

# 第02期-DDD领域驱动设计

2019年11月12日 15:33

## 概念

DDD (Domain-Driven Design 领域驱动设计) 是由 Eric Evans 最先提出, 目的是对软件所涉及到的领域进行建模, 以应对系统规模过大时引起的软件复杂性的问题。整个过程大概是这样的, 开发团队和领域专家一起通过通用语言去理解和消化领域知识, 从领域知识中提取和划分为一个一个的子领域(核心子域, 通用子域, 支撑子域), 并在子领域上建立模型, 再重复以上步骤, 这样周而复始, 构建出一套符合当前领域的模型。

要理解什么是领域驱动设计, 首先要理解什么是领域, 什么是设计, 还有驱动是什么意思, 什么驱动什么。

问题领域--->领域驱动建模(通用语言) <---模型驱动代码实现

最大价值我觉得是: 当我们要开发一个系统时, 应该尽量先把领域模型想清楚, 然后再开始动手编码, 这样的系统后期才会很好维护。但是, 很多项目都是一开始模型没想清楚, 一上来就开始建表写代码, 代码写的非常冗余。

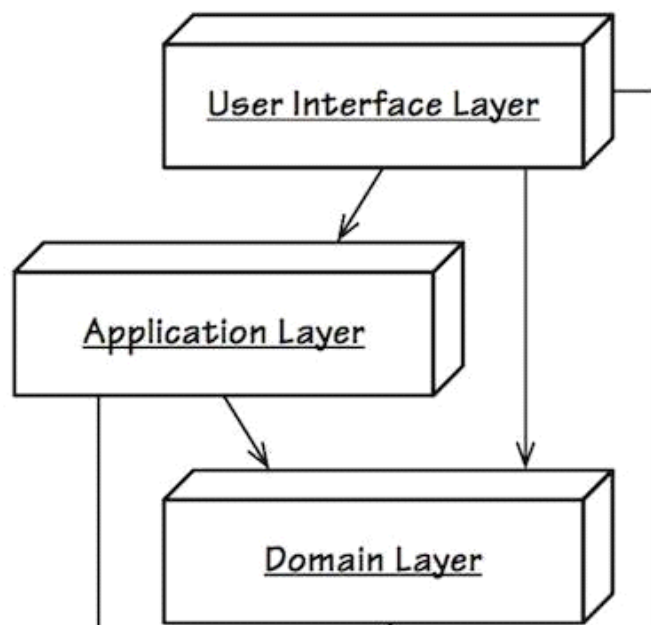
虽然, 我们都知道这个道理, 但是我也明白, 人的习惯很难改变的, 大部分人都很难从面向过程式的想到哪里写到哪里的思想转变为基于系统化的模型驱动的思维。我想, 这或许是DDD很难在中国或国外流行起来的原因吧。

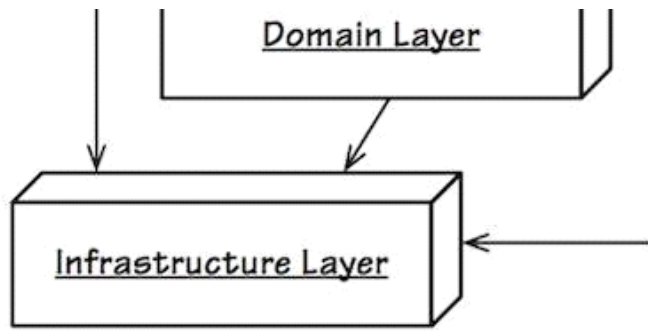
模型是关键, 代码只是让模型落地的一种工具。

建模方法: 领域、限定上下文、通用语言、UML 工具。

## 分层架构

Infrastructure(基础设施层), Domain(领域层), Application(应用层), Interfaces(表示层, 也叫用户界面层或是接口层), 各个层面的作用下面介绍。





### Infrastructure 基础设施层

向其它层提供通用的技术能力（比如工具类、第三方库类支持、常用基本配置、数据访问底层实现）

### Domain 领域层

主要负责表达业务概念，业务状态信息和业务规则，是整个系统的核心层，几乎全部的业务逻辑会在该层。

### Application 应用层

相对于领域层，应用层是很薄的一层，应用层定义了软件要完成的任务，要尽量简单。

### User Interfaces 用户接口层

负责向用户显示信息和解释用户命令，请求应用层以获取用户所需要展现的数据(比如获取首页的商品数据)

在层级结构中，上层模块调用下层模块提供的服务，这里就会存在一种依赖关系，使用倒置原则解决依赖：

*上层模块不应该依赖于下层模块，两者都应该依赖于抽象；*

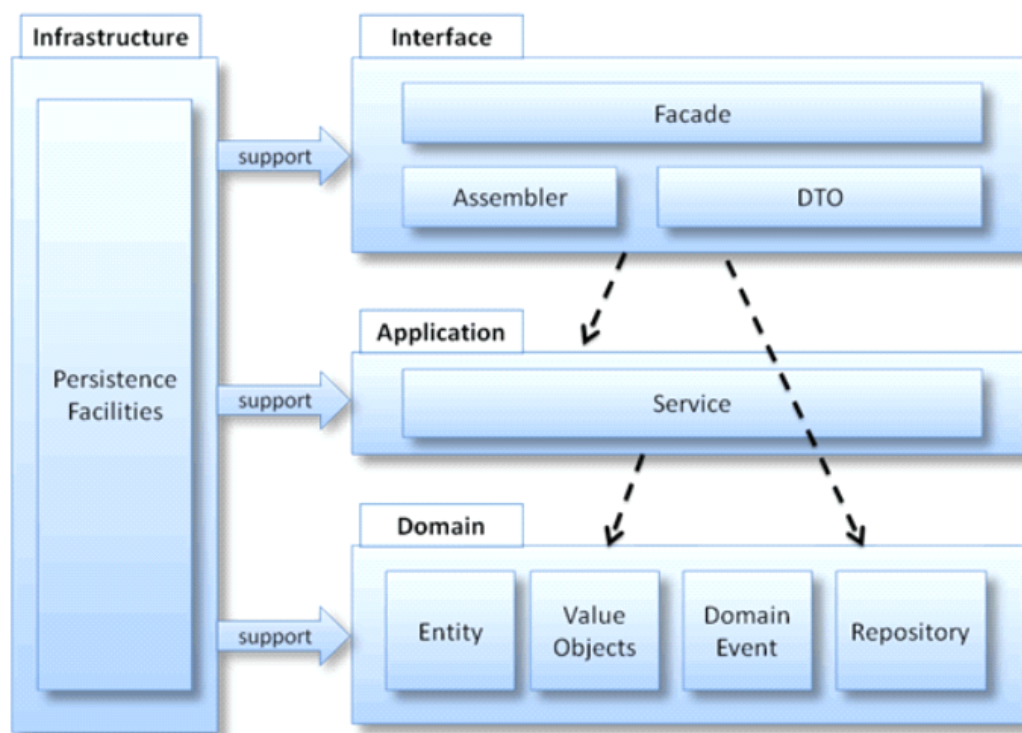
*抽象不应该依赖于实现，实现应该依赖于抽象；*

实际上就是：面向接口编程的思想，让接口对外，我们依赖于接口，这样我们不关心实现，实现随时可换，对我们代码无影响，以后内部实现变了，对我们没有影响，不需要修改代码，有了依赖注入容器后，它帮我们管理这种依赖接口和实现的关系，甚至生命周期，我们就更加简单了。

### DDD元素组成部分

在使用DDD设计系统时，主要包括：

Entity、Value Object、Service、Domain Event、Aggregate、Repository、Factory 和 Moudle



## Entity 实体

在建模时，Entity可以用来代表一个事物。既然Entity是用来代表一个事物的，那么它就应该有一个唯一值来标识这个事物，同时，他还能记录这个事物状态的变化。比如：Id为5的Person对象，它的名字是张三，过两天，他将自己的名字改为李四。但是，这个人（Id=5）还是这个人（Id=5），只是他名字的状态以及发生了变化，而这个变化后的状态被Person这个Entity记录了下来。

## Value Object 值对象

用来描述事物的某一方面的特征，所以它是一个无状态的，且是一个没有标识符的对象，这是和Entity的本质区别。拿订单来说，一张订单在系统中应该有一个唯一的标识，且具有状态，所以订单是一个Entity对象，而这张订单共¥100元，则是描述了这个订单的总额特征，¥100元可以用值对象表示为{100，¥}，及金额和币种的组合。{100，¥}在任何限界上下文中都描述的是¥100元，是因为他们比较的是里面的内容（金额和币种），而上面的张三在修了名后，还是原来的那个人，是因为它比较的是唯一标示ID的值。

又比如说：送货地址应该设计成值对象，一个地址就是唯一的，而且固定不变的，无状态的，很难说这个地址随着时间的推移会变化，也不能说这个地址在全球有二义性。

## Service 服务

应用服务和领域服务，都是服务，服务是行为的抽象。

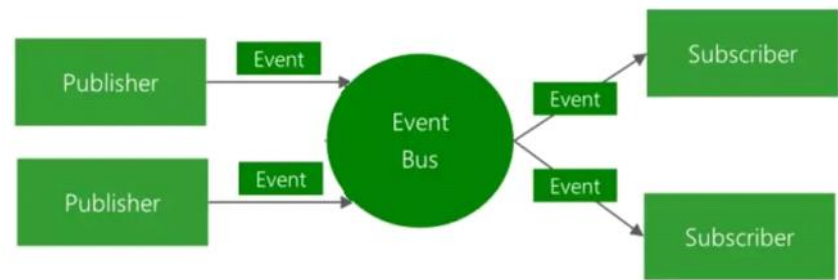
应用服务：负责展现层与领域层之间的协调，协调业务对象来执行特定的应用程序任务，它不包含业务逻辑。

领域服务：负责表达业务概念，业务状态信息以及业务规则，是业务软件的核心。

## Domain Event 领域事件

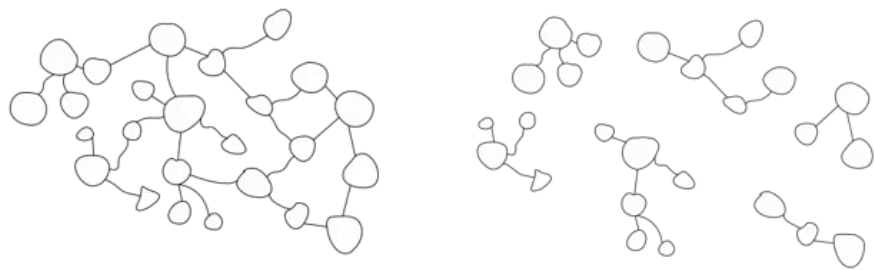
简而言之，领域事件是用来捕获领域中发生的具有业务价值的一些事情。它的本质就是事件，不要将其复杂化。

当用户在购物车点击结算时，生成待付款订单，若支付成功，则更新订单状态为已支付，扣减库存，并推送捡货通知信息到捡货中心。



### Aggregate 聚合根

相关对象的集合，它作为数据修改的基本单元，为数据修改提供了一个边界。每个聚合都有一个根和一个边界，根也是一个实体，聚合边界以外的对象只能通过根对聚合内部元素操作。聚合将一组相关的对象内聚到一起，从而将系统的复杂程度降低。



### Repository 仓储

来存储聚合，相当于每一个聚合都应该有一个仓库实例，仓储是原则上是领域模型与持久化存储之间明确的契约。

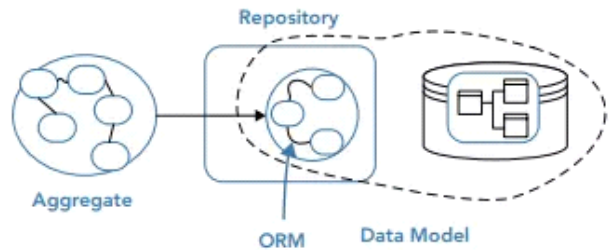
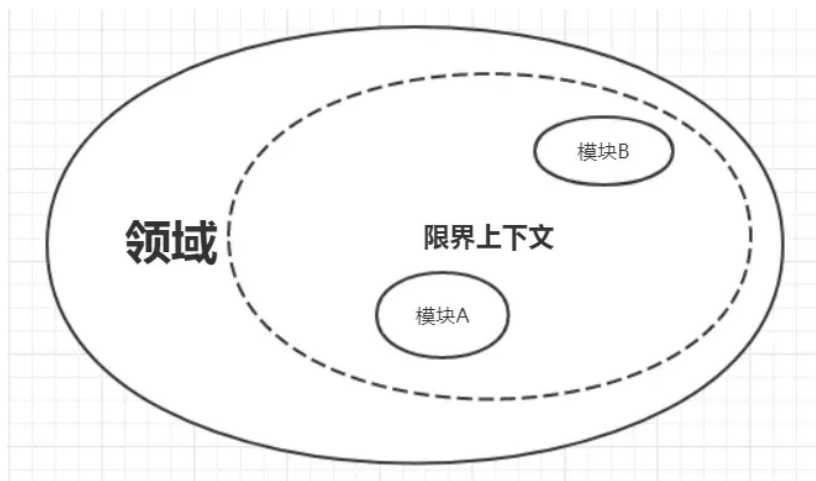


FIGURE 21-1: An ORM maps between the domain and persistence model.

### Moudle 模块

通过分解领域模型为不同的模块，以降低领域模型的复杂性，提高领域模型的可读性。



## Factory 工厂

DDD中工厂的主要目标是隐藏对象的复杂创建逻辑，次要目标就是要清楚的表达对象实例化的意图。而工厂模式是计模式中的创建类模式之一，借助工厂模式我们可以很好实现DDD中领域对象的创建。

## 有用的文章

<https://www.cnblogs.com/netfocus/category/361987.html>

<https://www.jianshu.com/c/27b2faf4ed7f>