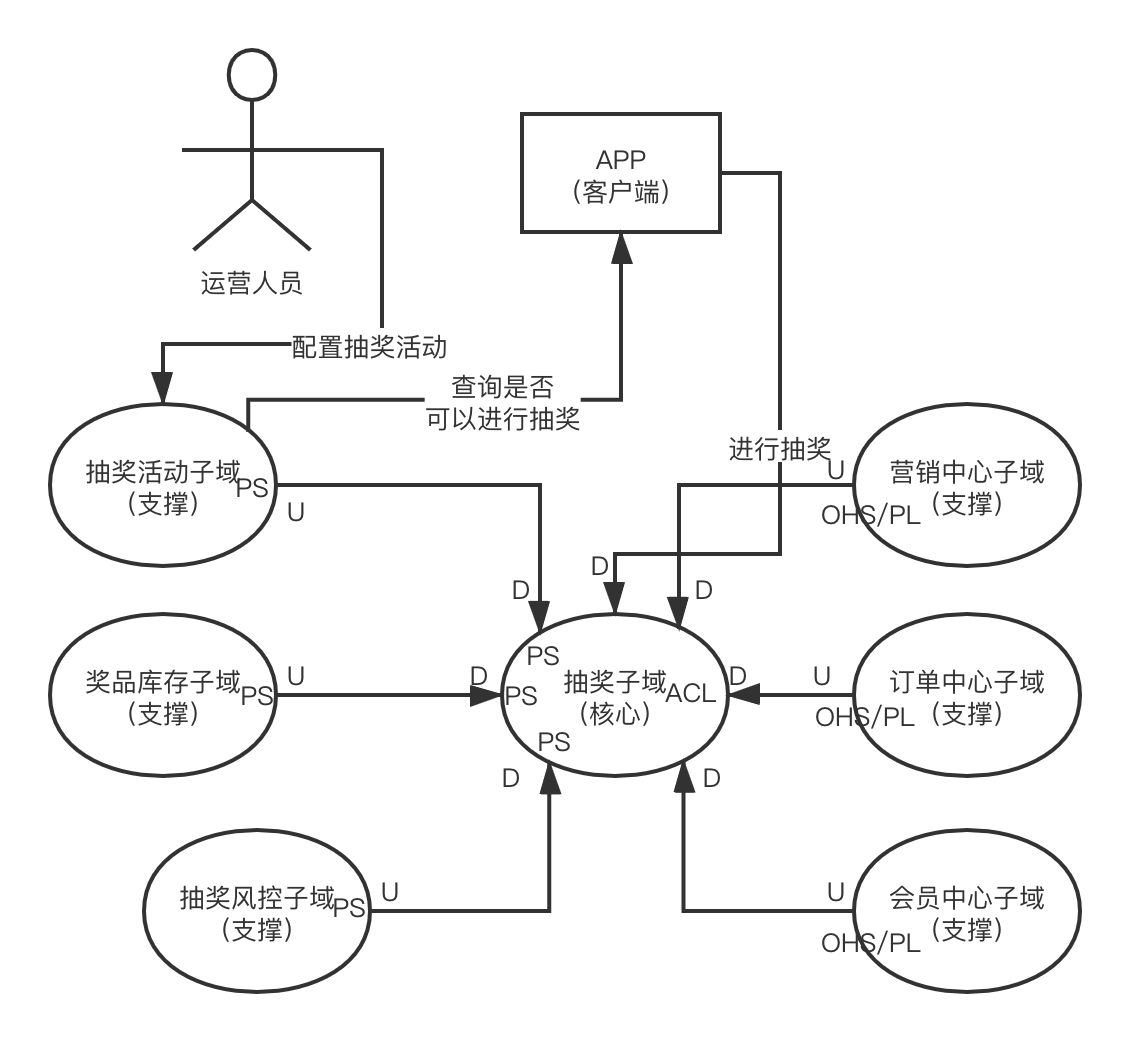
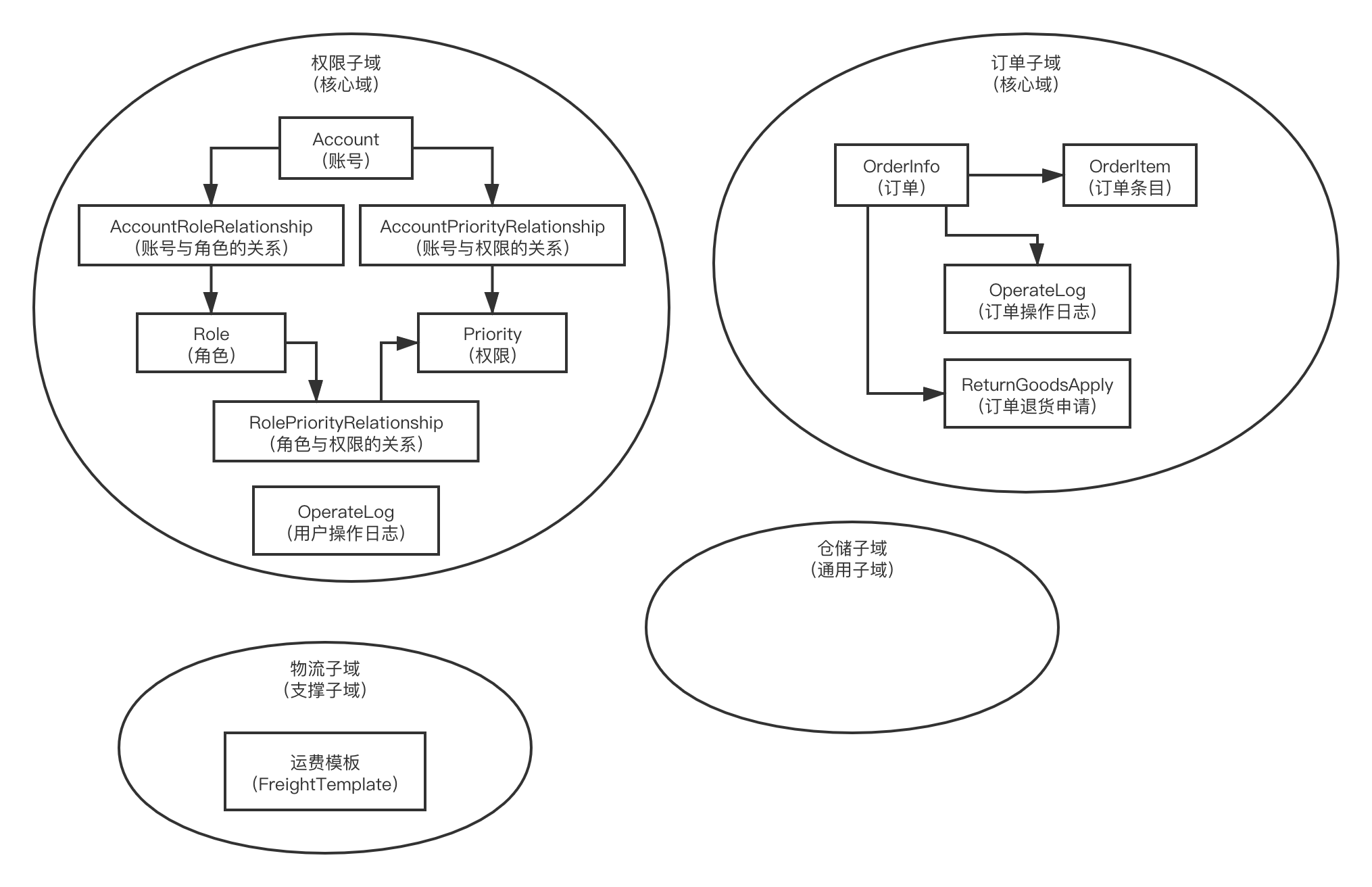
# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/041~060资料/抽奖系统DDD建模.png**



# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/011~020资料/电商系统v1.0的DDD建模.png**



# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/061~080资料/061\_完善抽奖系统的DDD建模案例的代码实现（1）/笔记.docx**

3616118202

《061\_完善抽奖系统的DDD建模案例的代码实现（1）》

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/061~080资料/073\_促销中心DDD建模是否还需要战略建模？/笔记.docx**

《073\_促销中心DDD建模是否还需要战略建模？》

项目阶段一里是有过营销中心服务开发的，其实一般电商的营销模块说白了就是可以设置各种营销活动，什么满减、秒杀、买赠、优惠券之类的，你都可以配置，有的活动是针对全场的，有的是针对某种商品的，然后还应该会提供对应的营销活动下的一些计算工作，比如计算价格的减免，或者是商品的赠予计算，或者是购买商品后发券，等等

回顾一下阶段写的营销中心的一些代码逻辑，当时是做的比较简化的

很糟糕，没有任何业务建模在里面，全是传统的CRUD，甚至数据模型用了JSON来替代，导致代码的可读性和可维护性特别差，这样的代码，在初期快速开发和上线是没问题的，但是后续持续迭代，人越来越多，这种代码几乎是无法维护的

我们认为是不需要了，因为其实就认为这个营销中间自己本身就是一个子域就可以了，就是一个营销域，营销域会跟其他的子域进行上下文映射和集成，这个我们其实都不用过多的考虑，重点关注的是营销域内部的战术建模就可以了

这里给大家讲一个技巧，比如你是个P6/P7，负责了一个营销子系统，带了几个人，也可能就你一个人，此时你可以选择按我们之前说的，营销系统内部再拆分N个服务，这样你就可以对营销系统做个战略建模了，内部划分子域，然后搞子域的映射和集成

但是也可以默认为是你营销子域就是一个独立子域，内部不划分子域了，你就内部建立一套通用语言模型，进行统一战术建模就可以了，其实战术建模说的太夸张了，说白了就是内部进行实体建模

至于你自己玩了自己系统的实体建模，内部全部按DDD来建模了一遍，还原真实/虚拟世界的业务流程之后，你如何跟其他系统集成呢？其实这里要讨巧一点，有可能其他系统根本就不是DDD建模的，也不想DDD建模，他们还是就希望你传统的提供一些接口就可以了

那此时你就可以还是提供一些接口出去，但是每个接口的业务逻辑，并不是跟以前一样，还调用service+dao之类的，而是还原出来实体，然后调用实体的行为，让内部多个实体串联起来跑，还原真实/虚拟世界的业务流程，让代码看着特别漂亮

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/061~080资料/062\_完善抽奖系统的DDD建模案例的代码实现（2）/笔记.docx**

《062\_完善抽奖系统的DDD建模案例的代码实现（2）》

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/061~080资料/068\_完善抽奖系统的DDD建模案例的代码实现（8）/笔记.docx**

3616118202

《068\_完善抽奖系统的DDD建模案例的代码实现（8）》

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/061~080资料/075\_促销中心DDD重构的起点：找产品聊聊天（2）/笔记.docx**

《075\_促销中心DDD重构的起点：找产品聊聊天（2）》

如果你想要对自己负责的营销中心进行DDD建模重构，那么就是跳过战略建模部分，也就是系统边界划分以及集成关系确定这些都跳过去了，然后直接进行内部的实体建模，那么首先就是得对你的系统里的核心关键术语，也就是所谓的通用语言吧，做一个提取

那么提取你负责系统里的核心术语，怎么提取呢？当然不能是你技术RD拍板了，你定的术语可能别人都不能理解，一般一个技术产品的负责人，其实不是RD，站在公司老板和高层的角度，他看到的是一个一个的系统产品，而不是系统背后的技术架构和技术逻辑

所以任何一个公司决定一个技术类产品的，并不是技术RD，而是产品经理

这个产品经理能沟通技术产品的使用方，外部C端/B端用户，or内部的公司其他人，比如运营之类的，还能沟通技术这边的RD小哥哥，还能把产品讲给高层领导听，所以产品经理才是任何一个技术类产品的核心人物

他们了解产品的使用方需求，会决定产品的形态，推动产品的 研发，管理产品的研发进度和项目进展，担负起产品的交付和上线，关注产品线上运行的情况，是否有bug，了解竞对产品情况，分析产品优劣势，等等

当然我并不是说国内每一个产品经理都这么牛，我只是说产品经理这个角色在一个技术类产品里的定位，无论是一个APP、网站还是一个系统，他的定位其实是极为重要的，技术负责人本质是对产品经理进行follow跟随的

因为技术负责人的任务，就是想尽办法把产品做出来，产品各项性能指标都很好，不出bug，研发速度要快，能管理技术团队，产品有问题随时解决，产品使用量大了，可以设计好的技术架构，或者进行技术升级

当然国内经常说RD和PM的矛盾很深，那是因为国内的PM大多都达不到我说的要求，28法则吧，大部分的产品经理的能力都是不专业的，能力平庸，还喜欢瞎搞，所以搞的RD小哥哥经常很愤怒

但是不管怎么说吧，哪怕你合作的PM能力有点渣，你也得找他聊，因为谁让他那么重要呢？所以这里关键第一步，你必须找他仔细聊聊，把你负责系统的各项功能以及业务流程都梳理的极为清晰

同时跟产品经理一起，把系统里的核心术语都提炼出来，不要有歧义，达成共识

举个例子吧，比如说针对“物流系统的发货单“这个概念，你要用什么术语来表达？可能你自己觉得是发货单，但是产品经理认为如果考虑到公司高层、运营、仓储等各个部门的人员平时熟悉的概念，更好的概念应该是发货出库单

所以说，营销系统也是一样的

首先，你必须找你营销系统的PM聊聊，那个PM他既然负责营销系统，就会去调研竞对产品的营销方式 有哪些，业内常见的营销方式有哪些，然后公司内部打算用哪些营销方式，每一种营销方式具体的使用规则是什么，他会跟公司内部的运营推广的部门的用户聊，看他们对营销方式的理解是什么

所以PM应该是这块的一个业务专家的角色，他是绝对熟悉和了解业务的

所以你必须跟他聊聊，确定下来你的营销系统内有哪些专业术语，然后你们统一口径，确定这些专业术语的叫法

这就是你进行DDD实体建模的第一步，跟PM沟通，提炼出你系统的一些核心术语，对术语的名称和定义都有是清晰无歧义，多方达成共识的（包括PM、RD、外部用户、内部用户、其他部门、公司高层）

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/061~080资料/072\_基于促销中心的现有代码分析贫血模式的缺陷/笔记.docx**

《072\_基于促销中心的现有代码分析贫血模式的缺陷》

项目阶段一里是有过营销中心服务开发的，其实一般电商的营销模块说白了就是可以设置各种营销活动，什么满减、秒杀、买赠、优惠券之类的，你都可以配置，有的活动是针对全场的，有的是针对某种商品的，然后还应该会提供对应的营销活动下的一些计算工作，比如计算价格的减免，或者是商品的赠予计算，或者是购买商品后发券，等等

回顾一下阶段写的营销中心的一些代码逻辑，当时是做的比较简化的

很糟糕，没有任何业务建模在里面，全是传统的CRUD，甚至数据模型用了JSON来替代，导致代码的可读性和可维护性特别差，这样的代码，在初期快速开发和上线是没问题的，但是后续持续迭代，人越来越多，这种代码几乎是无法维护的

我们认为是不需要了，因为其实就认为这个营销中间自己本身就是一个子域就可以了，就是一个营销域，营销域会跟其他的子域进行上下文映射和集成，这个我们其实都不用过多的考虑，重点关注的是营销域内部的战术建模就可以了

这里给大家讲一个技巧，比如你是个P6/P7，负责了一个营销子系统，带了几个人，也可能就你一个人，此时你可以选择按我们之前说的，营销系统内部再拆分N个服务，这样你就可以对营销系统做个战略建模了，内部划分子域，然后搞子域的映射和集成

但是也可以默认为是你营销子域就是一个独立子域，内部不划分子域了，你就内部建立一套通用语言模型，进行统一战术建模就可以了，其实战术建模说的太夸张了，说白了就是内部进行实体建模

至于你自己玩了自己系统的实体建模，内部全部按DDD来建模了一遍，还原真实/虚拟世界的业务流程之后，你如何跟其他系统集成呢？其实这里要讨巧一点，有可能其他系统根本就不是DDD建模的，也不想DDD建模，他们还是就希望你传统的提供一些接口就可以了

那此时你就可以还是提供一些接口出去，但是每个接口的业务逻辑，并不是跟以前一样，还调用service+dao之类的，而是还原出来实体，然后调用实体的行为，让内部多个实体串联起来跑，还原真实/虚拟世界的业务流程，让代码看着特别漂亮

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/061~080资料/079\_确定促销中心的核心术语属于哪种领域对象？（1）/笔记.docx**

3616118202

《079\_确定促销中心的核心术语属于哪种领域对象？（1）》

促销活动：促销中心的核心就是管理一个一个的促销活动，让运营部门去创建不同的促销活动，配合上一些促销活动的站内 or 站外的营销推广，进而拉动老用户的活跃度和复购率，或者是在站外进行拉新，每一种促销活动都包含了几个维度，包括了时间范围、人群范围、商品范围，等等

促销类型：这个其实我们之前就有过这个概念，促销是有很多种不同的类型的，我们认为完全是可以全部集成到促销这个概念里去的，那当然每一种促销活动都有一种促销类型，比如满减、买赠、直降、优惠券、秒杀，等等

促销规则：顾名思义，根据促销类型定义出来的具体促销规则，这个很灵活多变，之前我们都是考虑用JSON存储的

促销范围：每一个促销活动都会匹配一个促销范围，这个促销范围实际上是通过时间范围、人群范围、商品范围、平台范围（APP or 网站），等各种范围来确定的

可参加促销活动查询：大家如果平时玩儿电商APP的话就知道，其实一般来说，促销活动会通过如下方式展现给你，要不然是在电商首页搞一些促销活动的专题页，要不然就是你浏览某个商品的时候，商品详情页里会提示促销活动，要不然就是你把商品加入购物车之后，在购物车浏览界面里可以看到每个商品参加的促销活动，当你提交订单之后，在订单确认页里，也会提示你每个商品参加的促销活动以及每个商品最终的价格，所以在电商的各个环节里，都需要去根据用户所属的人群范围、当前时间、浏览的商品范围，到促销中心查询对应的促销活动列表

优惠结果计算：这个一般是用于订单确认页面的，当你选购了一些商品进入订单确认页面时，如果大家平时玩儿淘宝可能会有印象，就是一个商品可能会同时参与多种促销活动，此时你可以在订单确认页里选择这个商品你最终要参加哪个促销活动，然后根据你选择的促销活动，促销中心可以提供优惠价格计算的服务，要不然是赠送商品给你，要不然是提示可以赠送你几个优惠券，要不然是提示你减免后的价格，甚至有时一个商品可以同时参与多个活动

促销活动优先顺序查询：上面说了，有的商品可能可以同时参加多种促销活动，那么在优惠结果计算的时候，需要按照不同促销活动的优先顺序来依次进行计算

促销活动互斥判断：当然一个商品可以同时参加多个促销活动，也可以设置让多个促销活动进行互斥，也就是说，商品要么选择这个促销活动，要么选择另外一个促销活动

其实分析清楚了以后，你会发现，促销中心的核心就是促销活动、促销类型、候选促销活动查询、优惠结果计算，然后促销活动优先顺序以及促销活动互斥都是依附于促销活动本身的一些关键属性，在优惠结果计算的时候会用到

到此为止，我们就把促销中心的核心关键术语以及定义都提炼出来了

在确定了这些关键术语之后，接下来很关键的一步，就是要确定这些关键术语分别术语什么领域对象？因为在DDD里，要不然是实体，要不然是值对象，实体是有唯一标识id的，值对象其实是没有唯一标识的，或者即使他在数据库里有唯一标识，但是这个唯一标识在业务上没有意义的

所以上面我们分析一下，明确能够发现，促销活动当然是实体了，因为不同的促销活动是有唯一标识的，这个在业务上都是有含义的，然后促销类型当然是值对象了，虽然促销类型可以在数据库里存储单独的表，也可以有数据库层面的ID主键，但是其实他在业务含义里，那个标识是没有意义的

同理的就是促销规则和促销范围，每一个促销范围在数据库里可以单独存一个表，但是其实他本质是促销活动的一个附属数据而已，即使有数据库id也是没有业务上的意义的

这里就可以确定一个聚合根了，那就是优惠活动是聚合根，里面包含了促销类型、促销规则、促销范围

然后对于候选促销活动查询、优惠结果计算、促销活动优先顺序查询、促销活动互斥判断，这些其实本质都是属于行为，并不是在实体/值对象的范围内的，如果是纯行为逻辑的一些东西，那么最好就是把他们建模为领域服务这个概念

说白了，就是把他们还是还原为类似以前的Service层那个概念

比如候选促销活动查询，可能对外还是提供一个接口出去；然后优惠结果计算也是提供一个接口出去，作为一个领域服务接口，就是营销域的服务接口，然后促销活动优先顺序查询、促销活动互斥判断可以设计为内部的领域服务Service，让优惠结果计算去调用

其实最终一个正规的DDD建模好的字域对外暴露的，应该是领域模型（实体模型、值对象模型）+ 对外的领域服务接口 + 对内的实体行为调用接口，实体模型/值对象模型是让其他子域进行实体行为调用的，然后实体引用内部封装了RPC调用，请求到你的实体行为调用接口，你内部还原为实体再调用实体的行为

对外的领域服务接口，就是你领域提供的一些行为，直接提供出去就可以了

这是我比较推荐的一种DDD之间的具体落地的不同子域的集成模式

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/061~080资料/074\_促销中心DDD重构的起点：找产品聊聊天（1）/笔记.docx**

《074\_促销中心DDD重构的起点：找产品聊聊天（1）》

如果你想要对自己负责的营销中心进行DDD建模重构，那么就是跳过战略建模部分，也就是系统边界划分以及集成关系确定这些都跳过去了，然后直接进行内部的领域对象建模，那么首先就是得对你的系统里的核心关键术语，也就是所谓的通用语言吧，做一个提取

那么提取你负责系统里的核心术语，怎么提取呢？当然不能是你技术RD拍板了，你定的术语可能别人都不能理解，一般一个技术产品的负责人，其实不是RD，站在公司老板和高层的角度，他看到的是一个一个的系统产品，而不是系统背后的技术架构和技术逻辑

所以任何一个公司决定一个技术类产品的，并不是技术RD，而是产品经理

这个产品经理能沟通技术产品的使用方，外部C端/B端用户，or内部的公司其他人，比如运营之类的，还能沟通技术这边的RD小哥哥，还能把产品讲给高层领导听，所以产品经理才是任何一个技术类产品的核心人物

他们了解产品的使用方需求，会决定产品的形态，推动产品的 研发，管理产品的研发进度和项目进展，担负起产品的交付和上线，关注产品线上运行的情况，是否有bug，了解竞对产品情况，分析产品优劣势，等等

当然我并不是说国内每一个产品经理都这么牛，我只是说产品经理这个角色在一个技术类产品里的定位，无论是一个APP、网站还是一个系统，他的定位其实是极为重要的，技术负责人本质是对产品经理进行follow跟随的

因为技术负责人的任务，就是想尽办法把产品做出来，产品各项性能指标都很好，不出bug，研发速度要快，能管理技术团队，产品有问题随时解决，产品使用量大了，可以设计好的技术架构，或者进行技术升级

当然国内经常说RD和PM的矛盾很深，那是因为国内的PM大多都达不到我说的要求，28法则吧，大部分的产品经理的能力都是不专业的，能力平庸，还喜欢瞎搞，所以搞的RD小哥哥经常很愤怒

但是不管怎么说吧，哪怕你合作的PM能力有点渣，你也得找他聊，因为谁让他那么重要呢？所以这里关键第一步，你必须找他仔细聊聊，把你负责系统的各项功能以及业务流程都梳理的极为清晰

同时跟产品经理一起，把系统里的核心术语都提炼出来，不要有歧义，达成共识

举个例子吧，比如说针对“物流系统的发货单“这个概念，你要用什么术语来表达？可能你自己觉得是发货单，但是产品经理认为如果考虑到公司高层、运营、仓储等各个部门的人员平时熟悉的概念，更好的概念应该是发货出库单

所以说，营销系统也是一样的

首先，你必须找你营销系统的PM聊聊，那个PM他既然负责营销系统，就会去调研竞对产品的营销方式 有哪些，业内常见的营销方式有哪些，然后公司内部打算用哪些营销方式，每一种营销方式具体的使用规则是什么，他会跟公司内部的运营推广的部门的用户聊，看他们对营销方式的理解是什么

所以PM应该是这块的一个业务专家的角色，他是绝对熟悉和了解业务的

所以你必须跟他聊聊，确定下来你的营销系统内有哪些专业术语，然后你们统一口径，确定这些专业术语的叫法

这就是你进行DDD实体建模的第一步，跟PM沟通，提炼出你系统的一些核心术语，对术语的名称和定义都有是清晰无歧义，多方达成共识的（包括PM、RD、外部用户、内部用户、其他部门、公司高层）

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/061~080资料/065\_完善抽奖系统的DDD建模案例的代码实现（5）/笔记.docx**

《065\_完善抽奖系统的DDD建模案例的代码实现（5）》

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/061~080资料/080\_确定促销中心的核心术语属于哪种领域对象？（2）/笔记.docx**

《080\_确定促销中心的核心术语属于哪种领域对象？（2）》

促销活动：促销中心的核心就是管理一个一个的促销活动，让运营部门去创建不同的促销活动，配合上一些促销活动的站内 or 站外的营销推广，进而拉动老用户的活跃度和复购率，或者是在站外进行拉新，每一种促销活动都包含了几个维度，包括了时间范围、人群范围、商品范围，等等

促销类型：这个其实我们之前就有过这个概念，促销是有很多种不同的类型的，我们认为完全是可以全部集成到促销这个概念里去的，那当然每一种促销活动都有一种促销类型，比如满减、买赠、直降、优惠券、秒杀，等等

促销规则：顾名思义，根据促销类型定义出来的具体促销规则，这个很灵活多变，之前我们都是考虑用JSON存储的

促销范围：每一个促销活动都会匹配一个促销范围，这个促销范围实际上是通过时间范围、人群范围、商品范围、平台范围（APP or 网站），等各种范围来确定的

候选促销活动查询：大家如果平时玩儿电商APP的话就知道，其实一般来说，促销活动会通过如下方式展现给你，要不然是在电商首页搞一些促销活动的专题页，要不然就是你浏览某个商品的时候，商品详情页里会提示促销活动，要不然就是你把商品加入购物车之后，在购物车浏览界面里可以看到每个商品参加的促销活动，当你提交订单之后，在订单确认页里，也会提示你每个商品参加的促销活动以及每个商品最终的价格，所以在电商的各个环节里，都需要去根据用户所属的人群范围、当前时间、浏览的商品范围，到促销中心查询对应的促销活动列表

优惠结果计算：这个一般是用于订单确认页面的，当你选购了一些商品进入订单确认页面时，如果大家平时玩儿淘宝可能会有印象，就是一个商品可能会同时参与多种促销活动，此时你可以在订单确认页里选择这个商品你最终要参加哪个促销活动，然后根据你选择的促销活动，促销中心可以提供优惠价格计算的服务，要不然是赠送商品给你，要不然是提示可以赠送你几个优惠券，要不然是提示你减免后的价格，甚至有时一个商品可以同时参与多个活动

促销活动优先顺序查询：上面说了，有的商品可能可以同时参加多种促销活动，那么在优惠结果计算的时候，需要按照不同促销活动的优先顺序来依次进行计算

促销活动互斥判断：当然一个商品可以同时参加多个促销活动，也可以设置让多个促销活动进行互斥，也就是说，商品要么选择这个促销活动，要么选择另外一个促销活动

其实分析清楚了以后，你会发现，促销中心的核心就是促销活动、促销类型、候选促销活动查询、优惠结果计算，然后促销活动优先顺序以及促销活动互斥都是依附于促销活动本身的一些关键属性，在优惠结果计算的时候会用到

到此为止，我们就把促销中心的核心关键术语以及定义都提炼出来了

在确定了这些关键术语之后，接下来很关键的一步，就是要确定这些关键术语分别术语什么领域对象？因为在DDD里，要不然是实体，要不然是值对象，实体是有唯一标识id的，值对象其实是没有唯一标识的，或者即使他在数据库里有唯一标识，但是这个唯一标识在业务上没有意义的

所以上面我们分析一下，明确能够发现，促销活动当然是实体了，因为不同的促销活动是有唯一标识的，这个在业务上都是有含义的，然后促销类型当然是值对象了，虽然促销类型可以在数据库里存储单独的表，也可以有数据库层面的ID主键，但是其实他在业务含义里，那个标识是没有意义的

同理的就是促销规则和促销范围，每一个促销范围在数据库里可以单独存一个表，但是其实他本质是促销活动的一个附属数据而已，即使有数据库id也是没有业务上的意义的

这里就可以确定一个聚合根了，那就是优惠活动是聚合根，里面包含了促销类型、促销规则、促销范围

然后对于候选促销活动查询、优惠结果计算、促销活动优先顺序查询、促销活动互斥判断，这些其实本质都是属于行为，并不是在实体/值对象的范围内的，如果是纯行为逻辑的一些东西，那么最好就是把他们建模为领域服务这个概念

说白了，就是把他们还是还原为类似以前的Service层那个概念

比如候选促销活动查询，可能对外还是提供一个接口出去；然后优惠结果计算也是提供一个接口出去，作为一个领域服务接口，就是营销域的服务接口，然后促销活动优先顺序查询、促销活动互斥判断可以设计为内部的领域服务Service，让优惠结果计算去调用

其实最终一个正规的DDD建模好的字域对外暴露的，应该是领域模型（实体模型、值对象模型）+ 对外的领域服务接口 + 对内的实体行为调用接口，实体模型/值对象模型是让其他子域进行实体行为调用的，然后实体引用内部封装了RPC调用，请求到你的实体行为调用接口，你内部还原为实体再调用实体的行为

对外的领域服务接口，就是你领域提供的一些行为，直接提供出去就可以了

这是我比较推荐的一种DDD之间的具体落地的不同子域的集成模式

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/061~080资料/066\_完善抽奖系统的DDD建模案例的代码实现（6）/笔记.docx**

《066\_完善抽奖系统的DDD建模案例的代码实现（6）》

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/061~080资料/078\_促销中心的核心专业术语有哪些（2）/笔记.docx**

《078\_促销中心的核心专业术语有哪些（2）》

促销活动：促销中心的核心就是管理一个一个的促销活动，让运营部门去创建不同的促销活动，配合上一些促销活动的站内 or 站外的营销推广，进而拉动老用户的活跃度和复购率，或者是在站外进行拉新，每一种促销活动都包含了几个维度，包括了时间范围、人群范围、商品范围，等等

促销类型：这个其实我们之前就有过这个概念，促销是有很多种不同的类型的，我们认为完全是可以全部集成到促销这个概念里去的，那当然每一种促销活动都有一种促销类型，比如满减、买赠、直降、优惠券、秒杀，等等

促销规则：顾名思义，根据促销类型定义出来的具体促销规则，这个很灵活多变，之前我们都是考虑用JSON存储的

促销范围：每一个促销活动都会匹配一个促销范围，这个促销范围实际上是通过时间范围、人群范围、商品范围、平台范围（APP or 网站），等各种范围来确定的

在手机里的概念叫做通知，如果你给了一个APP通知的权限，这个APP是可以自动在手机顶部弹出一个通知来的；推送短信；推送邮件；那么就要在外部的渠道推广，自媒体、微博、微信公众号、视频网站、各种网站、APP、社交平台、地铁、小区电梯、公交站广告牌，用促销活动吸引新用户过来看

候选促销活动查询：大家如果平时玩儿电商APP的话就知道，其实一般来说，促销活动会通过如下方式展现给你，要不然是在电商首页搞一些促销活动的专题页，要不然就是你浏览某个商品的时候，商品详情页里会提示促销活动，要不然就是你把商品加入购物车之后，在购物车浏览界面里可以看到每个商品参加的促销活动，当你提交订单之后，在订单确认页里，也会提示你每个商品参加的促销活动以及每个商品最终的价格，所以在电商的各个环节里，都需要去根据用户所属的人群范围、当前时间、浏览的商品范围，到促销中心查询对应的促销活动列表

优惠结果计算：这个一般是用于订单确认页面的，当你选购了一些商品进入订单确认页面时，如果大家平时玩儿淘宝可能会有印象，就是一个商品可能会同时参与多种促销活动，此时你可以在订单确认页里选择这个商品你最终要参加哪个促销活动，然后根据你选择的促销活动，促销中心可以提供优惠价格计算的服务，要不然是赠送商品给你，要不然是提示可以赠送你几个优惠券，要不然是提示你减免后的价格，甚至有时一个商品可以同时参与多个活动

促销活动优先顺序查询：上面说了，有的商品可能可以同时参加多种促销活动，那么在优惠结果计算的时候，需要按照不同促销活动的优先顺序来依次进行计算

促销活动互斥判断：当然一个商品可以同时参加多个促销活动，也可以设置让多个促销活动进行互斥，也就是说，商品要么选择这个促销活动，要么选择另外一个促销活动

其实分析清楚了以后，你会发现，促销中心的核心就是促销活动、促销类型、候选促销活动查询、优惠结果计算，然后促销活动优先顺序以及促销活动互斥都是依附于促销活动本身的一些关键属性，在优惠结果计算的时候会用到

到此为止，我们就把促销中心的核心关键术语以及定义都提炼出来了

在确定了这些关键术语之后，接下来很关键的一步，就是要确定这些关键术语分别术语什么领域对象？因为在DDD里，要不然是实体，要不然是值对象，实体是有唯一标识id的，值对象其实是没有唯一标识的，或者即使他在数据库里有唯一标识，但是这个唯一标识在业务上没有意义的

所以上面我们分析一下，明确能够发现，促销活动当然是实体了，因为不同的促销活动是有唯一标识的，这个在业务上都是有含义的，然后促销类型当然是值对象了，虽然促销类型可以在数据库里存储单独的表，也可以有数据库层面的ID主键，但是其实他在业务含义里，那个标识是没有意义的

同理的就是促销规则和促销范围，每一个促销范围在数据库里可以单独存一个表，但是其实他本质是促销活动的一个附属数据而已，即使有数据库id也是没有业务上的意义的

这里就可以确定一个聚合根了，那就是优惠活动是聚合根，里面包含了促销类型、促销规则、促销范围

然后对于候选促销活动查询、优惠结果计算、促销活动优先顺序查询、促销活动互斥判断，这些其实本质都是属于行为，并不是在实体/值对象的范围内的，如果是纯行为逻辑的一些东西，那么最好就是把他们建模为领域服务这个概念

说白了，就是把他们还是还原为类似以前的Service层那个概念

比如候选促销活动查询，可能对外还是提供一个接口出去；然后优惠结果计算也是提供一个接口出去，作为一个领域服务接口，就是营销域的服务接口，然后促销活动优先顺序查询、促销活动互斥判断可以设计为内部的领域服务Service，让优惠结果计算去调用

其实最终一个正规的DDD建模好的字域对外暴露的，应该是领域模型（实体模型、值对象模型）+ 对外的领域服务接口 + 对内的实体行为调用接口，实体模型/值对象模型是让其他子域进行实体行为调用的，然后实体引用内部封装了RPC调用，请求到你的实体行为调用接口，你内部还原为实体再调用实体的行为

对外的领域服务接口，就是你领域提供的一些行为，直接提供出去就可以了

这是我比较推荐的一种DDD之间的具体落地的不同子域的集成模式

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/061~080资料/071\_电商系统促销中心的DDD设计案例/笔记.docx**

3616118202

《071\_电商系统促销中心的DDD设计案例》

项目阶段一里是有过营销中心服务开发的，其实一般电商的营销模块说白了就是可以设置各种营销活动，什么满减、秒杀、买赠、优惠券之类的，你都可以配置，有的活动是针对全场的，有的是针对某种商品的，然后还应该会提供对应的营销活动下的一些计算工作，比如计算价格的减免，或者是商品的赠予计算，或者是购买商品后发券，等等

回顾一下阶段写的营销中心的一些代码逻辑，当时是做的比较简化的

很糟糕，没有任何业务建模在里面，全是传统的CRUD，甚至数据模型用了JSON来替代，导致代码的可读性和可维护性特别差，这样的代码，在初期快速开发和上线是没问题的，但是后续持续迭代，人越来越多，这种代码几乎是无法维护的

我们认为是不需要了，因为其实就认为这个营销中间自己本身就是一个子域就可以了，就是一个营销域，营销域会跟其他的子域进行上下文映射和集成，这个我们其实都不用过多的考虑，重点关注的是营销域内部的战术建模就可以了

这里给大家讲一个技巧，比如你是个P6/P7，负责了一个营销子系统，带了几个人，也可能就你一个人，此时你可以选择按我们之前说的，营销系统内部再拆分N个服务，这样你就可以对营销系统做个战略建模了，内部划分子域，然后搞子域的映射和集成

但是也可以默认为是你营销子域就是一个独立子域，内部不划分子域了，你就内部建立一套通用语言模型，进行统一战术建模就可以了，其实战术建模说的太夸张了，说白了就是内部进行实体建模

至于你自己玩了自己系统的实体建模，内部全部按DDD来建模了一遍，还原真实/虚拟世界的业务流程之后，你如何跟其他系统集成呢？其实这里要讨巧一点，有可能其他系统根本就不是DDD建模的，也不想DDD建模，他们还是就希望你传统的提供一些接口就可以了

那此时你就可以还是提供一些接口出去，但是每个接口的业务逻辑，并不是跟以前一样，还调用service+dao之类的，而是还原出来实体，然后调用实体的行为，让内部多个实体串联起来跑，还原真实/虚拟世界的业务流程，让代码看着特别漂亮

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/061~080资料/069\_完善抽奖系统的DDD建模案例的代码实现（9）/笔记.docx**

3616118202

《069\_完善抽奖系统的DDD建模案例的代码实现（9）》

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/061~080资料/070\_完善抽奖系统的DDD建模案例的代码实现（10）/笔记.docx**

《070\_完善抽奖系统的DDD建模案例的代码实现（10）》

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/061~080资料/063\_完善抽奖系统的DDD建模案例的代码实现（3）/笔记.docx**

3616118202

《063\_完善抽奖系统的DDD建模案例的代码实现（3）》

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/061~080资料/064\_完善抽奖系统的DDD建模案例的代码实现（4）/笔记.docx**

3616118202

《064\_完善抽奖系统的DDD建模案例的代码实现（4）》

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/061~080资料/076\_促销中心DDD重构的起点：找产品聊聊天（3）/笔记.docx**

3616118202

《076\_促销中心DDD重构的起点：找产品聊聊天（3）》

如果你想要对自己负责的营销中心进行DDD建模重构，那么就是跳过战略建模部分，也就是系统边界划分以及集成关系确定这些都跳过去了，然后直接进行内部的实体建模，那么首先就是得对你的系统里的核心关键术语，也就是所谓的通用语言吧，做一个提取

那么提取你负责系统里的核心术语，怎么提取呢？当然不能是你技术RD拍板了，你定的术语可能别人都不能理解，一般一个技术产品的负责人，其实不是RD，站在公司老板和高层的角度，他看到的是一个一个的系统产品，而不是系统背后的技术架构和技术逻辑

所以任何一个公司决定一个技术类产品的，并不是技术RD，而是产品经理

这个产品经理能沟通技术产品的使用方，外部C端/B端用户，or内部的公司其他人，比如运营之类的，还能沟通技术这边的RD小哥哥，还能把产品讲给高层领导听，所以产品经理才是任何一个技术类产品的核心人物

他们了解产品的使用方需求，会决定产品的形态，推动产品的 研发，管理产品的研发进度和项目进展，担负起产品的交付和上线，关注产品线上运行的情况，是否有bug，了解竞对产品情况，分析产品优劣势，等等

当然我并不是说国内每一个产品经理都这么牛，我只是说产品经理这个角色在一个技术类产品里的定位，无论是一个APP、网站还是一个系统，他的定位其实是极为重要的，技术负责人本质是对产品经理进行follow跟随的

因为技术负责人的任务，就是想尽办法把产品做出来，产品各项性能指标都很好，不出bug，研发速度要快，能管理技术团队，产品有问题随时解决，产品使用量大了，可以设计好的技术架构，或者进行技术升级

当然国内经常说RD和PM的矛盾很深，那是因为国内的PM大多都达不到我说的要求，28法则吧，大部分的产品经理的能力都是不专业的，能力平庸，还喜欢瞎搞，所以搞的RD小哥哥经常很愤怒

但是不管怎么说吧，哪怕你合作的PM能力有点渣，你也得找他聊，因为谁让他那么重要呢？所以这里关键第一步，你必须找他仔细聊聊，把你负责系统的各项功能以及业务流程都梳理的极为清晰

同时跟产品经理一起，把系统里的核心术语都提炼出来，不要有歧义，达成共识

举个例子吧，比如说针对“物流系统的发货单“这个概念，你要用什么术语来表达？可能你自己觉得是发货单，但是产品经理认为如果考虑到公司高层、运营、仓储等各个部门的人员平时熟悉的概念，更好的概念应该是发货出库单

所以说，营销系统也是一样的

首先，你必须找你营销系统的PM聊聊，那个PM他既然负责营销系统，就会去调研竞对产品的营销方式 有哪些，业内常见的营销方式有哪些，然后公司内部打算用哪些营销方式，每一种营销方式具体的使用规则是什么，他会跟公司内部的运营推广的部门的用户聊，看他们对营销方式的理解是什么

所以PM应该是这块的一个业务专家的角色，他是绝对熟悉和了解业务的

所以你必须跟他聊聊，确定下来你的营销系统内有哪些专业术语，然后你们统一口径，确定这些专业术语的叫法

这就是你进行DDD实体建模的第一步，跟PM沟通，提炼出你系统的一些核心术语，对术语的名称和定义都有是清晰无歧义，多方达成共识的（包括PM、RD、外部用户、内部用户、其他部门、公司高层）

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/061~080资料/077\_促销中心的核心专业术语有哪些（1）/笔记.docx**

《077\_促销中心的核心专业术语有哪些（1）》

以下的专业术语，在我这里是直接告诉你了，对你来说，如果着手要做一个系统的DDD建模，建议你先跟PM深入透彻的聊一番，一起抽象业务功能、业务流程、业务术语，才能达到下面一番术语

促销活动：促销中心的核心就是管理一个一个的促销活动，让运营部门去创建不同的促销活动，配合上一些促销活动的站内 or 站外的营销推广，进而拉动老用户的活跃度和复购率，或者是在站外进行拉新，每一种促销活动都包含了几个维度，包括了时间范围、人群范围、商品范围，等等

促销类型：这个其实我们之前就有过这个概念，促销是有很多种不同的类型的，我们认为完全是可以全部集成到促销这个概念里去的，那当然每一种促销活动都有一种促销类型，比如满减、折扣、买赠、直降、优惠券、秒杀，等等

促销规则：顾名思义，根据促销类型定义出来的具体促销规则，这个很灵活多变，之前我们都是考虑用JSON存储的

促销范围：每一个促销活动都会匹配一个促销范围，这个促销范围实际上是通过时间范围、人群范围、商品范围、平台范围（APP or 网站），等各种范围来确定的

候选促销活动查询：大家如果平时玩儿电商APP的话就知道，其实一般来说，促销活动会通过如下方式展现给你，要不然是在电商首页搞一些促销活动的专题页，要不然就是你浏览某个商品的时候，商品详情页里会提示促销活动，要不然就是你把商品加入购物车之后，在购物车浏览界面里可以看到每个商品参加的促销活动，当你提交订单之后，在订单确认页里，也会提示你每个商品参加的促销活动以及每个商品最终的价格，所以在电商的各个环节里，都需要去根据用户所属的人群范围、当前时间、浏览的商品范围，到促销中心查询对应的促销活动列表

优惠结果计算：这个一般是用于订单确认页面的，当你选购了一些商品进入订单确认页面时，如果大家平时玩儿淘宝可能会有印象，就是一个商品可能会同时参与多种促销活动，此时你可以在订单确认页里选择这个商品你最终要参加哪个促销活动，然后根据你选择的促销活动，促销中心可以提供优惠价格计算的服务，要不然是赠送商品给你，要不然是提示可以赠送你几个优惠券，要不然是提示你减免后的价格，甚至有时一个商品可以同时参与多个活动

促销活动优先顺序查询：上面说了，有的商品可能可以同时参加多种促销活动，那么在优惠结果计算的时候，需要按照不同促销活动的优先顺序来依次进行计算

促销活动互斥判断：当然一个商品可以同时参加多个促销活动，也可以设置让多个促销活动进行互斥，也就是说，商品要么选择这个促销活动，要么选择另外一个促销活动

其实分析清楚了以后，你会发现，促销中心的核心就是促销活动、促销类型、候选促销活动查询、优惠结果计算，然后促销活动优先顺序以及促销活动互斥都是依附于促销活动本身的一些关键属性，在优惠结果计算的时候会用到

到此为止，我们就把促销中心的核心关键术语以及定义都提炼出来了

在确定了这些关键术语之后，接下来很关键的一步，就是要确定这些关键术语分别术语什么领域对象？因为在DDD里，要不然是实体，要不然是值对象，实体是有唯一标识id的，值对象其实是没有唯一标识的，或者即使他在数据库里有唯一标识，但是这个唯一标识在业务上没有意义的

所以上面我们分析一下，明确能够发现，促销活动当然是实体了，因为不同的促销活动是有唯一标识的，这个在业务上都是有含义的，然后促销类型当然是值对象了，虽然促销类型可以在数据库里存储单独的表，也可以有数据库层面的ID主键，但是其实他在业务含义里，那个标识是没有意义的

同理的就是促销规则和促销范围，每一个促销范围在数据库里可以单独存一个表，但是其实他本质是促销活动的一个附属数据而已，即使有数据库id也是没有业务上的意义的

这里就可以确定一个聚合根了，那就是优惠活动是聚合根，里面包含了促销类型、促销规则、促销范围

然后至于实体的行为，那肯定促销活动实体内部得有增删改查的行为；同时还得提供候选促销活动匹配判断行为、促销活动顺序查询、促销活动互斥查询等行为，说白了，你需要的促销活动的行为，都得集成到里面去

然后对于候选促销活动查询、优惠结果计算、促销活动优先顺序查询、促销活动互斥判断，这些其实本质都是属于行为，并不是在实体/值对象的范围内的，如果是纯行为逻辑的一些东西，那么最好就是把他们建模为领域服务这个概念

说白了，就是把他们还是还原为类似以前的Service层那个概念

比如候选促销活动查询，可能对外还是提供一个接口出去；然后优惠结果计算也是提供一个接口出去，作为一个领域服务接口，就是营销域的服务接口，然后促销活动优先顺序查询、促销活动互斥判断可以设计为内部的领域服务Service，让优惠结果计算去调用

其实最终一个正规的DDD建模好的字域对外暴露的，应该是领域模型（实体模型、值对象模型）+ 对外的领域服务接口 + 对内的实体行为调用接口，实体模型/值对象模型是让其他子域进行实体行为调用的，然后实体引用内部封装了RPC调用，请求到你的实体行为调用接口，你内部还原为实体再调用实体的行为

对外的领域服务接口，就是你领域提供的一些行为，直接提供出去就可以了

这是我比较推荐的一种DDD之间的具体落地的不同子域的集成模式

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/061~080资料/067\_完善抽奖系统的DDD建模案例的代码实现（7）/笔记.docx**

《067\_完善抽奖系统的DDD建模案例的代码实现（7）》

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/041~060资料/050\_用抽奖系统案例来实战枯燥的DDD上下文映射（4）/笔记.docx**

《050\_用抽奖系统案例来实战枯燥的DDD上下文映射（4）》

抽奖行为子域是D，抽奖活动、奖品库存、抽奖风控三个子域，都是U，营销中心、订单中心、会员中心三个字域，都是U

然后抽奖行为子域和抽奖活动、奖品库存、抽奖风控三个字域之间，全部是PS关系，也就是合作强依赖关系，因为大家的交互都是为了实现特定的同一个抽奖项目和需求而做的，所以就是强依赖关系

然后营销中心、订单中心、会员中心，如果是小型公司里，那么其实直接就是开放自己的一些接口，你抽奖服务直接调用就可以了，人家才不会care你，设计接口还来跟你商量商量呢，你也没法提需求，所以如果小公司里，就是尊奉关系

但是中型以上的公司里，一般各个中心都有中型规模的团队，可能都是一个P8带领每个方向的，所以一般中心对外，都是开放RESTful API的，当然，也有可能是各个中心专门弄一个服务，就是中心的开放API服务，也可以用RPC

不管是什么吧，反正这种中大型公司的中心，对外开放，无论RESTful API还是RPC，都是一个开放主机服务，可能用不同的发布语言，所以他们应该是一个OHS/PL，然后对于抽奖子域自己，一般需要做一个防腐层，ACL，对各个中心进行防腐，把别人的实体转换为自己定义的稳定实体

到此为止，上下文映射搞定，其实映射搞定之后，大致的集成方式你也知道了吧？基本上都是RPC，或者有肯能调用各个中心会是RESTful API，看你们公司具体怎么做的

现在最后一步，子域内部的细化建模，大家只要记住一点，尽可能的还原真实世界/虚拟世界的里的业务实体和业务流程，按照这个思路去建模

抽奖活动子域中，有一个LotteryDrawActivity，就是抽奖活动，lottery draw，是名词，抽奖，draw a lottery是动词，代表一个动作，就是抽一次奖的意思，所以抽奖活动可以作为一个聚合根，进行一下建模，包含了奖品池子，prize

Prize就得是一个建模，他是对应了一个奖品，可以有奖品类型，其实可以对不同的奖品类型做一个实现类的定义

然后就是对抽奖活动的各种配置，包括奖品的配置，都可以写一下各种字段

然后是奖品库存子域，里面就是一个独立的PrizeStock的实体，包含了实物商品的库存

抽奖子域中，得有一个DrawLottery，代表了一次抽奖行为

这个DrawLottery，是可以关联上一些实体的，核心就在于，你一次抽奖，是针对哪些东西来抽的，比如说抽奖活动id，这个就是实体之间的关联，另外得有一个抽奖用户的概念，简单的话，就是建模成一个抽奖用户的id就可以了

然后他发起抽奖之后，先要进行风控，接着会执行一系列的逻辑，这个逻辑，是不是可以封装在一个抽奖算法里？实际上，抽奖算法，在这里应该建模成一个独立的实体，那么抽奖行为调用抽奖算法，就可以执行一次抽奖

抽奖完毕之后，就有一个抽奖结果，也需要进行建模，还可以用于返回给用户

然后就是发奖，发奖就是调用各个中心就可以了

按照上述思路，把各个实体/值对象以及聚合关系，都设计好，然后行为方法都设计好，这就搞定了

下周来做工程实现，基于spring cloud alibaba来进行工程划分，建立实体，然后设计DAO，把DAO装载到领域实体里去，然后是接口层，接口就是负责根据请求找到对应的实体，基于实体去执行业务逻辑

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/041~060资料/054\_基于Spring Cloud Alibaba为抽奖系统搭建工程（4）/笔记.docx**

3616118202

《054\_基于Spring Cloud Alibaba为抽奖系统搭建工程（4）》

在一个分布式系统里，你在进行DDD建模的时候，最好是直接梳理清楚上下游的关系，尽量是反对说，服务A调用服务B，服务B也反过来去调用A，尽量不要这样子，有一个明确的上下游的关系

子域之间的映射关系，集成关系

实体建模，类建模，聚合关系，id标识符，先在文档里建模，我们这里为了省事儿，直接在代码层面建模了，实体在代码层面建出来了

微服务技术架构，Spring Cloud Alibaba，儒猿技术窝的C2C微服务专栏

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/041~060资料/053\_基于Spring Cloud Alibaba为抽奖系统搭建工程（3）/笔记.docx**

《053\_基于Spring Cloud Alibaba为抽奖系统搭建工程（3）》

在一个分布式系统里，你在进行DDD建模的时候，最好是直接梳理清楚上下游的关系，尽量是反对说，服务A调用服务B，服务B也反过来去调用A，尽量不要这样子，有一个明确的上下游的关系

子域之间的映射关系，集成关系

实体建模，类建模，聚合关系，id标识符，先在文档里建模，我们这里为了省事儿，直接在代码层面建模了，实体在代码层面建出来了

微服务技术架构，Spring Cloud Alibaba，儒猿技术窝的C2C微服务专栏

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/041~060资料/051\_基于Spring Cloud Alibaba为抽奖系统搭建工程（1）/笔记.docx**

3616118202

《051\_基于Spring Cloud Alibaba为抽奖系统搭建工程（1）》

在一个分布式系统里，你在进行DDD建模的时候，最好是直接梳理清楚上下游的关系，尽量是反对说，服务A调用服务B，服务B也反过来去调用A，尽量不要这样子，有一个明确的上下游的关系

子域之间的映射关系，集成关系

实体建模，类建模，聚合关系，id标识符，先在文档里建模，我们这里为了省事儿，直接在代码层面建模了，实体在代码层面建出来了

微服务技术架构，Spring Cloud Alibaba，儒猿技术窝的C2C微服务专栏

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/041~060资料/048\_用抽奖系统案例来实战枯燥的DDD上下文映射（2）/笔记.docx**

3616118202

《048\_用抽奖系统案例来实战枯燥的DDD上下文映射（2）》

回顾一下上下文映射的关系，说白了，他其实就是定义一下不同子域/子系统之间的上下游关系、集成关系，也决定了他们之间的集成方式（RPC、RESTful API、MQ）

U和D，一般U是上游，upstream，就是服务提供者，D是下游，就是downstream，就是服务调用者，所以谁调用谁就是D，谁提供谁就是U，表现成线条，就是U流向D，因为D在调用，所以数据是从U流向D的，是这么个意思

合作关系（Partnership）：两个上下文强耦合在一起，基本上就是比如做一个需求吧，多人一起做，大家都是为了一个需求设计自己服务的接口，基本上服务之间以及接口之间都是特定耦合，强行耦合的，为了完成一个完整的需求，难以分开，这种一般是紧密协作的团队内部或者多个团队，都是做一个项目和需求，常常是这种模式

共享内核（Shared Kernel）：两个上下文中间一起商量定义一个通用的领域模型，大家一起用，维护一个独立的通用工程，一般团队内部的不同子域会用这种方式，不同人负责不同的服务，但是抽象一些通用模型出来

客户方-供应方开发（Customer-Supplier Development）：上下文之间互相有商有量的进行通用语言的设计，比如接口提供方设计接口和模型，会问问调用方，调用方也会提自己的需求给提供方，这种的话，一般都是跟合作关系一起用的，有时候大家会这样商量协作，有组织的协作，当你需要做某个事儿的时候，需要团队内部或者其他团队协作，去调用人家的接口，人家可能没有，你可以提个需求，大家商量一下

遵从者（Conformist）：设计通用语言的时候，谁也别商量，调用的时候互相了解一下，直接调用，这种情况我觉得是很多的，就是有个服务按自己的想法暴露一些接口出来，你需要的时候直接去调用就行了，别多啰嗦，团队内部或者外部都有可能的，就是人家不会配合你的，你就调用

开放主机服务（Open Host Service）和发布语言（Published Language）：这一般是大公司内部，有那种中心之间的调用交互，暴露开放接口，定义一种协议/语言，一般都是RESTful API，或者是第三方公司开放给你的接口

防腐层（Anticorruption Layer）：对其他的服务进行接口调用的时候，要把接口返回给你的东西做一下转换，转换为你内部的实体，避免人家接口改变，你这里就变，这是防腐的一个层

大泥球（Big Ball of Mud）：搞的一塌糊涂，本来应该拆到多个子域里的东西，混杂到了一个子域里去

独立路线（SeparateWay）：没关系

抽奖行为子域是D，抽奖活动、奖品库存、抽奖风控三个子域，都是U，营销中心、订单中心、会员中心三个字域，都是U

然后抽奖行为子域和抽奖活动、奖品库存、抽奖风控三个字域之间，全部是PS关系，也就是合作强依赖关系，因为大家的交互都是为了实现特定的同一个抽奖项目和需求而做的，所以就是强依赖关系

然后营销中心、订单中心、会员中心，如果是小型公司里，那么其实直接就是开放自己的一些接口，你抽奖服务直接调用就可以了，人家才不会care你，设计接口还来跟你商量商量呢，你也没法提需求，所以如果小公司里，就是尊奉关系

但是中型以上的公司里，一般各个中心都有中型规模的团队，可能都是一个P8带领每个方向的，所以一般中心对外，都是开放RESTful API的，当然，也有可能是各个中心专门弄一个服务，就是中心的开放API服务，也可以用RPC

不管是什么吧，反正这种中大型公司的中心，对外开放，无论RESTful API还是RPC，都是一个开放主机服务，可能用不同的发布语言，所以他们应该是一个OHS/PL，然后对于抽奖子域自己，一般需要做一个防腐层，ACL，对各个中心进行防腐，把别人的实体转换为自己定义的稳定实体

到此为止，上下文映射搞定，其实映射搞定之后，大致的集成方式你也知道了吧？基本上都是RPC，或者有肯能调用各个中心会是RESTful API，看你们公司具体怎么做的

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/041~060资料/045\_根据抽奖业务流程来划分各个子域（2）/笔记.docx**

《045\_根据抽奖业务流程来划分各个子域（2）》

DDD建模，三个步骤

划分子域 -> 子域的集成关系 -> 实体建模，DDD建模就结束了

工程落地，概要设计、详细设计、编码开发

这是我们的第一个DDD建模实战案例，后续还有几个案例，包括了在线教育系统部分模块的DDD建模案例、商品中心部分模块的DDD建模重构案例、营销中心部分模块的DDD建模重构案例，这些都是案例，但是学好了之后，基本上就完全明白DDD应该怎么做了，配合之前的内容，效果非常好

大家初步就有能力可以进行DDD建模实战落地了，这些案例对标的是P6级别的同学，可以对自己手头负责的系统去进行DDD建模，万一要是说你没有权限用DDD推到重构，你可以自己私下里用DDD重新建模一遍，自己私下写一遍DDD重新建模后的代码

后续还会讲三个完整的DDD实战项目：完整订单系统的DDD建模重构项目、完整权限系统的DDD建模重构项目、完整供应链系统的DDD建模重构项目，中间会穿插所有DDD高阶的建模技巧、知识以及思想

学完之后，大家完全有能力站在P7级别的角度去中大型系统进行DDD建模了，哪怕是P8级别的同学，也可以用这套方法去进行大型系统的DDD建模

运营可以配置抽奖活动，每个抽奖活动可以配置多种不同的奖品（优惠券、虚拟金币、实物商品、第三方合作商家的抵扣券，等等），每一种奖品都有一个抽中的概率，每个抽奖活动可以配置面向指定的人群（根据用户画像标签来指定），还有抽奖开始和结束时间

每个人的抽奖次数的限制，在抽奖的时候，奖品除了抽中概率之外，还有一个库存的存在，包括每个人对一个奖品最多抽中几次

用户可能会在进入APP的时候看到一个抽奖的提示弹出层，一般都是这么做的，当然也可能是当你购买完某个商品的时候，或者是在别的时机下，然后你可以点击进入抽奖，一般是给你一个独立的界面，里面有个转盘，你可能会有1次到几次抽奖机会，接着点击按钮，就可以抽奖

抽奖的时候，会根据每种奖品的概率给你进行抽奖，大部分情况下，可能就是没中奖，或者某个第三方商家的抵扣券（本质就是引导你去合作商家消费罢了）

这就是一个典型的中小型抽奖系统的业务流程了

首先，分析一下，我们这里必然得有一个抽奖活动的概念，这个抽奖活动里会配置抽奖活动本身的一大堆数据，这个大家没问题吧？抽奖活动子域，这个可以作为一个支撑子域，因为核心的抽奖业务和行为不在这个子域里，这个子域更多的是在抽奖的过程封装一些数据，同时配合进行抽奖的各种校验和限制

其次，有了抽奖活动之后，C端用户必然会点击按钮发起抽奖的请求，这是一个抽奖的动作行为，本身就代表了复杂的业务逻辑，所以我们可以在这里做一个抽奖行为核心域，他就代表了一次抽奖行为，就是说用户的抽奖请求，都应该是由他来执行的，这才是抽奖系统的核心

其次，还得有一个奖品库存子域，因为奖品库存这个概念，跟抽奖活动的配置，实际上是两个概念，应该独立区分开来的，他也是一个支撑子域，配合抽奖行为过程中的库存校验相关的业务

最后，其实一般互联网系统，都会有一个风控，这个风控，一般会用到大数据的技术，包括秒杀系统也是的，他需要使用大数据技术去实时和离线分析用户行为，甚至可能会用到机器学习的算法，识别出来一些刷单、薅羊毛的逻辑，所以还应该有一个抽奖风控子域，这也是个支撑子域

另外，还有几个支撑子域，其实是在我们的抽奖系统外部的，就是在最后你发奖的时候，根据奖品的不同，需要调用营销中心的接口去发券、或者调用订单中心的接口去下单（实物商品）、或者调用会员中心的接口去增加虚拟金币，这些也是需要画出来的

这个需要画图，画图的时候，基本就是几个圆圈，然后抽奖行为子域有线连接到其他几个子域，这就是第一步，C端抽奖系统领域内又继续划分出了几个子域，但是其实最后这些子域反应到代码里的话呢

比如说抽奖活动子域，你没发现他可以做成一个独立的抽奖活动管理服务吗？这个服务既可以让M端去配置，也可以给C端去提供查询和校验的服务，当然给C端查询的时候，涉及到高并发，可以用redis

然后奖品库存子域可以作为一个独立的奖品库存服务，他需要负责跟公司的核心电商系统进行交互，有点类似于秒杀类的秒杀库存服务，当你设置好奖品库存的时候，他得负责跟电商库存系统进行交互，进行库存冻结，然后提前加载库存到自己的redis里去，跟秒杀可以说是如出一辙

然后抽奖行为子域，其实说白了他就是C端抽奖服务，里面封装了C端的抽奖核心业务逻辑，每一次抽奖请求，都可以转化为一个抽奖行为对象，然后执行一次抽奖行为，里面可以跟其他子域进行交互

最后是抽奖风控子域，这不就可以作为独立的抽奖风控服务么？他可以包含一些离线和实时的大数据作业，甚至是机器学习算法，如果公司有通用风控平台，也可以接入，然后他提供风控校验的功能

然后你发奖的时候，就会调用外部的营销中心、订单中心、会员中心开放出来的一些API接口去进行发奖

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/041~060资料/041\_抽奖系统DDD建模案例的背景介绍/笔记.docx**

3616118202

《041\_抽奖系统DDD建模案例的背景介绍》

运营可以配置抽奖活动，每个抽奖活动可以配置多种不同的奖品（优惠券、虚拟金币、实物商品、第三方合作商家的抵扣券，等等），每一种奖品都有一个抽中的概率，每个抽奖活动可以配置面向指定的人群（根据用户画像标签来指定），还有抽奖开始和结束时间

每个人的抽奖次数的限制，在抽奖的时候，奖品除了抽中概率之外，还有一个库存的存在，包括每个人对一个奖品最多抽中几次

用户可能会在进入APP的时候看到一个抽奖的提示弹出层，一般都是这么做的，当然也可能是当你购买完某个商品的时候，或者是在别的时机下，然后你可以点击进入抽奖，一般是给你一个独立的界面，里面有个转盘，你可能会有1次到几次抽奖机会，接着点击按钮，就可以抽奖

抽奖的时候，会根据每种奖品的概率给你进行抽奖，大部分情况下，可能就是没中奖，或者某个第三方商家的抵扣券（本质就是引导你去合作商家消费罢了）

这就是一个典型的中小型抽奖系统的业务流程了

之前给大家讲过，DDD这个事情，其实本质上应该是P8级别的人牵头来搞，一个业务线或者小业务线，站在一个整体的角度，划分为N多个限界上下文，也就是多个子域，有核心域，有支撑子域，有通用子域

然后大团队里肯定是有多个小团队的，每个小团队就可以分到一份子域，每个小团队对子域再进一步划分，每个P6都可以带一两个人，或者独立一个人，分到一两个子域，然后P6就可以对这个子域内部进行通用语言的建模了

或者就是P7牵头来搞，对5~10人团队负责的一个中型系统，进行子域划分，基本就是拆分为多个子域，每个人负责一两个子域，每个子域就是一个微服务，大概就这么个节奏，每个人再进行子域内部的建模

但是现在这个抽奖系统，一看就是一个非核心辅助类系统，基本就是一个P6的职责范围，除非是那种大电商平台，把抽奖作为一个较为核心的功能来发展的，才可能是P7带领一个团队来做一个复杂的抽奖系统

否则一般的公司，抽奖系统就是一个P6的事儿，也比较简单

所以这个案例，很明显就是教你，假设你就是个P6，那么你怎么对自己负责的一个中小型系统全流程来实践一把DDD建模

针对这个业务流程，咱们首先得把这个系统划分成两个明显需要区分开的业务领域，一个是面向C端的抽奖系统领域，一个是面向M端的抽奖管理平台领域，这就是在做DDD的时候，你应该做的第一件事情

就是当你负责的一个系统要进行DDD建模的时候，首先你自己脑子里得先按照最明显的业务角度去进行一个拆分和划分，像抽奖系统，首先你先大略的做一个C端和M端的拆分，那是必然的，因为后续在实际落地的时候，类似这么一个抽奖系统，往往就是拆分为两个服务/子系统，一个是面向M端的抽奖管理系统，一个是面向C端的抽奖系统，如果业务逻辑没有特别复杂，很难说会继续拆分成更多服务了

这就是DDD建模的第一步，一定是划分限界上下文，你先用屁股决定脑袋，一定是有想法是的，会想到大致该如何先划分一下

这是我们的第一个DDD建模实战案例，后续还有几个案例，包括了在线教育系统部分模块的DDD建模案例、商品中心部分模块的DDD建模重构案例、营销中心部分模块的DDD建模重构案例，这些都是案例，但是学好了之后，基本上就完全明白DDD应该怎么做了，配合之前的内容，效果非常好

大家初步就有能力可以进行DDD建模实战落地了，这些案例对标的是P6级别的同学，可以对自己手头负责的系统去进行DDD建模

后续还会讲三个完整的DDD实战项目：完整订单系统的DDD建模重构项目、完整权限系统的DDD建模重构项目、完整供应链系统的DDD建模重构项目，中间会穿插所有DDD高阶的建模技巧、知识以及思想

学完之后，大家完全有能力站在P7级别的角度去中大型系统进行DDD建模了，哪怕是P8级别的同学，也可以用这套方法去进行大型系统的DDD建模

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/041~060资料/052\_基于Spring Cloud Alibaba为抽奖系统搭建工程（2）/笔记.docx**

《052\_基于Spring Cloud Alibaba为抽奖系统搭建工程（2）》

在一个分布式系统里，你在进行DDD建模的时候，最好是直接梳理清楚上下游的关系，尽量是反对说，服务A调用服务B，服务B也反过来去调用A，尽量不要这样子，有一个明确的上下游的关系

子域之间的映射关系，集成关系

实体建模，类建模，聚合关系，id标识符，先在文档里建模，我们这里为了省事儿，直接在代码层面建模了，实体在代码层面建出来了

微服务技术架构，Spring Cloud Alibaba，儒猿技术窝的C2C微服务专栏

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/041~060资料/058\_对抽奖系统进行DDD实体建模（3）/笔记.docx**

3616118202

《058\_对抽奖系统进行DDD实体建模（3）》

在一个分布式系统里，你在进行DDD建模的时候，最好是直接梳理清楚上下游的关系，尽量是反对说，服务A调用服务B，服务B也反过来去调用A，尽量不要这样子，有一个明确的上下游的关系

子域之间的映射关系，集成关系

实体建模，类建模，聚合关系，id标识符，先在文档里建模，我们这里为了省事儿，直接在代码层面建模了，实体在代码层面建出来了

微服务技术架构，Spring Cloud Alibaba，儒猿技术窝的C2C微服务专栏

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/041~060资料/046\_根据抽奖业务流程来划分各个子域（3）/笔记.docx**

3616118202

《046\_根据抽奖业务流程来划分各个子域（3）》

DDD建模，三个步骤

划分子域 -> 子域的集成关系 -> 实体建模，DDD建模就结束了

工程落地，概要设计、详细设计、编码开发

这是我们的第一个DDD建模实战案例，后续还有几个案例，包括了在线教育系统部分模块的DDD建模案例、商品中心部分模块的DDD建模重构案例、营销中心部分模块的DDD建模重构案例，这些都是案例，但是学好了之后，基本上就完全明白DDD应该怎么做了，配合之前的内容，效果非常好

大家初步就有能力可以进行DDD建模实战落地了，这些案例对标的是P6级别的同学，可以对自己手头负责的系统去进行DDD建模，万一要是说你没有权限用DDD推到重构，你可以自己私下里用DDD重新建模一遍，自己私下写一遍DDD重新建模后的代码

后续还会讲三个完整的DDD实战项目：完整订单系统的DDD建模重构项目、完整权限系统的DDD建模重构项目、完整供应链系统的DDD建模重构项目，中间会穿插所有DDD高阶的建模技巧、知识以及思想

学完之后，大家完全有能力站在P7级别的角度去中大型系统进行DDD建模了，哪怕是P8级别的同学，也可以用这套方法去进行大型系统的DDD建模

运营可以配置抽奖活动，每个抽奖活动可以配置多种不同的奖品（优惠券、虚拟金币、实物商品、第三方合作商家的抵扣券，等等），每一种奖品都有一个抽中的概率，每个抽奖活动可以配置面向指定的人群（根据用户画像标签来指定），还有抽奖开始和结束时间

每个人的抽奖次数的限制，在抽奖的时候，奖品除了抽中概率之外，还有一个库存的存在，包括每个人对一个奖品最多抽中几次

用户可能会在进入APP的时候看到一个抽奖的提示弹出层，一般都是这么做的，当然也可能是当你购买完某个商品的时候，或者是在别的时机下，然后你可以点击进入抽奖，一般是给你一个独立的界面，里面有个转盘，你可能会有1次到几次抽奖机会，接着点击按钮，就可以抽奖

抽奖的时候，会根据每种奖品的概率给你进行抽奖，大部分情况下，可能就是没中奖，或者某个第三方商家的抵扣券（本质就是引导你去合作商家消费罢了）

这就是一个典型的中小型抽奖系统的业务流程了

首先，分析一下，我们这里必然得有一个抽奖活动的概念，这个抽奖活动里会配置抽奖活动本身的一大堆数据，这个大家没问题吧？抽奖活动子域，这个可以作为一个支撑子域，因为核心的抽奖业务和行为不在这个子域里，这个子域更多的是在抽奖的过程封装一些数据，同时配合进行抽奖的各种校验和限制

其次，有了抽奖活动之后，C端用户必然会点击按钮发起抽奖的请求，这是一个抽奖的动作行为，本身就代表了复杂的业务逻辑，所以我们可以在这里做一个抽奖行为核心域，他就代表了一次抽奖行为，就是说用户的抽奖请求，都应该是由他来执行的，这才是抽奖系统的核心

其次，还得有一个奖品库存子域，因为奖品库存这个概念，跟抽奖活动的配置，实际上是两个概念，应该独立区分开来的，他也是一个支撑子域，配合抽奖行为过程中的库存校验相关的业务

最后，其实一般互联网系统，都会有一个风控，这个风控，一般会用到大数据的技术，包括秒杀系统也是的，他需要使用大数据技术去实时和离线分析用户行为，甚至可能会用到机器学习的算法，识别出来一些刷单、薅羊毛的逻辑，所以还应该有一个抽奖风控子域，这也是个支撑子域

另外，还有几个支撑子域，其实是在我们的抽奖系统外部的，就是在最后你发奖的时候，根据奖品的不同，需要调用营销中心的接口去发券、或者调用订单中心的接口去下单（实物商品）、或者调用会员中心的接口去增加虚拟金币，这些也是需要画出来的

这个需要画图，画图的时候，基本就是几个圆圈，然后抽奖行为子域有线连接到其他几个子域，这就是第一步，C端抽奖系统领域内又继续划分出了几个子域，但是其实最后这些子域反应到代码里的话呢

比如说抽奖活动子域，你没发现他可以做成一个独立的抽奖活动管理服务吗？这个服务既可以让M端去配置，也可以给C端去提供查询和校验的服务，当然给C端查询的时候，涉及到高并发，可以用redis

然后奖品库存子域可以作为一个独立的奖品库存服务，他需要负责跟公司的核心电商系统进行交互，有点类似于秒杀类的秒杀库存服务，当你设置好奖品库存的时候，他得负责跟电商库存系统进行交互，进行库存冻结，然后提前加载库存到自己的redis里去，跟秒杀可以说是如出一辙

然后抽奖行为子域，其实说白了他就是C端抽奖服务，里面封装了C端的抽奖核心业务逻辑，每一次抽奖请求，都可以转化为一个抽奖行为对象，然后执行一次抽奖行为，里面可以跟其他子域进行交互

最后是抽奖风控子域，这不就可以作为独立的抽奖风控服务么？他可以包含一些离线和实时的大数据作业，甚至是机器学习算法，如果公司有通用风控平台，也可以接入，然后他提供风控校验的功能

然后你发奖的时候，就会调用外部的营销中心、订单中心、会员中心开放出来的一些API接口去进行发奖

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/041~060资料/055\_基于Spring Cloud Alibaba为抽奖系统搭建工程（5）/笔记.docx**

《055\_基于Spring Cloud Alibaba为抽奖系统搭建工程（5）》

在一个分布式系统里，你在进行DDD建模的时候，最好是直接梳理清楚上下游的关系，尽量是反对说，服务A调用服务B，服务B也反过来去调用A，尽量不要这样子，有一个明确的上下游的关系

子域之间的映射关系，集成关系

实体建模，类建模，聚合关系，id标识符，先在文档里建模，我们这里为了省事儿，直接在代码层面建模了，实体在代码层面建出来了

微服务技术架构，Spring Cloud Alibaba，儒猿技术窝的C2C微服务专栏

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/041~060资料/044\_根据抽奖业务流程来划分各个子域（1）/笔记.docx**

《044\_根据抽奖业务流程来划分各个子域（1）》

DDD建模，三个步骤

划分子域 -> 子域的集成关系 -> 实体建模，DDD建模就结束了

工程落地，概要设计、详细设计、编码开发

这是我们的第一个DDD建模实战案例，后续还有几个案例，包括了在线教育系统部分模块的DDD建模案例、商品中心部分模块的DDD建模重构案例、营销中心部分模块的DDD建模重构案例，这些都是案例，但是学好了之后，基本上就完全明白DDD应该怎么做了，配合之前的内容，效果非常好

大家初步就有能力可以进行DDD建模实战落地了，这些案例对标的是P6级别的同学，可以对自己手头负责的系统去进行DDD建模，万一要是说你没有权限用DDD推到重构，你可以自己私下里用DDD重新建模一遍，自己私下写一遍DDD重新建模后的代码

后续还会讲三个完整的DDD实战项目：完整订单系统的DDD建模重构项目、完整权限系统的DDD建模重构项目、完整供应链系统的DDD建模重构项目，中间会穿插所有DDD高阶的建模技巧、知识以及思想

学完之后，大家完全有能力站在P7级别的角度去中大型系统进行DDD建模了，哪怕是P8级别的同学，也可以用这套方法去进行大型系统的DDD建模

运营可以配置抽奖活动，每个抽奖活动可以配置多种不同的奖品（优惠券、虚拟金币、实物商品、第三方合作商家的抵扣券，等等），每一种奖品都有一个抽中的概率，每个抽奖活动可以配置面向指定的人群（根据用户画像标签来指定），还有抽奖开始和结束时间

每个人的抽奖次数的限制，在抽奖的时候，奖品除了抽中概率之外，还有一个库存的存在，包括每个人对一个奖品最多抽中几次

用户可能会在进入APP的时候看到一个抽奖的提示弹出层，一般都是这么做的，当然也可能是当你购买完某个商品的时候，或者是在别的时机下，然后你可以点击进入抽奖，一般是给你一个独立的界面，里面有个转盘，你可能会有1次到几次抽奖机会，接着点击按钮，就可以抽奖

抽奖的时候，会根据每种奖品的概率给你进行抽奖，大部分情况下，可能就是没中奖，或者某个第三方商家的抵扣券（本质就是引导你去合作商家消费罢了）

这就是一个典型的中小型抽奖系统的业务流程了

首先，分析一下，我们这里必然得有一个抽奖活动的概念，这个抽奖活动里会配置抽奖活动本身的一大堆数据，这个大家没问题吧？这个可以作为一个支撑子域，因为核心的抽奖业务和行为不在这个子域里，这个子域更多的是在抽奖的过程封装一些数据，同时配合进行抽奖的各种校验和限制

其次，有了抽奖活动之后，C端用户必然会点击按钮发起抽奖的请求，这是一个抽奖的动作行为，本身就代表了复杂的业务逻辑，所以我们可以在这里做一个抽奖行为核心域，他就代表了一次抽奖行为，就是说用户的抽奖请求，都应该是由他来执行的，这才是抽奖系统的核心

其次，还得有一个奖品库存子域，因为奖品库存这个概念，跟抽奖活动的配置，实际上是两个概念，应该独立区分开来的，他也是一个支撑子域，配合抽奖行为过程中的库存校验相关的业务

最后，其实一般互联网系统，都会有一个风控，这个风控，一般会用到大数据的技术，包括秒杀系统也是的，他需要使用大数据技术去实时和离线分析用户行为，甚至可能会用到机器学习的算法，识别出来一些刷单、薅羊毛的逻辑，所以还应该有一个抽奖风控子域，这也是个支撑子域

另外，还有几个支撑子域，其实是在我们的抽奖系统外部的，就是在最后你发奖的时候，根据奖品的不同，需要调用营销中心的接口去发券、或者调用订单中心的接口去下单（实物商品）、或者调用会员中心的接口去增加虚拟金币，这些也是需要画出来的

这个需要画图，画图的时候，基本就是几个圆圈，然后抽奖行为子域有线连接到其他几个子域，这就是第一步，C端抽奖系统领域内又继续划分出了几个子域，但是其实最后这些子域反应到代码里的话呢

比如说抽奖活动子域，你没发现他可以做成一个独立的抽奖活动管理服务吗？这个服务既可以让M端去配置，也可以给C端去提供查询和校验的服务，当然给C端查询的时候，涉及到高并发，可以用redis

然后奖品库存子域可以作为一个独立的奖品库存服务，他需要负责跟公司的核心电商系统进行交互，有点类似于秒杀类的秒杀库存服务，当你设置好奖品库存的时候，他得负责跟电商库存系统进行交互，进行库存冻结，然后提前加载库存到自己的redis里去，跟秒杀可以说是如出一辙

然后抽奖行为子域，其实说白了他就是C端抽奖服务，里面封装了C端的抽奖核心业务逻辑，每一次抽奖请求，都可以转化为一个抽奖行为对象，然后执行一次抽奖行为，里面可以跟其他子域进行交互

最后是抽奖风控子域，这不就可以作为独立的抽奖风控服务么？他可以包含一些离线和实时的大数据作业，甚至是机器学习算法，如果公司有通用风控平台，也可以接入，然后他提供风控校验的功能

然后你发奖的时候，就会调用外部的营销中心、订单中心、会员中心开放出来的一些API接口去进行发奖

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/041~060资料/043\_DDD建模第一步：用屁股决定脑袋，你一定有想法！/笔记.docx**

3616118202

《043\_DDD建模第一步：用屁股决定脑袋，你一定有想法！》

运营可以配置抽奖活动，每个抽奖活动可以配置多种不同的奖品（优惠券、虚拟金币、实物商品、第三方合作商家的抵扣券，等等），每一种奖品都有一个抽中的概率，每个抽奖活动可以配置面向指定的人群（根据用户画像标签来指定），还有抽奖开始和结束时间

每个人的抽奖次数的限制，在抽奖的时候，奖品除了抽中概率之外，还有一个库存的存在，包括每个人对一个奖品最多抽中几次

用户可能会在进入APP的时候看到一个抽奖的提示弹出层，一般都是这么做的，当然也可能是当你购买完某个商品的时候，或者是在别的时机下，然后你可以点击进入抽奖，一般是给你一个独立的界面，里面有个转盘，你可能会有1次到几次抽奖机会，接着点击按钮，就可以抽奖

抽奖的时候，会根据每种奖品的概率给你进行抽奖，大部分情况下，可能就是没中奖，或者某个第三方商家的抵扣券（本质就是引导你去合作商家消费罢了）

这就是一个典型的中小型抽奖系统的业务流程了

一个P6级别的人，带一个小弟，一起做这样的这么一个系统，甚至有可能是一个人独立维护，这个系统稳定之后，只是一个人抽一点时间维护而已

大型的商城，把抽奖作为一个核心的营销类的业务来做的，可能是一个独立的P7带的一个团队来负责这个事情

之前给大家讲过，DDD这个事情，其实本质上应该是P8级别的人牵头来搞，一个业务线或者小业务线，站在一个整体的角度，划分为N多个限界上下文，也就是多个子域，有核心域，有支撑子域，有通用子域

然后大团队里肯定是有多个小团队的，每个小团队就可以分到一份子域，每个小团队对子域再进一步划分，每个P6都可以带一两个人，或者独立一个人，分到一两个子域，然后P6就可以对这个子域内部进行通用语言的建模了

每一个子域都可以是一个微服务，有可能会是一个小系统，拆分为了几个微服务

或者就是P7牵头来搞，对5~10人团队负责的一个中型系统，进行子域划分，基本就是拆分为多个子域，每个人负责一两个子域，每个子域就是一个微服务，大概就这么个节奏，每个人再进行子域内部的建模

但是现在这个抽奖系统，一看就是一个非核心辅助类系统，基本就是一个P6的职责范围，除非是那种大电商平台，把抽奖作为一个较为核心的功能来发展的，才可能是P7带领一个团队来做一个复杂的抽奖系统

否则一般的公司，抽奖系统就是一个P6的事儿，也比较简单

所以这个案例，很明显就是教你，假设你就是个P6，那么你怎么对自己负责的一个中小型系统全流程来实践一把DDD建模

针对这个业务流程，咱们首先得把这个系统划分成两个明显需要区分开的业务领域，一个是面向C端的抽奖系统，一个是面向M端的抽奖管理平台，这就是在做DDD的时候，你应该做的第一件事情

就是当你负责的一个系统要进行DDD建模的时候，首先你自己脑子里得先按照最明显的业务角度去进行一个拆分和划分，像抽奖系统，首先你先大略的做一个C端和M端的拆分，那是必然的，因为后续在实际落地的时候，类似这么一个抽奖系统，往往就是拆分为两个服务/子系统，一个是面向M端的抽奖管理系统，一个是面向C端的抽奖系统，如果业务逻辑没有特别复杂，很难说会继续拆分成更多服务了

这就是DDD建模的第一步，一定是划分限界上下文，你先用屁股决定脑袋，一定是有想法是的，会想到大致该如何先划分一下

这是我们的第一个DDD建模实战案例，后续还有几个案例，包括了在线教育系统部分模块的DDD建模案例、商品中心部分模块的DDD建模重构案例、营销中心部分模块的DDD建模重构案例，这些都是案例，但是学好了之后，基本上就完全明白DDD应该怎么做了，配合之前的内容，效果非常好

大家初步就有能力可以进行DDD建模实战落地了，这些案例对标的是P6级别的同学，可以对自己手头负责的系统去进行DDD建模

后续还会讲三个完整的DDD实战项目：完整订单系统的DDD建模重构项目、完整权限系统的DDD建模重构项目、完整供应链系统的DDD建模重构项目，中间会穿插所有DDD高阶的建模技巧、知识以及思想

学完之后，大家完全有能力站在P7级别的角度去中大型系统进行DDD建模了，哪怕是P8级别的同学，也可以用这套方法去进行大型系统的DDD建模

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/041~060资料/049\_用抽奖系统案例来实战枯燥的DDD上下文映射（3）/笔记.docx**

《049\_用抽奖系统案例来实战枯燥的DDD上下文映射（3）》

回顾一下上下文映射的关系，说白了，他其实就是定义一下不同子域/子系统之间的上下游关系、集成关系，也决定了他们之间的集成方式（RPC、RESTful API、MQ）

U和D，一般U是上游，upstream，就是服务提供者，D是下游，就是downstream，就是服务调用者，所以谁调用谁就是D，谁提供谁就是U，表现成线条，就是U流向D，因为D在调用，所以数据是从U流向D的，是这么个意思

合作关系（Partnership）：两个上下文强耦合在一起，基本上就是比如做一个需求吧，多人一起做，大家都是为了一个需求设计自己服务的接口，基本上服务之间以及接口之间都是特定耦合，强行耦合的，为了完成一个完整的需求，难以分开，这种一般是紧密协作的团队内部或者多个团队，都是做一个项目和需求，常常是这种模式

共享内核（Shared Kernel）：两个上下文中间一起商量定义一个通用的领域模型，大家一起用，维护一个独立的通用工程，一般团队内部的不同子域会用这种方式，不同人负责不同的服务，但是抽象一些通用模型出来

客户方-供应方开发（Customer-Supplier Development）：上下文之间互相有商有量的进行通用语言的设计，比如接口提供方设计接口和模型，会问问调用方，调用方也会提自己的需求给提供方，这种的话，一般都是跟合作关系一起用的，有时候大家会这样商量协作，有组织的协作，当你需要做某个事儿的时候，需要团队内部或者其他团队协作，去调用人家的接口，人家可能没有，你可以提个需求，大家商量一下

遵从者（Conformist）：设计通用语言的时候，谁也别商量，调用的时候互相了解一下，直接调用，这种情况我觉得是很多的，就是有个服务按自己的想法暴露一些接口出来，你需要的时候直接去调用就行了，别多啰嗦，团队内部或者外部都有可能的，就是人家不会配合你的，你就调用

开放主机服务（Open Host Service）和发布语言（Published Language）：这一般是大公司内部，有那种中心之间的调用交互，暴露开放接口，定义一种协议/语言，一般都是RESTful API，或者是第三方公司开放给你的接口

防腐层（Anticorruption Layer）：对其他的服务进行接口调用的时候，要把接口返回给你的东西做一下转换，转换为你内部的实体，避免人家接口改变，你这里就变，这是防腐的一个层

大泥球（Big Ball of Mud）：搞的一塌糊涂，本来应该拆到多个子域里的东西，混杂到了一个子域里去

独立路线（SeparateWay）：没关系

抽奖行为子域是D，抽奖活动、奖品库存、抽奖风控三个子域，都是U，营销中心、订单中心、会员中心三个字域，都是U

然后抽奖行为子域和抽奖活动、奖品库存、抽奖风控三个字域之间，全部是PS关系，也就是合作强依赖关系，因为大家的交互都是为了实现特定的同一个抽奖项目和需求而做的，所以就是强依赖关系

然后营销中心、订单中心、会员中心，如果是小型公司里，那么其实直接就是开放自己的一些接口，你抽奖服务直接调用就可以了，人家才不会care你，设计接口还来跟你商量商量呢，你也没法提需求，所以如果小公司里，就是尊奉关系

但是中型以上的公司里，一般各个中心都有中型规模的团队，可能都是一个P8带领每个方向的，所以一般中心对外，都是开放RESTful API的，当然，也有可能是各个中心专门弄一个服务，就是中心的开放API服务，也可以用RPC

不管是什么吧，反正这种中大型公司的中心，对外开放，无论RESTful API还是RPC，都是一个开放主机服务，可能用不同的发布语言，所以他们应该是一个OHS/PL，然后对于抽奖子域自己，一般需要做一个防腐层，ACL，对各个中心进行防腐，把别人的实体转换为自己定义的稳定实体

到此为止，上下文映射搞定，其实映射搞定之后，大致的集成方式你也知道了吧？基本上都是RPC，或者有肯能调用各个中心会是RESTful API，看你们公司具体怎么做的

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/041~060资料/059\_对抽奖系统进行DDD实体建模（4）/笔记.docx**

3616118202

《059\_对抽奖系统进行DDD实体建模（4）》

在一个分布式系统里，你在进行DDD建模的时候，最好是直接梳理清楚上下游的关系，尽量是反对说，服务A调用服务B，服务B也反过来去调用A，尽量不要这样子，有一个明确的上下游的关系

子域之间的映射关系，集成关系

实体建模，类建模，聚合关系，id标识符，先在文档里建模，我们这里为了省事儿，直接在代码层面建模了，实体在代码层面建出来了

微服务技术架构，Spring Cloud Alibaba，儒猿技术窝的C2C微服务专栏

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/041~060资料/056\_对抽奖系统进行DDD实体建模（1）/笔记.docx**

《056\_对抽奖系统进行DDD实体建模（1）》

在一个分布式系统里，你在进行DDD建模的时候，最好是直接梳理清楚上下游的关系，尽量是反对说，服务A调用服务B，服务B也反过来去调用A，尽量不要这样子，有一个明确的上下游的关系

子域之间的映射关系，集成关系

实体建模，类建模，聚合关系，id标识符，先在文档里建模，我们这里为了省事儿，直接在代码层面建模了，实体在代码层面建出来了

微服务技术架构，Spring Cloud Alibaba，儒猿技术窝的C2C微服务专栏

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/041~060资料/042\_抽奖系统案例：一个P6进行DDD建模实战的典范/笔记.docx**

《042\_抽奖系统案例：一个P6进行DDD建模实战的典范》

运营可以配置抽奖活动，每个抽奖活动可以配置多种不同的奖品（优惠券、虚拟金币、实物商品、第三方合作商家的抵扣券，等等），每一种奖品都有一个抽中的概率，每个抽奖活动可以配置面向指定的人群（根据用户画像标签来指定），还有抽奖开始和结束时间

每个人的抽奖次数的限制，在抽奖的时候，奖品除了抽中概率之外，还有一个库存的存在，包括每个人对一个奖品最多抽中几次

用户可能会在进入APP的时候看到一个抽奖的提示弹出层，一般都是这么做的，当然也可能是当你购买完某个商品的时候，或者是在别的时机下，然后你可以点击进入抽奖，一般是给你一个独立的界面，里面有个转盘，你可能会有1次到几次抽奖机会，接着点击按钮，就可以抽奖

抽奖的时候，会根据每种奖品的概率给你进行抽奖，大部分情况下，可能就是没中奖，或者某个第三方商家的抵扣券（本质就是引导你去合作商家消费罢了）

这就是一个典型的中小型抽奖系统的业务流程了

一个P6级别的人，带一个小弟，一起做这样的这么一个系统，甚至有可能是一个人独立维护，这个系统稳定之后，只是一个人抽一点时间维护而已

大型的商城，把抽奖作为一个核心的营销类的业务来做的，可能是一个独立的P7带的一个团队来负责这个事情

之前给大家讲过，DDD这个事情，其实本质上应该是P8级别的人牵头来搞，一个业务线或者小业务线，站在一个整体的角度，划分为N多个限界上下文，也就是多个子域，有核心域，有支撑子域，有通用子域

然后大团队里肯定是有多个小团队的，每个小团队就可以分到一份子域，每个小团队对子域再进一步划分，每个P6都可以带一两个人，或者独立一个人，分到一两个子域，然后P6就可以对这个子域内部进行通用语言的建模了

每一个子域都可以是一个微服务，有可能会是一个小系统，拆分为了几个微服务

或者就是P7牵头来搞，对5~10人团队负责的一个中型系统，进行子域划分，基本就是拆分为多个子域，每个人负责一两个子域，每个子域就是一个微服务，大概就这么个节奏，每个人再进行子域内部的建模

但是现在这个抽奖系统，一看就是一个非核心辅助类系统，基本就是一个P6的职责范围，除非是那种大电商平台，把抽奖作为一个较为核心的功能来发展的，才可能是P7带领一个团队来做一个复杂的抽奖系统

否则一般的公司，抽奖系统就是一个P6的事儿，也比较简单

所以这个案例，很明显就是教你，假设你就是个P6，那么你怎么对自己负责的一个中小型系统全流程来实践一把DDD建模

针对这个业务流程，咱们首先得把这个系统划分成两个明显需要区分开的业务领域，一个是面向C端的抽奖系统领域，一个是面向M端的抽奖管理平台领域，这就是在做DDD的时候，你应该做的第一件事情

就是当你负责的一个系统要进行DDD建模的时候，首先你自己脑子里得先按照最明显的业务角度去进行一个拆分和划分，像抽奖系统，首先你先大略的做一个C端和M端的拆分，那是必然的，因为后续在实际落地的时候，类似这么一个抽奖系统，往往就是拆分为两个服务/子系统，一个是面向M端的抽奖管理系统，一个是面向C端的抽奖系统，如果业务逻辑没有特别复杂，很难说会继续拆分成更多服务了

这就是DDD建模的第一步，一定是划分限界上下文，你先用屁股决定脑袋，一定是有想法是的，会想到大致该如何先划分一下

这是我们的第一个DDD建模实战案例，后续还有几个案例，包括了在线教育系统部分模块的DDD建模案例、商品中心部分模块的DDD建模重构案例、营销中心部分模块的DDD建模重构案例，这些都是案例，但是学好了之后，基本上就完全明白DDD应该怎么做了，配合之前的内容，效果非常好

大家初步就有能力可以进行DDD建模实战落地了，这些案例对标的是P6级别的同学，可以对自己手头负责的系统去进行DDD建模

后续还会讲三个完整的DDD实战项目：完整订单系统的DDD建模重构项目、完整权限系统的DDD建模重构项目、完整供应链系统的DDD建模重构项目，中间会穿插所有DDD高阶的建模技巧、知识以及思想

学完之后，大家完全有能力站在P7级别的角度去中大型系统进行DDD建模了，哪怕是P8级别的同学，也可以用这套方法去进行大型系统的DDD建模

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/041~060资料/047\_用抽奖系统案例来实战枯燥的DDD上下文映射（1）/笔记.docx**

3616118202

《047\_用抽奖系统案例来实战枯燥的DDD上下文映射（1）》

回顾一下上下文映射的关系，说白了，他其实就是定义一下不同子域/子系统之间的上下游关系、集成关系，也决定了他们之间的集成方式（RPC、RESTful API、MQ）

U和D，一般U是上游，upstream，就是服务提供者，D是下游，就是downstream，就是服务调用者，所以谁调用谁就是D，谁提供谁就是U，表现成线条，就是U流向D，因为D在调用，所以数据是从U流向D的，是这么个意思

合作关系（Partnership）：两个上下文强耦合在一起，基本上就是比如做一个需求吧，多人一起做，大家都是为了一个需求设计自己服务的接口，基本上服务之间以及接口之间都是特定耦合，强行耦合的，为了完成一个完整的需求，难以分开，这种一般是紧密协作的团队内部或者多个团队，都是做一个项目和需求，常常是这种模式

共享内核（Shared Kernel）：两个上下文中间一起商量定义一个通用的领域模型，大家一起用，维护一个独立的通用工程，一般团队内部的不同子域会用这种方式，不同人负责不同的服务，但是抽象一些通用模型出来

客户方-供应方开发（Customer-Supplier Development）：上下文之间互相有商有量的进行通用语言的设计，比如接口提供方设计接口和模型，会问问调用方，调用方也会提自己的需求给提供方，这种的话，一般都是跟合作关系一起用的，有时候大家会这样商量协作，有组织的协作，当你需要做某个事儿的时候，需要团队内部或者其他团队协作，去调用人家的接口，人家可能没有，你可以提个需求，大家商量一下

遵奉者（Conformist）：设计通用语言的时候，谁也别商量，调用的时候互相了解一下，直接调用，这种情况我觉得是很多的，就是有个服务按自己的想法暴露一些接口出来，你需要的时候直接去调用就行了，别多啰嗦，团队内部或者外部都有可能的，就是人家不会配合你的，你就调用

防腐层（Anticorruption Layer）：对其他的服务进行接口调用的时候，要把接口返回给你的东西做一下转换，转换为你内部的实体，避免人家接口改变，你这里就变，这是防腐的一个层

开放主机服务（Open Host Service）和发布语言（Published Language）：这一般是大公司内部，有那种中心之间的调用交互，暴露开放接口，定义一种协议/语言，一般都是RESTful API，或者是第三方公司开放给你的接口

大泥球（Big Ball of Mud）：搞的一塌糊涂

独立路线（SeparateWay）：没关系

抽奖行为子域是D，抽奖活动、奖品库存、抽奖风控三个子域，都是U，营销中心、订单中心、会员中心三个字域，都是U

然后抽奖行为子域和抽奖活动、奖品库存、抽奖风控三个字域之间，全部是PS关系，也就是合作强依赖关系，因为大家的交互都是为了实现特定的同一个抽奖项目和需求而做的，所以就是强依赖关系

然后营销中心、订单中心、会员中心，如果是小型公司里，那么其实直接就是开放自己的一些接口，你抽奖服务直接调用就可以了，人家才不会care你，设计接口还来跟你商量商量呢，你也没法提需求，所以如果小公司里，就是尊奉关系

但是中型以上的公司里，一般各个中心都有中型规模的团队，可能都是一个P8带领每个方向的，所以一般中心对外，都是开放RESTful API的，当然，也有可能是各个中心专门弄一个服务，就是中心的开放API服务，也可以用RPC

不管是什么吧，反正这种中大型公司的中心，对外开放，无论RESTful API还是RPC，都是一个开放主机服务，可能用不同的发布语言，所以他们应该是一个OHS/PL，然后对于抽奖子域自己，一般需要做一个防腐层，ACL，对各个中心进行防腐，把别人的实体转换为自己定义的稳定实体

到此为止，上下文映射搞定，其实映射搞定之后，大致的集成方式你也知道了吧？基本上都是RPC，或者有肯能调用各个中心会是RESTful API，看你们公司具体怎么做的

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/041~060资料/060\_对抽奖系统进行DDD实体建模（5）/笔记.docx**

《060\_对抽奖系统进行DDD实体建模（5）》

在一个分布式系统里，你在进行DDD建模的时候，最好是直接梳理清楚上下游的关系，尽量是反对说，服务A调用服务B，服务B也反过来去调用A，尽量不要这样子，有一个明确的上下游的关系

子域之间的映射关系，集成关系

实体建模，类建模，聚合关系，id标识符，先在文档里建模，我们这里为了省事儿，直接在代码层面建模了，实体在代码层面建出来了

微服务技术架构，Spring Cloud Alibaba，儒猿技术窝的C2C微服务专栏

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/041~060资料/057\_对抽奖系统进行DDD实体建模（2）/笔记.docx**

《057\_对抽奖系统进行DDD实体建模（2）》

在一个分布式系统里，你在进行DDD建模的时候，最好是直接梳理清楚上下游的关系，尽量是反对说，服务A调用服务B，服务B也反过来去调用A，尽量不要这样子，有一个明确的上下游的关系

子域之间的映射关系，集成关系

实体建模，类建模，聚合关系，id标识符，先在文档里建模，我们这里为了省事儿，直接在代码层面建模了，实体在代码层面建出来了

微服务技术架构，Spring Cloud Alibaba，儒猿技术窝的C2C微服务专栏

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/011~020资料/020\_用大白话解释DDD后续要讲的一些牛叉术语/笔记.docx**

《020\_用大白话解释DDD后续要讲的一些牛叉术语》

上下文映射，听着特别牛，其实就是说子域如何进行集成，也就是各个系统如何通过接口进行集成，你需要把别的系统的一些通用语言里的东西在接口里转换/映射为你的系统里的通用语言，如果是在一个系统里划分多个子域的，那么就是一个系统内部的各个模块间如何通过interface进行集成，不同模块的类如何进行映射和转换

实体和值对象，说白了就是去建模你每个子域里的具体类；聚合，就是如何把实体和值对象聚合成一个关系；领域事件，其实有点类似于系统间进行交互时候的一些核心事件，类似什么创建订单之类的

战略设计？？：我极为反对这个翻译，因为搞的特别的高大上，其实完全没必要，说白了，就是限界上下文的划分，子域、核心域、支撑子域、通用子域的定义，上下文映射进行子域集成，我认为在软件设计中，这本质就是一个概要设计罢了，架构师做概要设计，整体设计好，交付各个团队/个人去做详细设计

战术设计？？：极为反对这个翻译，本质就是详细设计的一部分，每个团队/个人拿到自己负责的子域，在里面需要设计通用语言，也就是说，需要通过聚合、实体、值对象、领域事件这些概念，去设计对应的各个类以及核心方法，完事儿了以后还得去设计对应的数据库表结构，正式接口定义，子系统具体业务流程，时序图之类的东西

接下来两三周，把DDD整体概念过一遍，同时基于大家开发过的项目阶段一的电商系统v1.0作为案例，用DDD的思想和概念走一遍，大家对DDD建模的全部流程就清楚了，接下来会细讲DDD建模的各个部分，每个部分都有很多的讲究，中间会穿插大量的实战案例和实战项目，在建模完成后会有代码落地

最后会站在一个很高的角度，对DDD进行全访问的总结，同时梳理如何用DDD设计特别定而且复杂的系统和软件的方法，DDD课程将会持续半年以上，内容极为丰富

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/011~020资料/012\_不用DDD设计业务系统的一个反面例子/笔记.docx**

《012\_不用DDD设计业务系统的一个反面例子》

限界上下文和通用语言，限界上下文解决的是什么问题，如果没有一个好的架构师做一个好的几十人的技术团队的架构设计，此时可能会出现，几十个人维护一个单块系统，所有业务代码在里面混杂在一起

通用语言，一个大泥球的系统里，很混乱，没有所谓的通用语言，在订单的包下面，Order这个类是订单的概念，在仓储的包下面，Order类，是一个发货单的概念，大家没有建立一个通用语言，订单，Order；仓储里面应该有一个自己的通用语言，发货单，DeliveryOrder，通知发货的单据

其实假设不用DDD的话，典型例子就是混乱的开发，也就是搞一个单块系统，而且没有在单块系统里划分出多个限界上下文，代码都混杂在一起，几十个人一起维护，通用语言，也就是类名和术语都很混乱，有的人把Order作为订单，有的人把Order作为仓库发货单，只不过放在不同的包下

最后这种软件就是一个大泥球，经常可能出现乱改代码，改错代码，互相沟通不顺畅，等等问题，总之一句话，这种软件是非常混乱的

订单、商品、库存混在一个单块系统里；仓储、营销、物流混在一个单块系统里；大家对各种专业术语的定义都非常的混乱，有重复，有冲突，同样的一个含义的名词，订单，在订单系统里是Order，在仓储系统里就成了EOrder

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/011~020资料/011\_再次用大白话解释限界上下文、核心域以及通用语言/笔记.docx**

《011\_再次用大白话解释限界上下文、核心域以及通用语言》

第一周的课程里，我们用人话，大白话，初步的让大家对DDD有了一个最为浅显的了解，DDD的书籍、专栏，全部都是用极为晦涩和繁琐复杂的术语和概念上来就开始直接讲DDD，术语，讲的乱七八糟的，学习DDD，很难的

DDD，只是一种软件/业务系统的建模思想和方法论，你只要按照DDD的一整套方法就可以去对任何一个业务软件/业务系统/中间件去进行业务上的建模，业务建模，技术上的实现，编程语言，框架，接口，数据库，中间件，基础架构，架构设计，技术方案

限界上下文，其实说白了就应该是你们公司里的一个较为核心的系统，有独立的代码仓库，有一个独立的团队负责维护一个系统，就是工作在这个限界上下文里，然后在这个限界上下文里需要对各种核心概念定义一套通用的中文和英文的名词和解释，这就是通用语言，团队每个人对这些中文和英文名词的含义都有共识

一个团队可以同时负责维护多个系统，也就是多个限界上下文，但是多个团队不能在一个限界上下文里工作，也就是多个团队同时修改一个系统的代码，那是不可能的，一个限界上下文就是一个独立的完整的系统，有独立的代码仓库、数据库、测试代码，独立团队，独立通用语言

其实有点类似于国家，不同的国家就是不同的限界上下文，每个国家有自己的语言和发音，就是通用语言，比如同样一个发音，可能在一个国家是这个意思，换了一个国家是另外一个意思了

另外这个核心域的意思，其实就是说，如果你的一个限界上下文，也就是一个系统，是你的公司里极为核心的一个系统，就是一个核心系统吧，举个例子，一个电商平台里，商品、订单、库存、营销、仓储、物流、会员，每个大系统都可以是一个限界上下文，都是独立团队维护，都有自己内部的通用语言，然后他们都是公司最为核心的一些系统，所以这些限界上下文也可以叫做核心域

但是如果是一些非核心系统，比如说什么竞对数据爬虫系统、第三方比价系统、BI报表系统，类似这些，如果不支撑公司核心流程运转，主要是锦上添花的系统，那么他就不算是核心域

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/011~020资料/014\_在大厂以及中小公司里如何落地DDD设计？（1）/笔记.docx**

《014\_在大厂以及中小公司里如何落地DDD设计？（1）》

DDD，拆分一些子域（拆分子系统）

如果我们要用DDD去建模一个新系统，其实就很明确了，从业务角度出发，划分出多个子域（其实就是多个限界上下文），然后标记出来哪些是核心域（核心系统），哪些是支撑子域（非核心系统），哪些是通用子域（通用系统，权限、认证之类的），建模完了，划分团队去负责维护和开发，就可以了

子域可以进行集成，DDD，上下文映射，context mapping；DDD这套理论，一点都不亲民，枯燥和晦涩；类建模，实体，值对象，领域事件

DDD这套东西，如果要推行和落地，起码应该是大厂P8~P9的人对自己带的二三十人到100人不等的一个团队去整体的进行推广和落地，DDD这个东西，顶层业务建模和设计，应该是P8/P9去做的

其实这里要给大家说一点，你做DDD的时候，大致有三种角色，第一种是你很牛，是大厂的P8，P9，或者是小公司的高级架构师，一个人带50~100人以上规模的团队，此时你负责的通常是公司的一个业务线，那你必须承担起DDD的责任来，做高层设计，此时就需要运用DDD去拆分子域，标记核心域、支撑子域和通用子域，每个子域可能都是一个几个人或者10人+的团队去负责，有的是10多个人的团队可能负责维护多个子域

然后此时每个团队负责一个或者多个系统，也就是多个子域，那个团队的负责人假设是个P7之类的人物，那么此时他就需要在自己的子域里继续进行DDD建模和拆分，想办法把自己负责的一个系统可能拆分为多个子系统，或者多个服务，然后每个子系统或者服务交给一两个人去负责维护

但是有一点是肯定的，因为这个P7带的团队维护的是一个完整的系统，所以作为一个限界上下文，在这里虽然拆分为多个服务，此时也必须要在团队内部建立统一的通用语言，起码这10个人是按照这个通用语言（中文名词和英文的类名），去开发和维护各个服务的，这样大家才能够理解

另外一种情况，就是你在大厂里是一个P7，或者在中小公司也是个P7的角色，就带5~10人的团队，然后你就被扔了几个系统维护，这个时候没有一个统一的人去做各个团队的DDD，因为你上级老大就是个做管理的，不懂DDD

此时也没关系，你可以灵活落地，你就假设自己仅仅着眼于自己负责的一亩三分地，然后直接对自己负责的一个系统设计几个子域，有核心域，也有支撑子域，然后可以每个子域拆分为一个服务，团队里每个人负责维护一个服务

但是这种情况下，不知道你发现一个问题没？通常是说一个子域（一个限界上下文）里是一套通用语言，但是你现在自己玩儿DDD，搞的太细了，每个子域搞成了一个服务，总不能每个人负责的服务自己设计一套中文名词和英文类名吧，那团队内10个人都没法交流了，太坑爹了

实际上按照DDD理论，DDD应该是P8~P9级别的人玩儿的，拆分出来每个子域是一个限界上下文，对应一个系统，有5~10人负责维护，那5~10人有自己的一套通用语言，这是比较正常的

但是一旦你玩儿细了，教你一个办法，DDD是死的，人是活的，这个时候，你就直接在团队内部去建立一个统一的通用语言，所有子域（所有限界上下文）都共享一套通用语言，大家一起对核心的一个名词和概念，定义统一的中文名称和英文类名，统一思想，统一约定，交流无障碍，就可以了

最后一种比较恶心的情况，就是你无论在大厂还是小公司，都是个P6的角色，就负责一两个小系统，或者一两个小服务，这可怎么办？怎么玩儿DDD呢？还是那句话，强行玩儿！毕竟你得出去面试不是？

就你自己负责的一两个小系统或者小服务，直接把里面的不同的模块设计为不同的子域，这已经完全背离了DDD的思想初衷了，但是还是那句话，DDD是死的，人是活的，没办法，毕竟你得面试，得吃饭

然后不同的模块有核心模块，就是核心域，非核心的就是支撑子域，然后在你的系统范围内建立统一的通用语言，就ok了，出去起码能聊聊DDD的一些落地和实践，而且能从自己的业务角度去聊吧！

另外提一点，其实有时候可能你很多代码都耦合在一个大泥球遗留系统里，其实也是类似P6那种建模办法，直接把不同模块设计为不同的子域，然后不同的模块内部建立通用语言，不同的人负责维护不同包下的代码，互相别乱改代码，别冲突

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/011~020资料/016\_在大厂以及中小公司里如何落地DDD设计？（3）/笔记.docx**

《016\_在大厂以及中小公司里如何落地DDD设计？（3）》

DDD，拆分一些子域（拆分子系统）

如果我们要用DDD去建模一个新系统，其实就很明确了，从业务角度出发，划分出多个子域（其实就是多个限界上下文），然后标记出来哪些是核心域（核心系统），哪些是支撑子域（非核心系统），哪些是通用子域（通用系统，权限、认证之类的），建模完了，划分团队去负责维护和开发，就可以了

子域可以进行集成，DDD，上下文映射，context mapping；DDD这套理论，一点都不亲民，枯燥和晦涩；类建模，实体，值对象，领域事件

DDD这套东西，如果要推行和落地，起码应该是大厂P8~P9的人对自己带的二三十人到100人不等的一个团队去整体的进行推广和落地，DDD这个东西，顶层业务建模和设计，应该是P8/P9去做的

大厂：教育业务线，做的是一些教育类的系统；生鲜业务线

独角兽公司：业务线，订单业务线，商品业务线，教师业务线，CRM业务线，不同的业务线可能都是一个20人或者几十人的团队是在做

其实这里要给大家说一点，你做DDD的时候，大致有三种角色，第一种是你很牛，是大厂的P8，P9，或者是小公司的高级架构师，一个人带50~100人以上规模的团队，此时你负责的通常是公司的一个业务线，那你必须承担起DDD的责任来，做高层设计，此时就需要运用DDD去拆分子域，标记核心域、支撑子域和通用子域，每个子域可能都是一个几个人或者10人+的团队去负责，有的是10多个人的团队可能负责维护多个子域

然后此时每个团队负责一个或者多个系统，也就是多个子域，那个团队的负责人假设是个P7之类的人物，那么此时他就需要在自己的子域里继续进行DDD建模和拆分，想办法把自己负责的一个系统可能拆分为多个子系统，或者多个服务，然后每个子系统或者服务交给一两个人去负责维护

但是有一点是肯定的，因为这个P7带的团队维护的是一个完整的系统，所以作为一个限界上下文，在这里虽然拆分为多个服务，此时也必须要在团队内部建立统一的通用语言，起码这10个人是按照这个通用语言（中文名词和英文的类名），去开发和维护各个服务的，这样大家才能够理解

大部分的同学，根本就不可能说达到P8或者P9那样的一个带大团队的级别，牵头搞DDD，此时你可能还可以分派到一个子域，你可以参与这个大团队的DDD的开发，你可以按照DDD的理论在子域里做类建模，实体、值对象、事件

团队的leader自己对DDD也不太了解，管理角度，希望推行和落地，DDD把系统做的好一些，就让团队里各个负责人自己去搞DDD，没有人牵头在大团队里推行完整的DDD落地，这是目前主流的一个情况

另外一种情况，就是你在大厂里是一个P7，或者在中小公司也是个P7的角色，就带5~10人的团队，然后你就被扔了几个系统维护，这个时候没有一个统一的人去做各个团队的DDD，因为你上级老大就是个做管理的，不懂DDD

此时也没关系，你可以灵活落地，你就假设自己仅仅着眼于自己负责的一亩三分地，然后直接对自己负责的一个系统设计几个子域，有核心域，也有支撑子域，通用子域（开源项目搭建的一个认证系统），然后可以每个子域拆分为一个服务，团队里每个人负责维护一个服务，2个人负责维护一个服务

但是这种情况下，不知道你发现一个问题没？通常是说一个子域（一个限界上下文）里是一套通用语言，但是你现在自己玩儿DDD，搞的太细了，每个子域搞成了一个服务，总不能每个人负责的服务自己设计一套中文名词和英文类名吧，那团队内10个人都没法交流了，太坑爹了

实际上按照DDD理论，DDD应该是P8~P9级别的人玩儿的，拆分出来每个子域是一个限界上下文，对应一个系统，有5~10人负责维护，那5~10人有自己的一套通用语言，这是比较正常的

但是一旦你玩儿细了，教你一个办法，DDD是死的，人是活的，这个时候，你就直接在团队内部去建立一个统一的通用语言，所有子域（所有限界上下文）都共享一套通用语言，大家一起对核心的一个名词和概念，定义统一的中文名称和英文类名，统一思想，统一约定，交流无障碍，就可以了

DDD，我落地了，我带的团队，负责XX系统，我按照DDD的理念，做了子域的拆分，对应到服务的拆分，子域之间是如何集成，服务之间是如何集成的，各个子域（各个服务）内部是如何进行细化的建模（实体、值对象、事件）

最后一种比较恶心的情况，就是你无论在大厂还是小公司，都是个P6的角色，就负责一两个小系统，或者一两个小服务，这可怎么办？怎么玩儿DDD呢？还是那句话，强行玩儿！毕竟你得出去面试不是？

就你自己负责的一两个小系统或者小服务，直接把里面的不同的模块设计为不同的子域，这已经完全背离了DDD的思想初衷了，但是还是那句话，DDD是死的，人是活的，没办法，毕竟你得面试，得吃饭

然后不同的模块有核心模块，就是核心域，非核心的就是支撑子域，然后在你的系统范围内建立统一的通用语言，就ok了，出去起码能聊聊DDD的一些落地和实践，而且能从自己的业务角度去聊吧！

另外提一点，其实有时候可能你很多代码都耦合在一个大泥球遗留系统里，其实也是类似P6那种建模办法，直接把不同模块设计为不同的子域，DDD专业术语里，逻辑子域，然后不同的模块内部建立通用语言，不同的人负责维护不同包下的代码，互相别乱改代码，别冲突

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/011~020资料/019\_案例实战：对电商系统v1.0如何进行DDD建模？（3）/笔记.docx**

《019\_案例实战：对电商系统v1.0如何进行DDD建模？（3）》

domain，这个概念放在DDD里，其实不应该是放一些实体类，domain就是域的概念，core domain，sub domain，带着大家画几个DDD建模图，用电商系统v1.0来画，让大家稍微感受一下DDD建模到底是什么样的

很多第三方外包公司，都会给你做定制化的一些物流系统，不一定说必须你团队自己去做，也可以外包给第三方公司去做物流系统，然后跟你的电商核心系统做一个集成，电商核心系统内部都有一个履约系统，可以自己去实现物流系统，也可以跟外包的第三方公司做的物流系统去做一个集成

WMS系统，甚至都可以定位成是一个通用子域，是一个非常成熟的软件，甚至都不需要定制开发，直接采购成熟系统，做一些配置和简单的集成就可以了

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/011~020资料/018\_案例实战：对电商系统v1.0如何进行DDD建模？（2）/笔记.docx**

《018\_案例实战：对电商系统v1.0如何进行DDD建模？（2）》

domain，这个概念放在DDD里，其实不应该是放一些实体类，domain就是域的概念，core domain，sub domain，带着大家画几个DDD建模图，用电商系统v1.0来画，让大家稍微感受一下DDD建模到底是什么样的

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/011~020资料/013\_用大白话告诉你：子域有哪几种类型？/笔记.docx**

《013\_用大白话告诉你：子域有哪几种类型？》

DDD建模的时候，主要关注的就是域这个概念，子域，还有好几种类型，其实就是一个大系统；域之间需要做集成，大系统之间的集成；对每个域内部进行服务的划分

每一个限界上下文其实都是一个大平台里的完整子系统，所以对应来说，每一个限界上下文也都可以叫做一个子域，一般在做DDD建模的时候，如果简单一点，直接划分出多个子域，然后划分不同的子域类型就可以了，这其实是对一个大型平台或者大型系统进行建模的方法；各个域内部，各个子系统内部的通用语言建模，类建模，事件建模

必须提一点，即使以前没有DDD的时候，有经验的架构师只不过不用这个术语，然后业务系统建模过程跟DDD是一样的，首先就是对大系统划分各个子系统，各个子系统分派给不同的团队去维护，划分就是根据业务来划分的

子域有几种类型，一个是核心域，就是你的大系统里的最最很的几个子系统，通常我们认为是支撑你公司核心业务运转的那些系统就是不同的核心域，分配的是公司最多的研发资源，投入最大；

一个是支撑子域，这个说白了就是公司里的一些锦上添花的辅助系统，非核心系统，比如说什么爬虫系统、BI系统、竞对分析系统、社会化治理系统，这些系统可能会跟你的核心系统有交互，但是主要是辅助用的，甚至可能找外包来做，分配资源较少

最后一个是通用子域，类似什么HR系统、OA系统、CRM系统、权限系统、认证系统之类的，都有通用解决方案，可以直接找第三方厂商进行采购，然后进行集成，当然也可能是找外包做，或者内部做，但是投入的资源会很少很少

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/011~020资料/015\_在大厂以及中小公司里如何落地DDD设计？（2）/笔记.docx**

《015\_在大厂以及中小公司里如何落地DDD设计？（2）》

DDD，拆分一些子域（拆分子系统）

如果我们要用DDD去建模一个新系统，其实就很明确了，从业务角度出发，划分出多个子域（其实就是多个限界上下文），然后标记出来哪些是核心域（核心系统），哪些是支撑子域（非核心系统），哪些是通用子域（通用系统，权限、认证之类的），建模完了，划分团队去负责维护和开发，就可以了

子域可以进行集成，DDD，上下文映射，context mapping；DDD这套理论，一点都不亲民，枯燥和晦涩；类建模，实体，值对象，领域事件

DDD这套东西，如果要推行和落地，起码应该是大厂P8~P9的人对自己带的二三十人到100人不等的一个团队去整体的进行推广和落地，DDD这个东西，顶层业务建模和设计，应该是P8/P9去做的

大厂：教育业务线，做的是一些教育类的系统；生鲜业务线

独角兽公司：业务线，订单业务线，商品业务线，教师业务线，CRM业务线，不同的业务线可能都是一个20人或者几十人的团队是在做

其实这里要给大家说一点，你做DDD的时候，大致有三种角色，第一种是你很牛，是大厂的P8，P9，或者是小公司的高级架构师，一个人带50~100人以上规模的团队，此时你负责的通常是公司的一个业务线，那你必须承担起DDD的责任来，做高层设计，此时就需要运用DDD去拆分子域，标记核心域、支撑子域和通用子域，每个子域可能都是一个几个人或者10人+的团队去负责，有的是10多个人的团队可能负责维护多个子域

然后此时每个团队负责一个或者多个系统，也就是多个子域，那个团队的负责人假设是个P7之类的人物，那么此时他就需要在自己的子域里继续进行DDD建模和拆分，想办法把自己负责的一个系统可能拆分为多个子系统，或者多个服务，然后每个子系统或者服务交给一两个人去负责维护

但是有一点是肯定的，因为这个P7带的团队维护的是一个完整的系统，所以作为一个限界上下文，在这里虽然拆分为多个服务，此时也必须要在团队内部建立统一的通用语言，起码这10个人是按照这个通用语言（中文名词和英文的类名），去开发和维护各个服务的，这样大家才能够理解

大部分的同学，根本就不可能说达到P8或者P9那样的一个带大团队的级别，牵头搞DDD，此时你可能还可以分派到一个子域，你可以参与这个大团队的DDD的开发，你可以按照DDD的理论在子域里做类建模，实体、值对象、事件

团队的leader自己对DDD也不太了解，管理角度，希望推行和落地，DDD把系统做的好一些，就让团队里各个负责人自己去搞DDD，没有人牵头在大团队里推行完整的DDD落地，这是目前主流的一个情况

另外一种情况，就是你在大厂里是一个P7，或者在中小公司也是个P7的角色，就带5~10人的团队，然后你就被扔了几个系统维护，这个时候没有一个统一的人去做各个团队的DDD，因为你上级老大就是个做管理的，不懂DDD

此时也没关系，你可以灵活落地，你就假设自己仅仅着眼于自己负责的一亩三分地，然后直接对自己负责的一个系统设计几个子域，有核心域，也有支撑子域，通用子域（开源项目搭建的一个认证系统），然后可以每个子域拆分为一个服务，团队里每个人负责维护一个服务，2个人负责维护一个服务

但是这种情况下，不知道你发现一个问题没？通常是说一个子域（一个限界上下文）里是一套通用语言，但是你现在自己玩儿DDD，搞的太细了，每个子域搞成了一个服务，总不能每个人负责的服务自己设计一套中文名词和英文类名吧，那团队内10个人都没法交流了，太坑爹了

实际上按照DDD理论，DDD应该是P8~P9级别的人玩儿的，拆分出来每个子域是一个限界上下文，对应一个系统，有5~10人负责维护，那5~10人有自己的一套通用语言，这是比较正常的

但是一旦你玩儿细了，教你一个办法，DDD是死的，人是活的，这个时候，你就直接在团队内部去建立一个统一的通用语言，所有子域（所有限界上下文）都共享一套通用语言，大家一起对核心的一个名词和概念，定义统一的中文名称和英文类名，统一思想，统一约定，交流无障碍，就可以了

最后一种比较恶心的情况，就是你无论在大厂还是小公司，都是个P6的角色，就负责一两个小系统，或者一两个小服务，这可怎么办？怎么玩儿DDD呢？还是那句话，强行玩儿！毕竟你得出去面试不是？

就你自己负责的一两个小系统或者小服务，直接把里面的不同的模块设计为不同的子域，这已经完全背离了DDD的思想初衷了，但是还是那句话，DDD是死的，人是活的，没办法，毕竟你得面试，得吃饭

然后不同的模块有核心模块，就是核心域，非核心的就是支撑子域，然后在你的系统范围内建立统一的通用语言，就ok了，出去起码能聊聊DDD的一些落地和实践，而且能从自己的业务角度去聊吧！

另外提一点，其实有时候可能你很多代码都耦合在一个大泥球遗留系统里，其实也是类似P6那种建模办法，直接把不同模块设计为不同的子域，然后不同的模块内部建立通用语言，不同的人负责维护不同包下的代码，互相别乱改代码，别冲突

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/011~020资料/017\_案例实战：对电商系统v1.0如何进行DDD建模？（1）/笔记.docx**

《017\_案例实战：对电商系统v1.0如何进行DDD建模？（1）》

domain，这个概念放在DDD里，其实不应该是放一些实体类，domain就是域的概念，core domain，sub domain，带着大家画几个DDD建模图，用电商系统v1.0来画，让大家稍微感受一下DDD建模到底是什么样的

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/081~098资料/083\_对促销中心DDD建模进行代码落地（3）/笔记.docx**

3616118202

《083\_对促销中心DDD建模进行代码落地（3）》

促销活动：促销中心的核心就是管理一个一个的促销活动，让运营部门去创建不同的促销活动，配合上一些促销活动的站内 or 站外的营销推广，进而拉动老用户的活跃度和复购率，或者是在站外进行拉新，每一种促销活动都包含了几个维度，包括了时间范围、人群范围、商品范围，等等

促销类型：这个其实我们之前就有过这个概念，促销是有很多种不同的类型的，我们认为完全是可以全部集成到促销这个概念里去的，那当然每一种促销活动都有一种促销类型，比如满减、买赠、直降、优惠券、秒杀，等等

促销规则：顾名思义，根据促销类型定义出来的具体促销规则，这个很灵活多变，之前我们都是考虑用JSON存储的

促销范围：每一个促销活动都会匹配一个促销范围，这个促销范围实际上是通过时间范围、人群范围、商品范围、平台范围（APP or 网站），等各种范围来确定的

候选促销活动查询：大家如果平时玩儿电商APP的话就知道，其实一般来说，促销活动会通过如下方式展现给你，要不然是在电商首页搞一些促销活动的专题页，要不然就是你浏览某个商品的时候，商品详情页里会提示促销活动，要不然就是你把商品加入购物车之后，在购物车浏览界面里可以看到每个商品参加的促销活动，当你提交订单之后，在订单确认页里，也会提示你每个商品参加的促销活动以及每个商品最终的价格，所以在电商的各个环节里，都需要去根据用户所属的人群范围、当前时间、浏览的商品范围，到促销中心查询对应的促销活动列表

优惠结果计算：这个一般是用于订单确认页面的，当你选购了一些商品进入订单确认页面时，如果大家平时玩儿淘宝可能会有印象，就是一个商品可能会同时参与多种促销活动，此时你可以在订单确认页里选择这个商品你最终要参加哪个促销活动，然后根据你选择的促销活动，促销中心可以提供优惠价格计算的服务，要不然是赠送商品给你，要不然是提示可以赠送你几个优惠券，要不然是提示你减免后的价格，甚至有时一个商品可以同时参与多个活动

促销活动优先顺序查询：上面说了，有的商品可能可以同时参加多种促销活动，那么在优惠结果计算的时候，需要按照不同促销活动的优先顺序来依次进行计算

促销活动互斥判断：当然一个商品可以同时参加多个促销活动，也可以设置让多个促销活动进行互斥，也就是说，商品要么选择这个促销活动，要么选择另外一个促销活动

其实分析清楚了以后，你会发现，促销中心的核心就是促销活动、促销类型、候选促销活动查询、优惠结果计算，然后促销活动优先顺序以及促销活动互斥都是依附于促销活动本身的一些关键属性，在优惠结果计算的时候会用到

到此为止，我们就把促销中心的核心关键术语以及定义都提炼出来了

在确定了这些关键术语之后，接下来很关键的一步，就是要确定这些关键术语分别术语什么领域对象？因为在DDD里，要不然是实体，要不然是值对象，实体是有唯一标识id的，值对象其实是没有唯一标识的，或者即使他在数据库里有唯一标识，但是这个唯一标识在业务上没有意义的

所以上面我们分析一下，明确能够发现，促销活动当然是实体了，因为不同的促销活动是有唯一标识的，这个在业务上都是有含义的，然后促销类型当然是值对象了，虽然促销类型可以在数据库里存储单独的表，也可以有数据库层面的ID主键，但是其实他在业务含义里，那个标识是没有意义的

同理的就是促销规则和促销范围，每一个促销范围在数据库里可以单独存一个表，但是其实他本质是促销活动的一个附属数据而已，即使有数据库id也是没有业务上的意义的

这里就可以确定一个聚合根了，那就是优惠活动是聚合根，里面包含了促销类型、促销规则、促销范围

然后对于候选促销活动查询、优惠结果计算、促销活动优先顺序查询、促销活动互斥判断，这些其实本质都是属于行为，并不是在实体/值对象的范围内的，如果是纯行为逻辑的一些东西，那么最好就是把他们建模为领域服务这个概念

说白了，就是把他们还是还原为类似以前的Service层那个概念

比如候选促销活动查询，可能对外还是提供一个接口出去；然后优惠结果计算也是提供一个接口出去，作为一个领域服务接口，就是营销域的服务接口，然后促销活动优先顺序查询、促销活动互斥判断可以设计为内部的领域服务Service，让优惠结果计算去调用

其实最终一个正规的DDD建模好的字域对外暴露的，应该是领域模型（实体模型、值对象模型）+ 对外的领域服务接口 + 对内的实体行为调用接口，实体模型/值对象模型是让其他子域进行实体行为调用的，然后实体引用内部封装了RPC调用，请求到你的实体行为调用接口，你内部还原为实体再调用实体的行为

对外的领域服务接口，就是你领域提供的一些行为，直接提供出去就可以了

这是我比较推荐的一种DDD之间的具体落地的不同子域的集成模式

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/081~098资料/096\_借助销售线索案例的作业布置总结DDD（1）/笔记.docx**

3616118202

《096\_借助销售线索案例的作业布置总结DDD（1）》

销售线索案例业务背景，这个课程是DDD入门级课程，把DDD入门知识以及案例设计，都讲一下，大家学完之后，基本上思路都已经出来了，总结DDD建模入门知识东西，大厂里大电商，光是做营销这块，就几十个人，一个P8级的在带团队

指导朋友公司做的一次DDD项目，抽取出来的

启蒙教育，K12教育，考研/国考/职业证书考试/出国，职业技能教育

在线教育，一般是分成几大块，课程研发 -> 营销引流 -> 体验课社群 / 硬广告的点击&表单录入 -> 销售线索 -> 学生管理 -> 教学服务 -> 老带新促销，这么几个环节

那么销售线索这块一般包含了几个模块：一个是销售线索采集，包括了线上营销推广、线下地推推广带来的销售线索，需要有一个自动/手动录入的过程；销售线索数据管理，包括了潜在用户信息、标签管理、来源渠道，等等；销售线索转化，包括了销售线索自动/手动分配给销售人员，销售线索转化全流程数据化跟进，销售线索付费转化，等等；配置模块，对用户标签、来源渠道、线索分配之类的，都是做到可以配置的

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/081~098资料/094\_在线教育销售线索系统的案例介绍（4）/笔记.docx**

3616118202

《094\_在线教育销售线索系统的案例介绍（4）》

指导朋友公司做的一次DDD项目，抽取出来的

启蒙教育，K12教育，考研/国考/职业证书考试/出国，职业技能教育

在线教育，一般是分成几大块，课程研发 -> 营销引流 -> 体验课社群 / 硬广告的点击&表单录入 -> 销售线索 -> 学生管理 -> 教学服务 -> 老带新促销，这么几个环节

那么销售线索这块一般包含了几个模块：一个是销售线索采集，包括了线上营销推广、线下地推推广带来的销售线索，需要有一个自动/手动录入的过程；销售线索数据管理，包括了潜在用户信息、标签管理、来源渠道，等等；销售线索转化，包括了销售线索自动/手动分配给销售人员，销售线索转化全流程数据化跟进，销售线索付费转化，等等；配置模块，对用户标签、来源渠道、线索分配之类的，都是做到可以配置的

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/081~098资料/089\_对促销中心DDD建模进行代码落地（9）/笔记.docx**

《089\_对促销中心DDD建模进行代码落地（9）》

促销活动：促销中心的核心就是管理一个一个的促销活动，让运营部门去创建不同的促销活动，配合上一些促销活动的站内 or 站外的营销推广，进而拉动老用户的活跃度和复购率，或者是在站外进行拉新，每一种促销活动都包含了几个维度，包括了时间范围、人群范围、商品范围，等等

促销类型：这个其实我们之前就有过这个概念，促销是有很多种不同的类型的，我们认为完全是可以全部集成到促销这个概念里去的，那当然每一种促销活动都有一种促销类型，比如满减、买赠、直降、优惠券、秒杀，等等

促销规则：顾名思义，根据促销类型定义出来的具体促销规则，这个很灵活多变，之前我们都是考虑用JSON存储的

促销范围：每一个促销活动都会匹配一个促销范围，这个促销范围实际上是通过时间范围、人群范围、商品范围、平台范围（APP or 网站），等各种范围来确定的

候选促销活动查询：大家如果平时玩儿电商APP的话就知道，其实一般来说，促销活动会通过如下方式展现给你，要不然是在电商首页搞一些促销活动的专题页，要不然就是你浏览某个商品的时候，商品详情页里会提示促销活动，要不然就是你把商品加入购物车之后，在购物车浏览界面里可以看到每个商品参加的促销活动，当你提交订单之后，在订单确认页里，也会提示你每个商品参加的促销活动以及每个商品最终的价格，所以在电商的各个环节里，都需要去根据用户所属的人群范围、当前时间、浏览的商品范围，到促销中心查询对应的促销活动列表

优惠结果计算：这个一般是用于订单确认页面的，当你选购了一些商品进入订单确认页面时，如果大家平时玩儿淘宝可能会有印象，就是一个商品可能会同时参与多种促销活动，此时你可以在订单确认页里选择这个商品你最终要参加哪个促销活动，然后根据你选择的促销活动，促销中心可以提供优惠价格计算的服务，要不然是赠送商品给你，要不然是提示可以赠送你几个优惠券，要不然是提示你减免后的价格，甚至有时一个商品可以同时参与多个活动

促销活动优先顺序查询：上面说了，有的商品可能可以同时参加多种促销活动，那么在优惠结果计算的时候，需要按照不同促销活动的优先顺序来依次进行计算

促销活动互斥判断：当然一个商品可以同时参加多个促销活动，也可以设置让多个促销活动进行互斥，也就是说，商品要么选择这个促销活动，要么选择另外一个促销活动

其实分析清楚了以后，你会发现，促销中心的核心就是促销活动、促销类型、候选促销活动查询、优惠结果计算，然后促销活动优先顺序以及促销活动互斥都是依附于促销活动本身的一些关键属性，在优惠结果计算的时候会用到

到此为止，我们就把促销中心的核心关键术语以及定义都提炼出来了

在确定了这些关键术语之后，接下来很关键的一步，就是要确定这些关键术语分别术语什么领域对象？因为在DDD里，要不然是实体，要不然是值对象，实体是有唯一标识id的，值对象其实是没有唯一标识的，或者即使他在数据库里有唯一标识，但是这个唯一标识在业务上没有意义的

所以上面我们分析一下，明确能够发现，促销活动当然是实体了，因为不同的促销活动是有唯一标识的，这个在业务上都是有含义的，然后促销类型当然是值对象了，虽然促销类型可以在数据库里存储单独的表，也可以有数据库层面的ID主键，但是其实他在业务含义里，那个标识是没有意义的

同理的就是促销规则和促销范围，每一个促销范围在数据库里可以单独存一个表，但是其实他本质是促销活动的一个附属数据而已，即使有数据库id也是没有业务上的意义的

这里就可以确定一个聚合根了，那就是优惠活动是聚合根，里面包含了促销类型、促销规则、促销范围

然后对于候选促销活动查询、优惠结果计算、促销活动优先顺序查询、促销活动互斥判断，这些其实本质都是属于行为，并不是在实体/值对象的范围内的，如果是纯行为逻辑的一些东西，那么最好就是把他们建模为领域服务这个概念

说白了，就是把他们还是还原为类似以前的Service层那个概念

比如候选促销活动查询，可能对外还是提供一个接口出去；然后优惠结果计算也是提供一个接口出去，作为一个领域服务接口，就是营销域的服务接口，然后促销活动优先顺序查询、促销活动互斥判断可以设计为内部的领域服务Service，让优惠结果计算去调用

其实最终一个正规的DDD建模好的字域对外暴露的，应该是领域模型（实体模型、值对象模型）+ 对外的领域服务接口 + 对内的实体行为调用接口，实体模型/值对象模型是让其他子域进行实体行为调用的，然后实体引用内部封装了RPC调用，请求到你的实体行为调用接口，你内部还原为实体再调用实体的行为

对外的领域服务接口，就是你领域提供的一些行为，直接提供出去就可以了

这是我比较推荐的一种DDD之间的具体落地的不同子域的集成模式

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/081~098资料/095\_在线教育销售线索系统的案例介绍（5）/笔记.docx**

3616118202

《095\_在线教育销售线索系统的案例介绍（5）》

指导朋友公司做的一次DDD项目，抽取出来的

启蒙教育，K12教育，考研/国考/职业证书考试/出国，职业技能教育

在线教育，一般是分成几大块，课程研发 -> 营销引流 -> 体验课社群 / 硬广告的点击&表单录入 -> 销售线索 -> 学生管理 -> 教学服务 -> 老带新促销，这么几个环节

那么销售线索这块一般包含了几个模块：一个是销售线索采集，包括了线上营销推广、线下地推推广带来的销售线索，需要有一个自动/手动录入的过程；销售线索数据管理，包括了潜在用户信息、标签管理、来源渠道，等等；销售线索转化，包括了销售线索自动/手动分配给销售人员，销售线索转化全流程数据化跟进，销售线索付费转化，等等；配置模块，对用户标签、来源渠道、线索分配之类的，都是做到可以配置的

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/081~098资料/087\_对促销中心DDD建模进行代码落地（7）/笔记.docx**

3616118202

《087\_对促销中心DDD建模进行代码落地（7）》

促销活动：促销中心的核心就是管理一个一个的促销活动，让运营部门去创建不同的促销活动，配合上一些促销活动的站内 or 站外的营销推广，进而拉动老用户的活跃度和复购率，或者是在站外进行拉新，每一种促销活动都包含了几个维度，包括了时间范围、人群范围、商品范围，等等

促销类型：这个其实我们之前就有过这个概念，促销是有很多种不同的类型的，我们认为完全是可以全部集成到促销这个概念里去的，那当然每一种促销活动都有一种促销类型，比如满减、买赠、直降、优惠券、秒杀，等等

促销规则：顾名思义，根据促销类型定义出来的具体促销规则，这个很灵活多变，之前我们都是考虑用JSON存储的

促销范围：每一个促销活动都会匹配一个促销范围，这个促销范围实际上是通过时间范围、人群范围、商品范围、平台范围（APP or 网站），等各种范围来确定的

候选促销活动查询：大家如果平时玩儿电商APP的话就知道，其实一般来说，促销活动会通过如下方式展现给你，要不然是在电商首页搞一些促销活动的专题页，要不然就是你浏览某个商品的时候，商品详情页里会提示促销活动，要不然就是你把商品加入购物车之后，在购物车浏览界面里可以看到每个商品参加的促销活动，当你提交订单之后，在订单确认页里，也会提示你每个商品参加的促销活动以及每个商品最终的价格，所以在电商的各个环节里，都需要去根据用户所属的人群范围、当前时间、浏览的商品范围，到促销中心查询对应的促销活动列表

优惠结果计算：这个一般是用于订单确认页面的，当你选购了一些商品进入订单确认页面时，如果大家平时玩儿淘宝可能会有印象，就是一个商品可能会同时参与多种促销活动，此时你可以在订单确认页里选择这个商品你最终要参加哪个促销活动，然后根据你选择的促销活动，促销中心可以提供优惠价格计算的服务，要不然是赠送商品给你，要不然是提示可以赠送你几个优惠券，要不然是提示你减免后的价格，甚至有时一个商品可以同时参与多个活动

促销活动优先顺序查询：上面说了，有的商品可能可以同时参加多种促销活动，那么在优惠结果计算的时候，需要按照不同促销活动的优先顺序来依次进行计算

促销活动互斥判断：当然一个商品可以同时参加多个促销活动，也可以设置让多个促销活动进行互斥，也就是说，商品要么选择这个促销活动，要么选择另外一个促销活动

其实分析清楚了以后，你会发现，促销中心的核心就是促销活动、促销类型、候选促销活动查询、优惠结果计算，然后促销活动优先顺序以及促销活动互斥都是依附于促销活动本身的一些关键属性，在优惠结果计算的时候会用到

到此为止，我们就把促销中心的核心关键术语以及定义都提炼出来了

在确定了这些关键术语之后，接下来很关键的一步，就是要确定这些关键术语分别术语什么领域对象？因为在DDD里，要不然是实体，要不然是值对象，实体是有唯一标识id的，值对象其实是没有唯一标识的，或者即使他在数据库里有唯一标识，但是这个唯一标识在业务上没有意义的

所以上面我们分析一下，明确能够发现，促销活动当然是实体了，因为不同的促销活动是有唯一标识的，这个在业务上都是有含义的，然后促销类型当然是值对象了，虽然促销类型可以在数据库里存储单独的表，也可以有数据库层面的ID主键，但是其实他在业务含义里，那个标识是没有意义的

同理的就是促销规则和促销范围，每一个促销范围在数据库里可以单独存一个表，但是其实他本质是促销活动的一个附属数据而已，即使有数据库id也是没有业务上的意义的

这里就可以确定一个聚合根了，那就是优惠活动是聚合根，里面包含了促销类型、促销规则、促销范围

然后对于候选促销活动查询、优惠结果计算、促销活动优先顺序查询、促销活动互斥判断，这些其实本质都是属于行为，并不是在实体/值对象的范围内的，如果是纯行为逻辑的一些东西，那么最好就是把他们建模为领域服务这个概念

说白了，就是把他们还是还原为类似以前的Service层那个概念

比如候选促销活动查询，可能对外还是提供一个接口出去；然后优惠结果计算也是提供一个接口出去，作为一个领域服务接口，就是营销域的服务接口，然后促销活动优先顺序查询、促销活动互斥判断可以设计为内部的领域服务Service，让优惠结果计算去调用

其实最终一个正规的DDD建模好的字域对外暴露的，应该是领域模型（实体模型、值对象模型）+ 对外的领域服务接口 + 对内的实体行为调用接口，实体模型/值对象模型是让其他子域进行实体行为调用的，然后实体引用内部封装了RPC调用，请求到你的实体行为调用接口，你内部还原为实体再调用实体的行为

对外的领域服务接口，就是你领域提供的一些行为，直接提供出去就可以了

这是我比较推荐的一种DDD之间的具体落地的不同子域的集成模式

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/081~098资料/084\_对促销中心DDD建模进行代码落地（4）/笔记.docx**

3616118202

《084\_对促销中心DDD建模进行代码落地（4）》

促销活动：促销中心的核心就是管理一个一个的促销活动，让运营部门去创建不同的促销活动，配合上一些促销活动的站内 or 站外的营销推广，进而拉动老用户的活跃度和复购率，或者是在站外进行拉新，每一种促销活动都包含了几个维度，包括了时间范围、人群范围、商品范围，等等

促销类型：这个其实我们之前就有过这个概念，促销是有很多种不同的类型的，我们认为完全是可以全部集成到促销这个概念里去的，那当然每一种促销活动都有一种促销类型，比如满减、买赠、直降、优惠券、秒杀，等等

促销规则：顾名思义，根据促销类型定义出来的具体促销规则，这个很灵活多变，之前我们都是考虑用JSON存储的

促销范围：每一个促销活动都会匹配一个促销范围，这个促销范围实际上是通过时间范围、人群范围、商品范围、平台范围（APP or 网站），等各种范围来确定的

候选促销活动查询：大家如果平时玩儿电商APP的话就知道，其实一般来说，促销活动会通过如下方式展现给你，要不然是在电商首页搞一些促销活动的专题页，要不然就是你浏览某个商品的时候，商品详情页里会提示促销活动，要不然就是你把商品加入购物车之后，在购物车浏览界面里可以看到每个商品参加的促销活动，当你提交订单之后，在订单确认页里，也会提示你每个商品参加的促销活动以及每个商品最终的价格，所以在电商的各个环节里，都需要去根据用户所属的人群范围、当前时间、浏览的商品范围，到促销中心查询对应的促销活动列表

优惠结果计算：这个一般是用于订单确认页面的，当你选购了一些商品进入订单确认页面时，如果大家平时玩儿淘宝可能会有印象，就是一个商品可能会同时参与多种促销活动，此时你可以在订单确认页里选择这个商品你最终要参加哪个促销活动，然后根据你选择的促销活动，促销中心可以提供优惠价格计算的服务，要不然是赠送商品给你，要不然是提示可以赠送你几个优惠券，要不然是提示你减免后的价格，甚至有时一个商品可以同时参与多个活动

促销活动优先顺序查询：上面说了，有的商品可能可以同时参加多种促销活动，那么在优惠结果计算的时候，需要按照不同促销活动的优先顺序来依次进行计算

促销活动互斥判断：当然一个商品可以同时参加多个促销活动，也可以设置让多个促销活动进行互斥，也就是说，商品要么选择这个促销活动，要么选择另外一个促销活动

其实分析清楚了以后，你会发现，促销中心的核心就是促销活动、促销类型、候选促销活动查询、优惠结果计算，然后促销活动优先顺序以及促销活动互斥都是依附于促销活动本身的一些关键属性，在优惠结果计算的时候会用到

到此为止，我们就把促销中心的核心关键术语以及定义都提炼出来了

在确定了这些关键术语之后，接下来很关键的一步，就是要确定这些关键术语分别术语什么领域对象？因为在DDD里，要不然是实体，要不然是值对象，实体是有唯一标识id的，值对象其实是没有唯一标识的，或者即使他在数据库里有唯一标识，但是这个唯一标识在业务上没有意义的

所以上面我们分析一下，明确能够发现，促销活动当然是实体了，因为不同的促销活动是有唯一标识的，这个在业务上都是有含义的，然后促销类型当然是值对象了，虽然促销类型可以在数据库里存储单独的表，也可以有数据库层面的ID主键，但是其实他在业务含义里，那个标识是没有意义的

同理的就是促销规则和促销范围，每一个促销范围在数据库里可以单独存一个表，但是其实他本质是促销活动的一个附属数据而已，即使有数据库id也是没有业务上的意义的

这里就可以确定一个聚合根了，那就是优惠活动是聚合根，里面包含了促销类型、促销规则、促销范围

然后对于候选促销活动查询、优惠结果计算、促销活动优先顺序查询、促销活动互斥判断，这些其实本质都是属于行为，并不是在实体/值对象的范围内的，如果是纯行为逻辑的一些东西，那么最好就是把他们建模为领域服务这个概念

说白了，就是把他们还是还原为类似以前的Service层那个概念

比如候选促销活动查询，可能对外还是提供一个接口出去；然后优惠结果计算也是提供一个接口出去，作为一个领域服务接口，就是营销域的服务接口，然后促销活动优先顺序查询、促销活动互斥判断可以设计为内部的领域服务Service，让优惠结果计算去调用

其实最终一个正规的DDD建模好的字域对外暴露的，应该是领域模型（实体模型、值对象模型）+ 对外的领域服务接口 + 对内的实体行为调用接口，实体模型/值对象模型是让其他子域进行实体行为调用的，然后实体引用内部封装了RPC调用，请求到你的实体行为调用接口，你内部还原为实体再调用实体的行为

对外的领域服务接口，就是你领域提供的一些行为，直接提供出去就可以了

这是我比较推荐的一种DDD之间的具体落地的不同子域的集成模式

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/081~098资料/091\_在线教育销售线索系统的案例介绍（1）/笔记.docx**

《091\_在线教育销售线索系统的案例介绍》

指导朋友公司做的一次DDD项目，抽取出来的

启蒙教育，K12教育，考研/国考/职业证书考试，职业技能教育

在线教育，一般是分成几大块，营销管理 -> 销售线索 -> 学生管理 -> 教学服务 -> 老带新促销，这么几个环节

那么销售线索这块一般包含了几个模块：一个是销售线索采集，包括了线上营销推广、线下地推推广带来的销售线索，需要有一个自动/手动录入的过程；销售线索数据管理，包括了潜在用户信息、标签管理、来源渠道，等等；销售线索转化，包括了销售线索自动/手动分配给销售人员，销售线索转化全流程数据化跟进，销售线索付费转化，等等；配置模块，对用户标签、来源渠道、线索分配之类的，都是做到可以配置的

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/081~098资料/085\_对促销中心DDD建模进行代码落地（5）/笔记.docx**

3616118202

《085\_对促销中心DDD建模进行代码落地（5）》

促销活动：促销中心的核心就是管理一个一个的促销活动，让运营部门去创建不同的促销活动，配合上一些促销活动的站内 or 站外的营销推广，进而拉动老用户的活跃度和复购率，或者是在站外进行拉新，每一种促销活动都包含了几个维度，包括了时间范围、人群范围、商品范围，等等

促销类型：这个其实我们之前就有过这个概念，促销是有很多种不同的类型的，我们认为完全是可以全部集成到促销这个概念里去的，那当然每一种促销活动都有一种促销类型，比如满减、买赠、直降、优惠券、秒杀，等等

促销规则：顾名思义，根据促销类型定义出来的具体促销规则，这个很灵活多变，之前我们都是考虑用JSON存储的

促销范围：每一个促销活动都会匹配一个促销范围，这个促销范围实际上是通过时间范围、人群范围、商品范围、平台范围（APP or 网站），等各种范围来确定的

候选促销活动查询：大家如果平时玩儿电商APP的话就知道，其实一般来说，促销活动会通过如下方式展现给你，要不然是在电商首页搞一些促销活动的专题页，要不然就是你浏览某个商品的时候，商品详情页里会提示促销活动，要不然就是你把商品加入购物车之后，在购物车浏览界面里可以看到每个商品参加的促销活动，当你提交订单之后，在订单确认页里，也会提示你每个商品参加的促销活动以及每个商品最终的价格，所以在电商的各个环节里，都需要去根据用户所属的人群范围、当前时间、浏览的商品范围，到促销中心查询对应的促销活动列表

优惠结果计算：这个一般是用于订单确认页面的，当你选购了一些商品进入订单确认页面时，如果大家平时玩儿淘宝可能会有印象，就是一个商品可能会同时参与多种促销活动，此时你可以在订单确认页里选择这个商品你最终要参加哪个促销活动，然后根据你选择的促销活动，促销中心可以提供优惠价格计算的服务，要不然是赠送商品给你，要不然是提示可以赠送你几个优惠券，要不然是提示你减免后的价格，甚至有时一个商品可以同时参与多个活动

促销活动优先顺序查询：上面说了，有的商品可能可以同时参加多种促销活动，那么在优惠结果计算的时候，需要按照不同促销活动的优先顺序来依次进行计算

促销活动互斥判断：当然一个商品可以同时参加多个促销活动，也可以设置让多个促销活动进行互斥，也就是说，商品要么选择这个促销活动，要么选择另外一个促销活动

其实分析清楚了以后，你会发现，促销中心的核心就是促销活动、促销类型、候选促销活动查询、优惠结果计算，然后促销活动优先顺序以及促销活动互斥都是依附于促销活动本身的一些关键属性，在优惠结果计算的时候会用到

到此为止，我们就把促销中心的核心关键术语以及定义都提炼出来了

在确定了这些关键术语之后，接下来很关键的一步，就是要确定这些关键术语分别术语什么领域对象？因为在DDD里，要不然是实体，要不然是值对象，实体是有唯一标识id的，值对象其实是没有唯一标识的，或者即使他在数据库里有唯一标识，但是这个唯一标识在业务上没有意义的

所以上面我们分析一下，明确能够发现，促销活动当然是实体了，因为不同的促销活动是有唯一标识的，这个在业务上都是有含义的，然后促销类型当然是值对象了，虽然促销类型可以在数据库里存储单独的表，也可以有数据库层面的ID主键，但是其实他在业务含义里，那个标识是没有意义的

同理的就是促销规则和促销范围，每一个促销范围在数据库里可以单独存一个表，但是其实他本质是促销活动的一个附属数据而已，即使有数据库id也是没有业务上的意义的

这里就可以确定一个聚合根了，那就是优惠活动是聚合根，里面包含了促销类型、促销规则、促销范围

然后对于候选促销活动查询、优惠结果计算、促销活动优先顺序查询、促销活动互斥判断，这些其实本质都是属于行为，并不是在实体/值对象的范围内的，如果是纯行为逻辑的一些东西，那么最好就是把他们建模为领域服务这个概念

说白了，就是把他们还是还原为类似以前的Service层那个概念

比如候选促销活动查询，可能对外还是提供一个接口出去；然后优惠结果计算也是提供一个接口出去，作为一个领域服务接口，就是营销域的服务接口，然后促销活动优先顺序查询、促销活动互斥判断可以设计为内部的领域服务Service，让优惠结果计算去调用

其实最终一个正规的DDD建模好的字域对外暴露的，应该是领域模型（实体模型、值对象模型）+ 对外的领域服务接口 + 对内的实体行为调用接口，实体模型/值对象模型是让其他子域进行实体行为调用的，然后实体引用内部封装了RPC调用，请求到你的实体行为调用接口，你内部还原为实体再调用实体的行为

对外的领域服务接口，就是你领域提供的一些行为，直接提供出去就可以了

这是我比较推荐的一种DDD之间的具体落地的不同子域的集成模式

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/081~098资料/098\_借助销售线索案例的作业布置总结DDD（3）/笔记.docx**

《098\_借助销售线索案例的作业布置总结DDD（3）》

销售线索案例业务背景，这个课程是DDD入门级课程，把DDD入门知识以及案例设计，都讲一下，大家学完之后，基本上思路都已经出来了，总结DDD建模入门知识东西，大厂里大电商，光是做营销这块，就几十个人，一个P8级的在带团队

指导朋友公司做的一次DDD项目，抽取出来的

实体、值对象、领域服务

DDD概念不是核心，哪怕没有DDD的时候，我曾经在多年之前，也没接触所谓的国外的DDD，当时我们也会按照DDD的思路去设计，按照纯面向对象的方式去设计，把业务流程还原出来

设计对外提供的接口了，在不同的领域之间其实是可以出现不同的实体进行调用的，有一些人或者国外的书，或者是国内的人，在实践DDD的时候，领域之间还是会基于普通的接口来进行调用

如果说其他的系统/域，也是基于DDD思想做的，暴露出去一个实体的引用，UserRef，对于其他系统就不要考虑所谓的接口的概念了，直接就是调用你的User，屏蔽掉一些接口的调用，人家调用你的UserRef的行为，底层直接RPC调用到你的系统接口

你的系统接口还原出来对应的实体，User，执行里面的行为就可以了

提供的还是以前的RPC接口，或者是HTTP接口，接口收到了请求之后，可以进行一定的编排，他需要去还原出来各个领域对象，编排他们的执行顺序，依次去调用他们，完成一个请求的真实的业务流程

设计模式里的构造器模式，构造实体是很优雅的代码

启蒙教育，K12教育，考研/国考/职业证书考试/出国，职业技能教育

在线教育，一般是分成几大块，课程研发 -> 营销引流 -> 体验课社群 / 硬广告的点击&表单录入 -> 销售线索 -> 学生管理 -> 教学服务 -> 老带新促销，这么几个环节

那么销售线索这块一般包含了几个模块：一个是销售线索采集，包括了线上营销推广、线下地推推广带来的销售线索，需要有一个自动/手动录入的过程；销售线索数据管理，包括了潜在用户信息、标签管理、来源渠道，等等；销售线索转化，包括了销售线索自动/手动分配给销售人员，销售线索转化全流程数据化跟进，销售线索付费转化，等等；配置模块，对用户标签、来源渠道、线索分配之类的，都是做到可以配置的

DDD这块，会开启两个真实的项目，一个是订单系统的DDD建模，一个是供应链系统的DDD建模，在则两个项目里会融入大量的DDD高阶建模的思想和知识，以及基于DDD的业务架构的设计

业务架构这块

如何做一个复杂的业务系统，DDD、业务系统架构设计、业务系统重构、JDK（源码）、Spring/Spring Boot/Spring Cloud Alibaba/MyBatis（源码），mysql和jvm的知识在儒猿技术窝的专栏里都有了，软件工程这块的知识在阶段一里都有了

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/081~098资料/092\_在线教育销售线索系统的案例介绍（2）/笔记.docx**

《092\_在线教育销售线索系统的案例介绍（2）》

指导朋友公司做的一次DDD项目，抽取出来的

启蒙教育，K12教育，考研/国考/职业证书考试/出国，职业技能教育

在线教育，一般是分成几大块，课程研发 -> 营销管理 -> 销售线索 -> 学生管理 -> 教学服务 -> 老带新促销，这么几个环节

那么销售线索这块一般包含了几个模块：一个是销售线索采集，包括了线上营销推广、线下地推推广带来的销售线索，需要有一个自动/手动录入的过程；销售线索数据管理，包括了潜在用户信息、标签管理、来源渠道，等等；销售线索转化，包括了销售线索自动/手动分配给销售人员，销售线索转化全流程数据化跟进，销售线索付费转化，等等；配置模块，对用户标签、来源渠道、线索分配之类的，都是做到可以配置的

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/081~098资料/097\_借助销售线索案例的作业布置总结DDD（2）/笔记.docx**

《097\_借助销售线索案例的作业布置总结DDD（2）》

销售线索案例业务背景，这个课程是DDD入门级课程，把DDD入门知识以及案例设计，都讲一下，大家学完之后，基本上思路都已经出来了，总结DDD建模入门知识东西，大厂里大电商，光是做营销这块，就几十个人，一个P8级的在带团队

指导朋友公司做的一次DDD项目，抽取出来的

实体、值对象、领域服务

DDD概念不是核心，哪怕没有DDD的时候，我曾经在多年之前，也没接触所谓的国外的DDD，当时我们也会按照DDD的思路去设计，按照纯面向对象的方式去设计，把业务流程还原出来

启蒙教育，K12教育，考研/国考/职业证书考试/出国，职业技能教育

在线教育，一般是分成几大块，课程研发 -> 营销引流 -> 体验课社群 / 硬广告的点击&表单录入 -> 销售线索 -> 学生管理 -> 教学服务 -> 老带新促销，这么几个环节

那么销售线索这块一般包含了几个模块：一个是销售线索采集，包括了线上营销推广、线下地推推广带来的销售线索，需要有一个自动/手动录入的过程；销售线索数据管理，包括了潜在用户信息、标签管理、来源渠道，等等；销售线索转化，包括了销售线索自动/手动分配给销售人员，销售线索转化全流程数据化跟进，销售线索付费转化，等等；配置模块，对用户标签、来源渠道、线索分配之类的，都是做到可以配置的

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/081~098资料/086\_对促销中心DDD建模进行代码落地（6）/笔记.docx**

3616118202

《086\_对促销中心DDD建模进行代码落地（6）》

促销活动：促销中心的核心就是管理一个一个的促销活动，让运营部门去创建不同的促销活动，配合上一些促销活动的站内 or 站外的营销推广，进而拉动老用户的活跃度和复购率，或者是在站外进行拉新，每一种促销活动都包含了几个维度，包括了时间范围、人群范围、商品范围，等等

促销类型：这个其实我们之前就有过这个概念，促销是有很多种不同的类型的，我们认为完全是可以全部集成到促销这个概念里去的，那当然每一种促销活动都有一种促销类型，比如满减、买赠、直降、优惠券、秒杀，等等

促销规则：顾名思义，根据促销类型定义出来的具体促销规则，这个很灵活多变，之前我们都是考虑用JSON存储的

促销范围：每一个促销活动都会匹配一个促销范围，这个促销范围实际上是通过时间范围、人群范围、商品范围、平台范围（APP or 网站），等各种范围来确定的

候选促销活动查询：大家如果平时玩儿电商APP的话就知道，其实一般来说，促销活动会通过如下方式展现给你，要不然是在电商首页搞一些促销活动的专题页，要不然就是你浏览某个商品的时候，商品详情页里会提示促销活动，要不然就是你把商品加入购物车之后，在购物车浏览界面里可以看到每个商品参加的促销活动，当你提交订单之后，在订单确认页里，也会提示你每个商品参加的促销活动以及每个商品最终的价格，所以在电商的各个环节里，都需要去根据用户所属的人群范围、当前时间、浏览的商品范围，到促销中心查询对应的促销活动列表

优惠结果计算：这个一般是用于订单确认页面的，当你选购了一些商品进入订单确认页面时，如果大家平时玩儿淘宝可能会有印象，就是一个商品可能会同时参与多种促销活动，此时你可以在订单确认页里选择这个商品你最终要参加哪个促销活动，然后根据你选择的促销活动，促销中心可以提供优惠价格计算的服务，要不然是赠送商品给你，要不然是提示可以赠送你几个优惠券，要不然是提示你减免后的价格，甚至有时一个商品可以同时参与多个活动

促销活动优先顺序查询：上面说了，有的商品可能可以同时参加多种促销活动，那么在优惠结果计算的时候，需要按照不同促销活动的优先顺序来依次进行计算

促销活动互斥判断：当然一个商品可以同时参加多个促销活动，也可以设置让多个促销活动进行互斥，也就是说，商品要么选择这个促销活动，要么选择另外一个促销活动

其实分析清楚了以后，你会发现，促销中心的核心就是促销活动、促销类型、候选促销活动查询、优惠结果计算，然后促销活动优先顺序以及促销活动互斥都是依附于促销活动本身的一些关键属性，在优惠结果计算的时候会用到

到此为止，我们就把促销中心的核心关键术语以及定义都提炼出来了

在确定了这些关键术语之后，接下来很关键的一步，就是要确定这些关键术语分别术语什么领域对象？因为在DDD里，要不然是实体，要不然是值对象，实体是有唯一标识id的，值对象其实是没有唯一标识的，或者即使他在数据库里有唯一标识，但是这个唯一标识在业务上没有意义的

所以上面我们分析一下，明确能够发现，促销活动当然是实体了，因为不同的促销活动是有唯一标识的，这个在业务上都是有含义的，然后促销类型当然是值对象了，虽然促销类型可以在数据库里存储单独的表，也可以有数据库层面的ID主键，但是其实他在业务含义里，那个标识是没有意义的

同理的就是促销规则和促销范围，每一个促销范围在数据库里可以单独存一个表，但是其实他本质是促销活动的一个附属数据而已，即使有数据库id也是没有业务上的意义的

这里就可以确定一个聚合根了，那就是优惠活动是聚合根，里面包含了促销类型、促销规则、促销范围

然后对于候选促销活动查询、优惠结果计算、促销活动优先顺序查询、促销活动互斥判断，这些其实本质都是属于行为，并不是在实体/值对象的范围内的，如果是纯行为逻辑的一些东西，那么最好就是把他们建模为领域服务这个概念

说白了，就是把他们还是还原为类似以前的Service层那个概念

比如候选促销活动查询，可能对外还是提供一个接口出去；然后优惠结果计算也是提供一个接口出去，作为一个领域服务接口，就是营销域的服务接口，然后促销活动优先顺序查询、促销活动互斥判断可以设计为内部的领域服务Service，让优惠结果计算去调用

其实最终一个正规的DDD建模好的字域对外暴露的，应该是领域模型（实体模型、值对象模型）+ 对外的领域服务接口 + 对内的实体行为调用接口，实体模型/值对象模型是让其他子域进行实体行为调用的，然后实体引用内部封装了RPC调用，请求到你的实体行为调用接口，你内部还原为实体再调用实体的行为

对外的领域服务接口，就是你领域提供的一些行为，直接提供出去就可以了

这是我比较推荐的一种DDD之间的具体落地的不同子域的集成模式

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/081~098资料/093\_在线教育销售线索系统的案例介绍（3）/笔记.docx**

3616118202

《093\_在线教育销售线索系统的案例介绍（3）》

指导朋友公司做的一次DDD项目，抽取出来的

启蒙教育，K12教育，考研/国考/职业证书考试/出国，职业技能教育

在线教育，一般是分成几大块，课程研发 -> 营销引流 -> 销售线索 -> 学生管理 -> 教学服务 -> 老带新促销，这么几个环节

那么销售线索这块一般包含了几个模块：一个是销售线索采集，包括了线上营销推广、线下地推推广带来的销售线索，需要有一个自动/手动录入的过程；销售线索数据管理，包括了潜在用户信息、标签管理、来源渠道，等等；销售线索转化，包括了销售线索自动/手动分配给销售人员，销售线索转化全流程数据化跟进，销售线索付费转化，等等；配置模块，对用户标签、来源渠道、线索分配之类的，都是做到可以配置的

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/081~098资料/088\_对促销中心DDD建模进行代码落地（8）/笔记.docx**

3616118202

《088\_对促销中心DDD建模进行代码落地（8）》

促销活动：促销中心的核心就是管理一个一个的促销活动，让运营部门去创建不同的促销活动，配合上一些促销活动的站内 or 站外的营销推广，进而拉动老用户的活跃度和复购率，或者是在站外进行拉新，每一种促销活动都包含了几个维度，包括了时间范围、人群范围、商品范围，等等

促销类型：这个其实我们之前就有过这个概念，促销是有很多种不同的类型的，我们认为完全是可以全部集成到促销这个概念里去的，那当然每一种促销活动都有一种促销类型，比如满减、买赠、直降、优惠券、秒杀，等等

促销规则：顾名思义，根据促销类型定义出来的具体促销规则，这个很灵活多变，之前我们都是考虑用JSON存储的

促销范围：每一个促销活动都会匹配一个促销范围，这个促销范围实际上是通过时间范围、人群范围、商品范围、平台范围（APP or 网站），等各种范围来确定的

候选促销活动查询：大家如果平时玩儿电商APP的话就知道，其实一般来说，促销活动会通过如下方式展现给你，要不然是在电商首页搞一些促销活动的专题页，要不然就是你浏览某个商品的时候，商品详情页里会提示促销活动，要不然就是你把商品加入购物车之后，在购物车浏览界面里可以看到每个商品参加的促销活动，当你提交订单之后，在订单确认页里，也会提示你每个商品参加的促销活动以及每个商品最终的价格，所以在电商的各个环节里，都需要去根据用户所属的人群范围、当前时间、浏览的商品范围，到促销中心查询对应的促销活动列表

优惠结果计算：这个一般是用于订单确认页面的，当你选购了一些商品进入订单确认页面时，如果大家平时玩儿淘宝可能会有印象，就是一个商品可能会同时参与多种促销活动，此时你可以在订单确认页里选择这个商品你最终要参加哪个促销活动，然后根据你选择的促销活动，促销中心可以提供优惠价格计算的服务，要不然是赠送商品给你，要不然是提示可以赠送你几个优惠券，要不然是提示你减免后的价格，甚至有时一个商品可以同时参与多个活动

促销活动优先顺序查询：上面说了，有的商品可能可以同时参加多种促销活动，那么在优惠结果计算的时候，需要按照不同促销活动的优先顺序来依次进行计算

促销活动互斥判断：当然一个商品可以同时参加多个促销活动，也可以设置让多个促销活动进行互斥，也就是说，商品要么选择这个促销活动，要么选择另外一个促销活动

其实分析清楚了以后，你会发现，促销中心的核心就是促销活动、促销类型、候选促销活动查询、优惠结果计算，然后促销活动优先顺序以及促销活动互斥都是依附于促销活动本身的一些关键属性，在优惠结果计算的时候会用到

到此为止，我们就把促销中心的核心关键术语以及定义都提炼出来了

在确定了这些关键术语之后，接下来很关键的一步，就是要确定这些关键术语分别术语什么领域对象？因为在DDD里，要不然是实体，要不然是值对象，实体是有唯一标识id的，值对象其实是没有唯一标识的，或者即使他在数据库里有唯一标识，但是这个唯一标识在业务上没有意义的

所以上面我们分析一下，明确能够发现，促销活动当然是实体了，因为不同的促销活动是有唯一标识的，这个在业务上都是有含义的，然后促销类型当然是值对象了，虽然促销类型可以在数据库里存储单独的表，也可以有数据库层面的ID主键，但是其实他在业务含义里，那个标识是没有意义的

同理的就是促销规则和促销范围，每一个促销范围在数据库里可以单独存一个表，但是其实他本质是促销活动的一个附属数据而已，即使有数据库id也是没有业务上的意义的

这里就可以确定一个聚合根了，那就是优惠活动是聚合根，里面包含了促销类型、促销规则、促销范围

然后对于候选促销活动查询、优惠结果计算、促销活动优先顺序查询、促销活动互斥判断，这些其实本质都是属于行为，并不是在实体/值对象的范围内的，如果是纯行为逻辑的一些东西，那么最好就是把他们建模为领域服务这个概念

说白了，就是把他们还是还原为类似以前的Service层那个概念

比如候选促销活动查询，可能对外还是提供一个接口出去；然后优惠结果计算也是提供一个接口出去，作为一个领域服务接口，就是营销域的服务接口，然后促销活动优先顺序查询、促销活动互斥判断可以设计为内部的领域服务Service，让优惠结果计算去调用

其实最终一个正规的DDD建模好的字域对外暴露的，应该是领域模型（实体模型、值对象模型）+ 对外的领域服务接口 + 对内的实体行为调用接口，实体模型/值对象模型是让其他子域进行实体行为调用的，然后实体引用内部封装了RPC调用，请求到你的实体行为调用接口，你内部还原为实体再调用实体的行为

对外的领域服务接口，就是你领域提供的一些行为，直接提供出去就可以了

这是我比较推荐的一种DDD之间的具体落地的不同子域的集成模式

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/081~098资料/082\_对促销中心DDD建模进行代码落地（2）/笔记.docx**

《082\_对促销中心DDD建模进行代码落地（2）》

促销活动：促销中心的核心就是管理一个一个的促销活动，让运营部门去创建不同的促销活动，配合上一些促销活动的站内 or 站外的营销推广，进而拉动老用户的活跃度和复购率，或者是在站外进行拉新，每一种促销活动都包含了几个维度，包括了时间范围、人群范围、商品范围，等等

促销类型：这个其实我们之前就有过这个概念，促销是有很多种不同的类型的，我们认为完全是可以全部集成到促销这个概念里去的，那当然每一种促销活动都有一种促销类型，比如满减、买赠、直降、优惠券、秒杀，等等

促销规则：顾名思义，根据促销类型定义出来的具体促销规则，这个很灵活多变，之前我们都是考虑用JSON存储的

促销范围：每一个促销活动都会匹配一个促销范围，这个促销范围实际上是通过时间范围、人群范围、商品范围、平台范围（APP or 网站），等各种范围来确定的

候选促销活动查询：大家如果平时玩儿电商APP的话就知道，其实一般来说，促销活动会通过如下方式展现给你，要不然是在电商首页搞一些促销活动的专题页，要不然就是你浏览某个商品的时候，商品详情页里会提示促销活动，要不然就是你把商品加入购物车之后，在购物车浏览界面里可以看到每个商品参加的促销活动，当你提交订单之后，在订单确认页里，也会提示你每个商品参加的促销活动以及每个商品最终的价格，所以在电商的各个环节里，都需要去根据用户所属的人群范围、当前时间、浏览的商品范围，到促销中心查询对应的促销活动列表

优惠结果计算：这个一般是用于订单确认页面的，当你选购了一些商品进入订单确认页面时，如果大家平时玩儿淘宝可能会有印象，就是一个商品可能会同时参与多种促销活动，此时你可以在订单确认页里选择这个商品你最终要参加哪个促销活动，然后根据你选择的促销活动，促销中心可以提供优惠价格计算的服务，要不然是赠送商品给你，要不然是提示可以赠送你几个优惠券，要不然是提示你减免后的价格，甚至有时一个商品可以同时参与多个活动

促销活动优先顺序查询：上面说了，有的商品可能可以同时参加多种促销活动，那么在优惠结果计算的时候，需要按照不同促销活动的优先顺序来依次进行计算

促销活动互斥判断：当然一个商品可以同时参加多个促销活动，也可以设置让多个促销活动进行互斥，也就是说，商品要么选择这个促销活动，要么选择另外一个促销活动

其实分析清楚了以后，你会发现，促销中心的核心就是促销活动、促销类型、候选促销活动查询、优惠结果计算，然后促销活动优先顺序以及促销活动互斥都是依附于促销活动本身的一些关键属性，在优惠结果计算的时候会用到

到此为止，我们就把促销中心的核心关键术语以及定义都提炼出来了

在确定了这些关键术语之后，接下来很关键的一步，就是要确定这些关键术语分别术语什么领域对象？因为在DDD里，要不然是实体，要不然是值对象，实体是有唯一标识id的，值对象其实是没有唯一标识的，或者即使他在数据库里有唯一标识，但是这个唯一标识在业务上没有意义的

所以上面我们分析一下，明确能够发现，促销活动当然是实体了，因为不同的促销活动是有唯一标识的，这个在业务上都是有含义的，然后促销类型当然是值对象了，虽然促销类型可以在数据库里存储单独的表，也可以有数据库层面的ID主键，但是其实他在业务含义里，那个标识是没有意义的

同理的就是促销规则和促销范围，每一个促销范围在数据库里可以单独存一个表，但是其实他本质是促销活动的一个附属数据而已，即使有数据库id也是没有业务上的意义的

这里就可以确定一个聚合根了，那就是优惠活动是聚合根，里面包含了促销类型、促销规则、促销范围

然后对于候选促销活动查询、优惠结果计算、促销活动优先顺序查询、促销活动互斥判断，这些其实本质都是属于行为，并不是在实体/值对象的范围内的，如果是纯行为逻辑的一些东西，那么最好就是把他们建模为领域服务这个概念

说白了，就是把他们还是还原为类似以前的Service层那个概念

比如候选促销活动查询，可能对外还是提供一个接口出去；然后优惠结果计算也是提供一个接口出去，作为一个领域服务接口，就是营销域的服务接口，然后促销活动优先顺序查询、促销活动互斥判断可以设计为内部的领域服务Service，让优惠结果计算去调用

其实最终一个正规的DDD建模好的字域对外暴露的，应该是领域模型（实体模型、值对象模型）+ 对外的领域服务接口 + 对内的实体行为调用接口，实体模型/值对象模型是让其他子域进行实体行为调用的，然后实体引用内部封装了RPC调用，请求到你的实体行为调用接口，你内部还原为实体再调用实体的行为

对外的领域服务接口，就是你领域提供的一些行为，直接提供出去就可以了

这是我比较推荐的一种DDD之间的具体落地的不同子域的集成模式

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/081~098资料/081\_对促销中心DDD建模进行代码落地（1）/笔记.docx**

《081\_对促销中心DDD建模进行代码落地（1）》

促销活动：促销中心的核心就是管理一个一个的促销活动，让运营部门去创建不同的促销活动，配合上一些促销活动的站内 or 站外的营销推广，进而拉动老用户的活跃度和复购率，或者是在站外进行拉新，每一种促销活动都包含了几个维度，包括了时间范围、人群范围、商品范围，等等

促销类型：这个其实我们之前就有过这个概念，促销是有很多种不同的类型的，我们认为完全是可以全部集成到促销这个概念里去的，那当然每一种促销活动都有一种促销类型，比如满减、买赠、直降、优惠券、秒杀，等等

促销规则：顾名思义，根据促销类型定义出来的具体促销规则，这个很灵活多变，之前我们都是考虑用JSON存储的

促销范围：每一个促销活动都会匹配一个促销范围，这个促销范围实际上是通过时间范围、人群范围、商品范围、平台范围（APP or 网站），等各种范围来确定的

候选促销活动查询：大家如果平时玩儿电商APP的话就知道，其实一般来说，促销活动会通过如下方式展现给你，要不然是在电商首页搞一些促销活动的专题页，要不然就是你浏览某个商品的时候，商品详情页里会提示促销活动，要不然就是你把商品加入购物车之后，在购物车浏览界面里可以看到每个商品参加的促销活动，当你提交订单之后，在订单确认页里，也会提示你每个商品参加的促销活动以及每个商品最终的价格，所以在电商的各个环节里，都需要去根据用户所属的人群范围、当前时间、浏览的商品范围，到促销中心查询对应的促销活动列表

优惠结果计算：这个一般是用于订单确认页面的，当你选购了一些商品进入订单确认页面时，如果大家平时玩儿淘宝可能会有印象，就是一个商品可能会同时参与多种促销活动，此时你可以在订单确认页里选择这个商品你最终要参加哪个促销活动，然后根据你选择的促销活动，促销中心可以提供优惠价格计算的服务，要不然是赠送商品给你，要不然是提示可以赠送你几个优惠券，要不然是提示你减免后的价格，甚至有时一个商品可以同时参与多个活动

促销活动优先顺序查询：上面说了，有的商品可能可以同时参加多种促销活动，那么在优惠结果计算的时候，需要按照不同促销活动的优先顺序来依次进行计算

促销活动互斥判断：当然一个商品可以同时参加多个促销活动，也可以设置让多个促销活动进行互斥，也就是说，商品要么选择这个促销活动，要么选择另外一个促销活动

其实分析清楚了以后，你会发现，促销中心的核心就是促销活动、促销类型、候选促销活动查询、优惠结果计算，然后促销活动优先顺序以及促销活动互斥都是依附于促销活动本身的一些关键属性，在优惠结果计算的时候会用到

到此为止，我们就把促销中心的核心关键术语以及定义都提炼出来了

在确定了这些关键术语之后，接下来很关键的一步，就是要确定这些关键术语分别术语什么领域对象？因为在DDD里，要不然是实体，要不然是值对象，实体是有唯一标识id的，值对象其实是没有唯一标识的，或者即使他在数据库里有唯一标识，但是这个唯一标识在业务上没有意义的

所以上面我们分析一下，明确能够发现，促销活动当然是实体了，因为不同的促销活动是有唯一标识的，这个在业务上都是有含义的，然后促销类型当然是值对象了，虽然促销类型可以在数据库里存储单独的表，也可以有数据库层面的ID主键，但是其实他在业务含义里，那个标识是没有意义的

同理的就是促销规则和促销范围，每一个促销范围在数据库里可以单独存一个表，但是其实他本质是促销活动的一个附属数据而已，即使有数据库id也是没有业务上的意义的

这里就可以确定一个聚合根了，那就是优惠活动是聚合根，里面包含了促销类型、促销规则、促销范围

然后对于候选促销活动查询、优惠结果计算、促销活动优先顺序查询、促销活动互斥判断，这些其实本质都是属于行为，并不是在实体/值对象的范围内的，如果是纯行为逻辑的一些东西，那么最好就是把他们建模为领域服务这个概念

说白了，就是把他们还是还原为类似以前的Service层那个概念

比如候选促销活动查询，可能对外还是提供一个接口出去；然后优惠结果计算也是提供一个接口出去，作为一个领域服务接口，就是营销域的服务接口，然后促销活动优先顺序查询、促销活动互斥判断可以设计为内部的领域服务Service，让优惠结果计算去调用

其实最终一个正规的DDD建模好的字域对外暴露的，应该是领域模型（实体模型、值对象模型）+ 对外的领域服务接口 + 对内的实体行为调用接口，实体模型/值对象模型是让其他子域进行实体行为调用的，然后实体引用内部封装了RPC调用，请求到你的实体行为调用接口，你内部还原为实体再调用实体的行为

对外的领域服务接口，就是你领域提供的一些行为，直接提供出去就可以了

这是我比较推荐的一种DDD之间的具体落地的不同子域的集成模式

3616118202

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/081~098资料/090\_对促销中心DDD建模进行代码落地（10）/笔记.docx**

3616118202

《090\_对促销中心DDD建模进行代码落地（10）》

促销活动：促销中心的核心就是管理一个一个的促销活动，让运营部门去创建不同的促销活动，配合上一些促销活动的站内 or 站外的营销推广，进而拉动老用户的活跃度和复购率，或者是在站外进行拉新，每一种促销活动都包含了几个维度，包括了时间范围、人群范围、商品范围，等等

促销类型：这个其实我们之前就有过这个概念，促销是有很多种不同的类型的，我们认为完全是可以全部集成到促销这个概念里去的，那当然每一种促销活动都有一种促销类型，比如满减、买赠、直降、优惠券、秒杀，等等

促销规则：顾名思义，根据促销类型定义出来的具体促销规则，这个很灵活多变，之前我们都是考虑用JSON存储的

促销范围：每一个促销活动都会匹配一个促销范围，这个促销范围实际上是通过时间范围、人群范围、商品范围、平台范围（APP or 网站），等各种范围来确定的

候选促销活动查询：大家如果平时玩儿电商APP的话就知道，其实一般来说，促销活动会通过如下方式展现给你，要不然是在电商首页搞一些促销活动的专题页，要不然就是你浏览某个商品的时候，商品详情页里会提示促销活动，要不然就是你把商品加入购物车之后，在购物车浏览界面里可以看到每个商品参加的促销活动，当你提交订单之后，在订单确认页里，也会提示你每个商品参加的促销活动以及每个商品最终的价格，所以在电商的各个环节里，都需要去根据用户所属的人群范围、当前时间、浏览的商品范围，到促销中心查询对应的促销活动列表

优惠结果计算：这个一般是用于订单确认页面的，当你选购了一些商品进入订单确认页面时，如果大家平时玩儿淘宝可能会有印象，就是一个商品可能会同时参与多种促销活动，此时你可以在订单确认页里选择这个商品你最终要参加哪个促销活动，然后根据你选择的促销活动，促销中心可以提供优惠价格计算的服务，要不然是赠送商品给你，要不然是提示可以赠送你几个优惠券，要不然是提示你减免后的价格，甚至有时一个商品可以同时参与多个活动

促销活动优先顺序查询：上面说了，有的商品可能可以同时参加多种促销活动，那么在优惠结果计算的时候，需要按照不同促销活动的优先顺序来依次进行计算

促销活动互斥判断：当然一个商品可以同时参加多个促销活动，也可以设置让多个促销活动进行互斥，也就是说，商品要么选择这个促销活动，要么选择另外一个促销活动

其实分析清楚了以后，你会发现，促销中心的核心就是促销活动、促销类型、候选促销活动查询、优惠结果计算，然后促销活动优先顺序以及促销活动互斥都是依附于促销活动本身的一些关键属性，在优惠结果计算的时候会用到

到此为止，我们就把促销中心的核心关键术语以及定义都提炼出来了

在确定了这些关键术语之后，接下来很关键的一步，就是要确定这些关键术语分别术语什么领域对象？因为在DDD里，要不然是实体，要不然是值对象，实体是有唯一标识id的，值对象其实是没有唯一标识的，或者即使他在数据库里有唯一标识，但是这个唯一标识在业务上没有意义的

所以上面我们分析一下，明确能够发现，促销活动当然是实体了，因为不同的促销活动是有唯一标识的，这个在业务上都是有含义的，然后促销类型当然是值对象了，虽然促销类型可以在数据库里存储单独的表，也可以有数据库层面的ID主键，但是其实他在业务含义里，那个标识是没有意义的

同理的就是促销规则和促销范围，每一个促销范围在数据库里可以单独存一个表，但是其实他本质是促销活动的一个附属数据而已，即使有数据库id也是没有业务上的意义的

这里就可以确定一个聚合根了，那就是优惠活动是聚合根，里面包含了促销类型、促销规则、促销范围

然后对于候选促销活动查询、优惠结果计算、促销活动优先顺序查询、促销活动互斥判断，这些其实本质都是属于行为，并不是在实体/值对象的范围内的，如果是纯行为逻辑的一些东西，那么最好就是把他们建模为领域服务这个概念

说白了，就是把他们还是还原为类似以前的Service层那个概念

比如候选促销活动查询，可能对外还是提供一个接口出去；然后优惠结果计算也是提供一个接口出去，作为一个领域服务接口，就是营销域的服务接口，然后促销活动优先顺序查询、促销活动互斥判断可以设计为内部的领域服务Service，让优惠结果计算去调用

其实最终一个正规的DDD建模好的字域对外暴露的，应该是领域模型（实体模型、值对象模型）+ 对外的领域服务接口 + 对内的实体行为调用接口，实体模型/值对象模型是让其他子域进行实体行为调用的，然后实体引用内部封装了RPC调用，请求到你的实体行为调用接口，你内部还原为实体再调用实体的行为

对外的领域服务接口，就是你领域提供的一些行为，直接提供出去就可以了

这是我比较推荐的一种DDD之间的具体落地的不同子域的集成模式

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/001~010资料/004\_为什么现在去互联网大厂面试都要问DDD？/笔记.docx**

《004\_为什么现在去互联网大厂面试都要问DDD？》

DDD概念在国外已经有很多年很多年了，国外，我觉得老外有一点，确实值得我们去学习，老外确实有很多人多年就专注于一个领域，就把一个领域吃的特别的透，特别的深，很多专家就可以成长为世界顶尖的领域大牛

30年专门搞数据仓库，出了很多书，输出的都是自己的经验总结出来的系统体系和理论，指导所有人照着走就可以了；软件设计，DDD也是，比如有的专家，也不玩儿互联网高并发，redis内核，mongodb内核，高并发秒杀系统架构，人家未必懂的比你们多，人家就是专门搞软件设计，一搞就是30年

最终沉淀出来类似于DDD这样的很牛的一套业务软件设计的一整套理论和体系，可以去指导全世界，立志于就在某个领域深挖，你必须在那个领域的公司里从事一线工作，一直待项目，做项目，做20年以上，沉淀出那个领域属于你的一套理论体系和方法，甚至可以做出世界级顶尖的技术，理论，产品

DDD这个东西，我，少数人，对于DDD这个东西，是在很多年前就接触了，也经常会找机会自己设计系统去实践，最近两年才开始兴起，因为资料的匮乏，导致99.9%的人都是听过，了解一点，看资料也看不懂，盲目跟风，觉得DDD很牛，很流行

落地，很少有DDD落地成功的情况，顶级大厂里的大牛，少数人是真的懂DDD，也真的可以落地实战的

有一些同学出去面大厂的P7，P7都要问他DDD

国内的软件行业，大致也就20年的时间，90年代的软件很匮乏的，也是很少的，硬是算上，也可以，80年代也有一点点软件设计和开发，互联网真正作为一个大的行业有大的发展，也就10年的时间，2010年之后，移动互联网涌现之后

几十年，涌现了大量的专家，在各个细分领域里一干就是20年~30年的大叔，四五十岁的大叔，这帮人提炼出了各个细分领域里的很多理论体系和概念，极大的给软件、互联网这些行业的系统设计，树立了一个标准

国内最近一两年到了一个阶段，很多公司都在做复杂的业务系统，controller、service、dao、sql、数据库表，围绕数据库在做设计系统，开发系统，代码呈现出来跟真实世界里的系统的组件的关系和逻辑交互完全不一样，很多公司其实都在想办法改进这个现状，DDD软件设计的理论体系，就可以用武之地了

很多公司都在关注这个事情，也在尝试推动这个事情，很多公司的人都在探索和尝试这个事情，在招聘的时候也希望把这个东西作为一个考核点，希望招聘到对应的人才，这就是现在的一个现状

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/001~010资料/006\_让我们讲人话行吗：限界上下文和核心域是什么？/笔记.docx**

《006\_让我们讲人话行吗：限界上下文和核心域是什么？》

DDD里有很多的概念，核心域，子域，bounded context（限界上下文），通用语言，一大堆的概念，DDD的书籍和专栏，最大的问题就在于所，这些概念没有把他们用接地气方式来解释一下

电商阶段一里设计了一个小型的电商系统v1.0的雏形，划分成了很多的子系统，订单系统，商品系统，库存系统，WMS系统，营销系统，十几个子系统，一个限界上下文，其实就是圈定了你的一个大的系统（电商系统）里的一个小块的范围，订单系统就是一个限界上下文，商品系统就可以认为是一个限界上下文

就是给每个系统划分一个边界，把每个系统拆分出来，他们都有自己的领域和领地，并不是混合在一起的，一个限界上下文可以对应我们一个大系统（电商系统、广告系统、在线教育系统、IM系统、推荐系统、SaaS系统）里的一个子系统（订单系统，是几个人负责维护，他们共享一个订单系统这么一个限界上下文）

订单系统就有几十个人负责维护，生单系统（5个人负责维护）、价格系统（5个人）、拆单系统（5个人），对于一个完整的系统，可以在里面先划分出来多个限界上下文，每个限界上下文可以对应一个子系统，独立的小团队专门负责维护这个子系统以及对应的限界上下文

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/001~010资料/001\_中国的程序员都是如何设计乱七八糟的CRUD系统的？/笔记.docx**

《001\_中国的程序员都是如何设计乱七八糟的CRUD系统的？》

DDD的，电商阶段一，就是一个非常典型的软件设计的过程，他是以数据库模型为核心的一个系统，先设计数据库模型，包括数据库里的一些表以及表的字段，然后呢，在我们的java工程里，都是建立domain包，里面的每个类都是跟数据库的表一一对应的

controller、service、dao、mapper(SQL)，系统的核心，就是对数据库进行各种各样的crud的操作，通过大量的crud操作的组合，最终完成一个业务系统的所有功能的实现，乱七八糟的

你的一个系统按照上面的方法一直做，做到中后期，有大量的数据库表，有大量的controller、service、dao和Mapper（包含很多的SQL语句），这个系统，让任何一个新人来理解，都很艰难，这个系统并没有一个清晰的业务模型

完全不是按照真实的世界以及业务流程来设计的，模型也不是真实的，甚至，java是面向对象的语言，面向数据库的一个系统，他里面的各个类的交互，都不是按照真实的世界在进行交互，而是执行大量的crud

你的系统，可以去执行一个启动汽车这样的一个行为，按照真实世界的面相对象来设计的话，Driver -> Engine -> Car，面向数据库设计的系统来执行的话，Controller.startCar -> Service -> 调用了三个DAO，Driver表插入一条数据，engine表修改一条数据，car表修改一条数据，做了三个数据库更新操作

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/001~010资料/007\_让我们讲人话行吗：通用语言又是什么？/笔记.docx**

《007\_让我们讲人话行吗：通用语言又是什么？》

DDD里有很多的概念，核心域，子域，bounded context（限界上下文），通用语言，一大堆的概念，DDD的书籍和专栏，最大的问题就在于所，这些概念没有把他们用接地气方式来解释一下

电商阶段一里设计了一个小型的电商系统v1.0的雏形，划分成了很多的子系统，订单系统，商品系统，库存系统，WMS系统，营销系统，十几个子系统，一个限界上下文，其实就是圈定了你的一个大的系统（电商系统）里的一个小块的范围，订单系统就是一个限界上下文，商品系统就可以认为是一个限界上下文

就是给每个系统划分一个边界，把每个系统拆分出来，他们都有自己的领域和领地，并不是混合在一起的，一个限界上下文可以对应我们一个大系统（电商系统、广告系统、在线教育系统、IM系统、推荐系统、SaaS系统）里的一个子系统（订单系统，是几个人负责维护，他们共享一个订单系统这么一个限界上下文）

订单系统就有几十个人负责维护，生单系统（5个人负责维护）、价格系统（5个人）、拆单系统（5个人），对于一个完整的系统，可以在里面先划分出来多个限界上下文，每个限界上下文可以对应一个子系统，独立的小团队专门负责维护这个子系统以及对应的限界上下文

通用语言

电商系统v1.0来举个例子，订单是一块，wms是一块，营销是一块，采购是一块

订单系统里，有一个类，Customer，对订单系统这个限界上下文里，指的是下订单的用户；对于采购系统里，Customer，指的不是下订单的用户，是批发商进行批量采购的客户，同样是Customer，在不同的系统（限界上下文里），对应的含义是不同的

在不同的系统对应的限界上下文里，定义出来一批名词（一堆类），代表了这个限界上文里的通用语言，只有这个限界上下文里的几个系统的开发工程师，会按照同样的一批通用语言来理解所有的名词，大家对Customer这个名词的理解是一样

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/001~010资料/003\_目前国内的整体DDD水平大致是个什么样？/笔记.docx**

《003\_目前国内的整体DDD水平大致是个什么样？》

DDD概念在国外已经有很多年很多年了，国外，我觉得老外有一点，确实值得我们去学习，老外确实有很多人多年就专注于一个领域，就把一个领域吃的特别的透，特别的深，很多专家就可以成长为世界顶尖的领域大牛

30年专门搞数据仓库，出了很多书，输出的都是自己的经验总结出来的系统体系和理论，指导所有人照着走就可以了；软件设计，DDD也是，比如有的专家，也不玩儿互联网高并发，redis内核，mongodb内核，高并发秒杀系统架构，人家未必懂的比你们多，人家就是专门搞软件设计，一搞就是30年

最终沉淀出来类似于DDD这样的很牛的一套业务软件设计的一整套理论和体系，可以去指导全世界，立志于就在某个领域深挖，你必须在那个领域的公司里从事一线工作，一直待项目，做项目，做20年以上，沉淀出那个领域属于你的一套理论体系和方法，甚至可以做出世界级顶尖的技术，理论，产品

DDD这个东西，我，少数人，对于DDD这个东西，是在很多年前就接触了，也经常会找机会自己设计系统去实践，最近两年才开始兴起，因为资料的匮乏，导致99.9%的人都是听过，了解一点，看资料也看不懂，盲目跟风，觉得DDD很牛，很流行

落地，很少有DDD落地成功的情况，顶级大厂里的大牛，少数人是真的懂DDD，也真的可以落地实战的

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/001~010资料/010\_明确一个前提，DDD软件设计是不是非用不可？/笔记.docx**

《010\_明确一个前提，DDD软件设计是不是非用不可？》

三层架构、先设计数据库表结构、controller、service、dao、sql，这种以数据库为核心的软件设计的过程，他最大的好处在于速度快，如果你的系统不是特别的复杂，可能你的技术团队总共就20人以内，小团队，做小系统，拆分成10个左右的服务，业务逻辑也没那么的复杂，此时用数据库为核心的软件设计方法，快

那需要你的团队对DDD都有一定的了解和理解，因为你作为架构师，你需要做DDD业务建模和顶层设计，你的团队成员必须理解DDD的概念和你的设计，能够按照你的设计去进行具体的开发，中间不要走样

新系统，业务比较复杂，团队成员不要太多，你培训培训，三五个人能理解，大家一起搞了，从0开始，大家一起商讨限界上下文、通用语言、子域，大家一起做，一起开发，一起实践，体验DDD设计出来的系统的样子

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/001~010资料/008\_如果没有限界上下文和通用语言，为什么会一团糟？/笔记.docx**

《008\_如果没有限界上下文和通用语言，为什么会一团糟？》

DDD里有很多的概念，核心域，子域，bounded context（限界上下文），通用语言，一大堆的概念，DDD的书籍和专栏，最大的问题就在于所，这些概念没有把他们用接地气方式来解释一下

电商阶段一里设计了一个小型的电商系统v1.0的雏形，划分成了很多的子系统，订单系统，商品系统，库存系统，WMS系统，营销系统，十几个子系统，一个限界上下文，其实就是圈定了你的一个大的系统（电商系统）里的一个小块的范围，订单系统就是一个限界上下文，商品系统就可以认为是一个限界上下文

就是给每个系统划分一个边界，把每个系统拆分出来，他们都有自己的领域和领地，并不是混合在一起的，一个限界上下文可以对应我们一个大系统（电商系统、广告系统、在线教育系统、IM系统、推荐系统、SaaS系统）里的一个子系统（订单系统，是几个人负责维护，他们共享一个订单系统这么一个限界上下文）

订单系统就有几十个人负责维护，生单系统（5个人负责维护）、价格系统（5个人）、拆单系统（5个人），对于一个完整的系统，可以在里面先划分出来多个限界上下文，每个限界上下文可以对应一个子系统，独立的小团队专门负责维护这个子系统以及对应的限界上下文

通用语言

电商系统v1.0来举个例子，订单是一块，wms是一块，营销是一块，采购是一块

订单系统里，有一个类，Customer，对订单系统这个限界上下文里，指的是下订单的用户；对于采购系统里，Customer，指的不是下订单的用户，是批发商进行批量采购的客户，同样是Customer，在不同的系统（限界上下文里），对应的含义是不同的

在不同的系统对应的限界上下文里，定义出来一批名词（一堆类），代表了这个限界上文里的通用语言，只有这个限界上下文里的几个系统的开发工程师，会按照同样的一批通用语言来理解所有的名词，大家对Customer这个名词的理解是一样

假设你有一个电商系统，就是一个单块系统，几十个人一起维护，然后在com.zhss.eshop.order.domain里有一个类，是Customer；com.zhss.eshop.purchase.domain有一个类，是Customer，同样的类名在不同的包下面

拆分出来限界上下文，也有可能，这个限界上下文对应的是一个单块系统里面的一部分包，order相关的包下面的代码都属于一个订单相关的限界上下文，一个限界上下文是独立的系统，有独立的代码仓库，独立的工程，独立的数据库，独立的测试环境，独立的维护团队，完全独立开来的，有自己内部的一套通用语言

负责采购的人可以去修改订单的代码，因为他完全不理解订单那个限界上下文里的通用语言，对Customer都理解错误了，是不能去修改代码的

N多年之前，我都还没接触过DDD的时候，这个道理我就懂，可以拆分为多个子系统，独立的团队去负责维护，系统内部有自己的一套名词和类体系，有自己独立的含义

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/001~010资料/002\_为什么我们需要使用DDD来设计复杂业务系统？/笔记.docx**

《002\_为什么我们需要使用DDD来设计复杂业务系统？》

DDD的，电商阶段一，就是一个非常典型的软件设计的过程，他是以数据库模型为核心的一个系统，先设计数据库模型，包括数据库里的一些表以及表的字段，然后呢，在我们的java工程里，都是建立domain包，里面的每个类都是跟数据库的表一一对应的

controller、service、dao、mapper(SQL)，系统的核心，就是对数据库进行各种各样的crud的操作，通过大量的crud操作的组合，最终完成一个业务系统的所有功能的实现，乱七八糟的

你的一个系统按照上面的方法一直做，做到中后期，有大量的数据库表，有大量的controller、service、dao和Mapper（包含很多的SQL语句），这个系统，让任何一个新人来理解，都很艰难，这个系统并没有一个清晰的业务模型

完全不是按照真实的世界以及业务流程来设计的，模型也不是真实的，甚至，java是面向对象的语言，面向数据库的一个系统，他里面的各个类的交互，都不是按照真实的世界在进行交互，而是执行大量的crud

你的系统，可以去执行一个启动汽车这样的一个行为，按照真实世界的面相对象来设计的话，Driver -> Engine -> Car，面向数据库设计的系统来执行的话，Controller.startCar -> Service -> 调用了三个DAO，Driver表插入一条数据，engine表修改一条数据，car表修改一条数据，做了三个数据库更新操作

理想中的系统，应该Driver -> Engine -> Car在交互协作，CarOperateSystem，对外暴露接口的是这么一个系统，用于操作汽车的这么一个系统，他提供了很多的功能和按钮，你执行不同的操作，都会去找这个CarOperateSystem

这个操作系统底层应该是封装了很多的组件，完全模拟出了真实世界的运行的过程，构造出来一个Driver的概念，Engine，Car，Wheels，系统跑的时候，应该是把真实世界的所有的构件模拟出来

即使是真实世界里没有的一些组件和概念，我们也希望把一些抽象的概念给模拟出来，而不是说都是Controller、Service、DAO、SQL

针对一些复杂的业务系统，很多国外的软件设计的专家，不要老是三层模型，controller，service，DAO，建立了一个整套的软件设计的思想和体系，DDD，领域驱动模型设计，用DDD希望做到什么呢？可以把你的软件系统里的所有的类和代码组件，从controller、service、dao和SQL里解放出来

设计出来的类以及类之间的交互流程，应该是尽可能模拟真实世界的一个情况，任何人过来看这个代码，很容易就可以理解了，这个软件因为模拟了真实世界运行的一个情况，所以哪怕是后续长期维护，人脑理解起来都非常的容易

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/001~010资料/005\_现在国内程序员学习DDD设计思想的痛点在哪儿？/笔记.docx**

《005\_现在国内程序员学习DDD设计思想的痛点在哪儿？》

DDD概念在国外已经有很多年很多年了，国外，我觉得老外有一点，确实值得我们去学习，老外确实有很多人多年就专注于一个领域，就把一个领域吃的特别的透，特别的深，很多专家就可以成长为世界顶尖的领域大牛

30年专门搞数据仓库，出了很多书，输出的都是自己的经验总结出来的系统体系和理论，指导所有人照着走就可以了；软件设计，DDD也是，比如有的专家，也不玩儿互联网高并发，redis内核，mongodb内核，高并发秒杀系统架构，人家未必懂的比你们多，人家就是专门搞软件设计，一搞就是30年

最终沉淀出来类似于DDD这样的很牛的一套业务软件设计的一整套理论和体系，可以去指导全世界，立志于就在某个领域深挖，你必须在那个领域的公司里从事一线工作，一直待项目，做项目，做20年以上，沉淀出那个领域属于你的一套理论体系和方法，甚至可以做出世界级顶尖的技术，理论，产品

DDD这个东西，我，少数人，对于DDD这个东西，是在很多年前就接触了，也经常会找机会自己设计系统去实践，最近两年才开始兴起，因为资料的匮乏，导致99.9%的人都是听过，了解一点，看资料也看不懂，盲目跟风，觉得DDD很牛，很流行

落地，很少有DDD落地成功的情况，顶级大厂里的大牛，少数人是真的懂DDD，也真的可以落地实战的

有一些同学出去面大厂的P7，P7都要问他DDD

国内的软件行业，大致也就20年的时间，90年代的软件很匮乏的，也是很少的，硬是算上，也可以，80年代也有一点点软件设计和开发，互联网真正作为一个大的行业有大的发展，也就10年的时间，2010年之后，移动互联网涌现之后

几十年，涌现了大量的专家，在各个细分领域里一干就是20年~30年的大叔，四五十岁的大叔，这帮人提炼出了各个细分领域里的很多理论体系和概念，极大的给软件、互联网这些行业的系统设计，树立了一个标准

国内最近一两年到了一个阶段，很多公司都在做复杂的业务系统，controller、service、dao、sql、数据库表，围绕数据库在做设计系统，开发系统，代码呈现出来跟真实世界里的系统的组件的关系和逻辑交互完全不一样，很多公司其实都在想办法改进这个现状，DDD软件设计的理论体系，就可以用武之地了

很多公司都在关注这个事情，也在尝试推动这个事情，很多公司的人都在探索和尝试这个事情，在招聘的时候也希望把这个东西作为一个考核点，希望招聘到对应的人才，这就是现在的一个现状

国外讲DDD的书籍，翻译成中文的，就两三本，都极为的晦涩，你去看可能都看不懂，语言非常深奥，也不大白话，也不好懂；国内确实有一些知识付费的平台出了一两个DDD的专栏，按照那种专栏的讲法，我觉得还是不够接地气

除此之外，国内的程序员想要学习DDD，就再也没有更多的更好的资料了

课程来说，必须得加入DDD这个东西，都是为了以后可以进大厂去做P7，DDD未来两年几乎是必备的一个东西，都要可以设计复杂的业务系统，按照老外写的书的那种讲法来一步一步讲，大家会听的很痛苦

不愿意讲，按照那种讲法，完全就是不说人话，上来就给你讲一大堆的DDD领域的专业术语，不跟你把这个专业术语跟实际的软件开发过程中的一些概念有一个匹配，软件工程，是 我第一个大白话讲的比较好

r

# **File Path: /Users/xiaotingting/Downloads/学习资料/16\_专治复杂业务场景的领域驱动模型设计unzip/001~010资料/009\_在国内的公司里，哪些人是DDD业务专家？/笔记.docx**

《009\_在国内的公司里，哪些人是DDD业务专家？》

DDD里有很多的概念，核心域，子域，bounded context（限界上下文），通用语言，一大堆的概念，DDD的书籍和专栏，最大的问题就在于所，这些概念没有把他们用接地气方式来解释一下

电商阶段一里设计了一个小型的电商系统v1.0的雏形，划分成了很多的子系统，订单系统，商品系统，库存系统，WMS系统，营销系统，十几个子系统，一个限界上下文，其实就是圈定了你的一个大的系统（电商系统）里的一个小块的范围，订单系统就是一个限界上下文，商品系统就可以认为是一个限界上下文

就是给每个系统划分一个边界，把每个系统拆分出来，他们都有自己的领域和领地，并不是混合在一起的，一个限界上下文可以对应我们一个大系统（电商系统、广告系统、在线教育系统、IM系统、推荐系统、SaaS系统）里的一个子系统（订单系统，是几个人负责维护，他们共享一个订单系统这么一个限界上下文）

订单系统就有几十个人负责维护，生单系统（5个人负责维护）、价格系统（5个人）、拆单系统（5个人），对于一个完整的系统，可以在里面先划分出来多个限界上下文，每个限界上下文可以对应一个子系统，独立的小团队专门负责维护这个子系统以及对应的限界上下文

通用语言

电商系统v1.0来举个例子，订单是一块，wms是一块，营销是一块，采购是一块

订单系统里，有一个类，Customer，对订单系统这个限界上下文里，指的是下订单的用户；对于采购系统里，Customer，指的不是下订单的用户，是批发商进行批量采购的客户，同样是Customer，在不同的系统（限界上下文里），对应的含义是不同的

在不同的系统对应的限界上下文里，定义出来一批名词（一堆类），代表了这个限界上文里的通用语言，只有这个限界上下文里的几个系统的开发工程师，会按照同样的一批通用语言来理解所有的名词，大家对Customer这个名词的理解是一样

假设你有一个电商系统，就是一个单块系统，几十个人一起维护，然后在com.zhss.eshop.order.domain里有一个类，是Customer；com.zhss.eshop.purchase.domain有一个类，是Customer，同样的类名在不同的包下面

拆分出来限界上下文，也有可能，这个限界上下文对应的是一个单块系统里面的一部分包，order相关的包下面的代码都属于一个订单相关的限界上下文，一个限界上下文是独立的系统，有独立的代码仓库，独立的工程，独立的数据库，独立的测试环境，独立的维护团队，完全独立开来的，有自己内部的一套通用语言

负责采购的人可以去修改订单的代码，因为他完全不理解订单那个限界上下文里的通用语言，对Customer都理解错误了，是不能去修改代码的

N多年之前，我都还没接触过DDD的时候，这个道理我就懂，可以拆分为多个子系统，独立的团队去负责维护，系统内部有自己的一套名词和类体系，有自己独立的含义

设计复杂系统的时候，得有业务领域专家，业务领域专家可以根据真实世界里的对象和交互流程，去定义你的系统应该包含哪些限界上下文，每个限界上下文里有哪些通用语言，这个也是业务专家去定义出来的

替代掉这个所谓的业务专家的，PM，产品经理，产品经理去调研用户需求，无论是C端还是B端，设计产品，画一些原型图，写产品需求文档，他对于限界上下文，通用语言，完全没概念

真正应该替代国外的业务领域专家的人，就是你，负责一个大项目的技术leader，架构师，系统负责人，懂技术，懂DDD的理论体系和思想，从PM那里理解了系统需求之后，由你抽象和总结，设计出来一套符合DDD理论的限界上下文，通用语言，按照DDD理论对系统进行了业务建模之后

接着才会进入到技术设计里去，数据库里的表设计，是根据你的DDD业务建模的结果来的，不是上来先设计数据库表，反过来根据表设计类；先设计DDD业务模型、划分系统、设计类、设计表、写代码实现符合DDD的系统交互流程