辆汽车，应该传入较低层抽象的轮胎，发动机等象。

当我们利用购物车模型进行结算的时候，可以从购物车模型的下单方法去生成一个订单模型。这就通购物车聚合的工厂方法去生成了订单聚合。

举个例子

我们以一个论坛对象发起一个讨论为例

```
public class Forum {

    public Discussion startDiscussion(DiscussionId aDiscussionId, Author anAuthor,
                                      String aSubject) {

        if(this.isClosed()){
            throw new IllegalStateException("Forum is closed!");
        }


        Discussion discussion = new Discussion(this.tenant(), this.forumId(),
                                           aDiscusstionId, anAuthor, aSubject);

        // .. 发布领域事件


        return discusstion;

    }


}
```

客户端如何使用这个模型呢?

```
// 由论坛实体生成这个讨论
Discussion discussion = aForum.startDiscussion(this.discussionRepository.nextIdentity(),
new Author("jdoe", "John Doe", "jdoe@gmail.com"),
"Dealing with Aggregate Concurrency Issues");
// 保存这个讨论
this.discussionRepository.add(discussion);
```

工厂不仅达到了封装创建对象的细节，并有效的表达了限界上下文中的通用语言，减轻客户端在创建聚合实例时的负担，确保所创建的实例处于正确的状态（符合业务规则）。

工厂有多种形式，可以是一个独立的Factory对象，也可以是聚合根上的工厂方法，也可以是领域服
。
工厂不仅用于对象生命初期的创建，还用于在对象生命周期的中期从数据库中的数据装配成聚合的情
。