05 | 聚合和聚合根：怎样设计聚合？

2019-10-23 欧创新

DDD实战课

进入课程 >

讲述：欧创新

时长 13:19　大小 10.68M

你好，我是欧创新。今天我们来学习聚合（Aggregate）和聚合根（AggregateRoot）。

我们先回顾下上一讲，在事件风暴中，我们会根据一些业务操作和行为找出实体（Entity）或值对象（ValueObject），进而将业务关联紧密的实体和值对象进行组合，构成聚合，再根据业务语义将多个聚合划定到同一个限界上下文（Bounded Context）中，并在限界上下文内完成领域建模。

那你知道为什么要在限界上下文和实体之间增加聚合和聚合根这两个概念吗？它们的作用是什么？怎么设计聚合？这就是我们这一讲重点要关注的问题。

聚合

在 DDD 中，实体和值对象是很基础的领域对象。实体一般对应业务对象，它具有业务属性和业务行为；而值对象主要是属性集合，对实体的状态和特征进行描述。但实体和值对象都只是个体化的对象，它们的行为表现出来的是个体的能力。

那聚合在其中起什么作用呢?

举个例子。社会是由一个个的个体组成的，象征着我们每一个人。随着社会的发展，慢慢出现了社团、机构、部门等组织，我们开始从个人变成了组织的一员，大家可以协同一致的工作，朝着一个最大的目标前进，发挥出更大的力量。

领域模型内的实体和值对象就好比个体，而能让实体和值对象协同工作的组织就是聚合，它用来确保这些领域对象在实现共同的业务逻辑时，能保证数据的一致性。

你可以这么理解，聚合就是由业务和逻辑紧密关联的实体和值对象组合而成的，聚合是数据修改和持久化的基本单元，每一个聚合对应一个仓储，实现数据的持久化。

聚合有一个聚合根和上下文边界，这个边界根据业务单一职责和高内聚原则，定义了聚合内部应该包含哪些实体和值对象，而聚合之间的边界是松耦合的。按照这种方式设计出来的微服务很自然就是“高内聚、低耦合”的。

聚合在 DDD 分层架构里属于领域层，领域层包含了多个聚合，共同实现核心业务逻辑。聚合内实体以充血模型实现个体业务能力，以及业务逻辑的高内聚。跨多个实体的业务逻辑通过领域服务来实现，跨多个聚合的业务逻辑通过应用服务来实现。比如有的业务场景需要同一个聚合的 A 和 B 两个实体来共同完成，我们就可以将这段业务逻辑用领域服务来实现；而有的业务逻辑需要聚合 C 和聚合 D 中的两个服务共同完成，这时你就可以用应用服务来组合这两个服务。

聚合根

聚合根的主要目的是为了避免由于复杂数据模型缺少统一的业务规则控制，而导致聚合、实体之间数据不一致性的问题。

传统数据模型中的每一个实体都是对等的，如果任由实体进行无控制地调用和数据修改，很可能会导致实体之间数据逻辑的不一致。而如果采用锁的方式则会增加软件的复杂度，也会降低系统的性能。

如果把聚合比作组织，那聚合根就是这个组织的负责人。聚合根也称为根实体，它不仅是实体，还是聚合的管理者。

首先它作为实体本身，拥有实体的属性和业务行为，实现自身的业务逻辑。

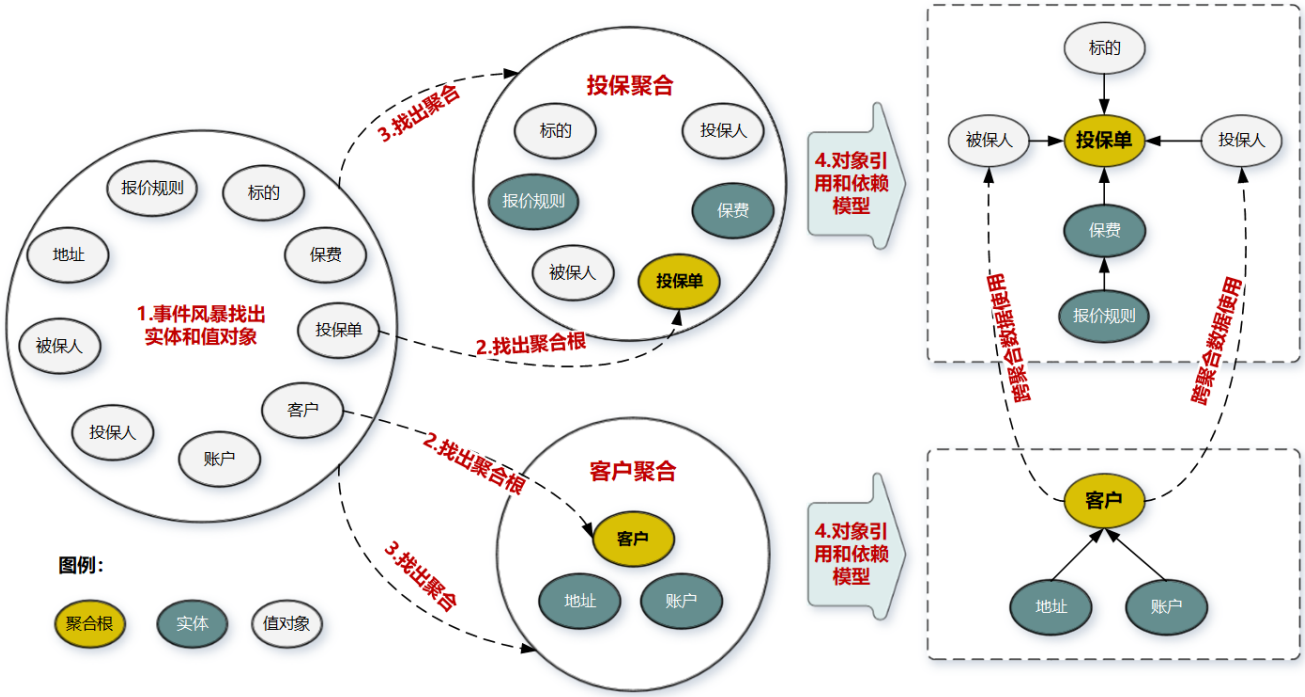
其次它作为聚合的管理者，在聚合内部负责协调实体和值对象按照固定的业务规则协同完成共同的业务逻辑。

最后在聚合之间，它还是聚合对外的接口人，以聚合根 ID 关联的方式接受外部任务和请求，在上下文内实现聚合之间的业务协同。也就是说，聚合之间通过聚合根 ID 关联引用，如果需要访问其它聚合的实体，就要先访问聚合根，再导航到聚合内部实体，外部对象不能直接访问聚合内实体。

怎样设计聚合？

DDD 领域建模通常采用事件风暴，它通常采用用例分析、场景分析和用户旅程分析等方法，通过头脑风暴列出所有可能的业务行为和事件，然后找出产生这些行为的领域对象，并梳理领域对象之间的关系，找出聚合根，找出与聚合根业务紧密关联的实体和值对象，再将聚合根、实体和值对象组合，构建聚合。

下面我们以保险的投保业务场景为例，看一下聚合的构建过程主要都包括哪些步骤。



第 1 步：采用事件风暴，根据业务行为，梳理出在投保过程中发生这些行为的所有的实体和值对象，比如投保单、标的、客户、被保人等等。

第 2 步：从众多实体中选出适合作为对象管理者的根实体，也就是聚合根。判断一个实体是否是聚合根，你可以结合以下场景分析：是否有独立的生命周期？是否有全局唯一 ID？是否可以创建或修改其它对象？是否有专门的模块来管这个实体。图中的聚合根分别是投保单和客户实体。

第 3 步：根据业务单一职责和高内聚原则，找出与聚合根关联的所有紧密依赖的实体和值对象。构建出 1 个包含聚合根（唯一）、多个实体和值对象的对象集合，这个集合就是聚合。在图中我们构建了客户和投保这两个聚合。

第 4 步：在聚合内根据聚合根、实体和值对象的依赖关系，画出对象的引用和依赖模型。这里我需要说明一下：投保人和被保人的数据，是通过关联客户 ID 从客户聚合中获取的，在投保聚合里它们是投保单的值对象，这些值对象的数据是客户的冗余数据，即使未来客户聚合的数据发生了变更，也不会影响投保单的值对象数据。从图中我们还可以看出实体之间的引用关系，比如在投保聚合里投保单聚合根引用了报价单实体，报价单实体则引用了报价规则子实体。

第 5 步：多个聚合根据业务语义和上下文一起划分到同一个限界上下文内。

这就是一个聚合诞生的完整过程了。

聚合的一些设计原则

我们不妨先看一下《实现领域驱动设计》一书中对聚合设计原则的描述，原文是有点不太好理解的，我来给你解释一下。

1. 在一致性边界内建模真正的不变条件。聚合用来封装真正的不变性，而不是简单地将对象组合在一起。聚合内有一套不变的业务规则，各实体和值对象按照统一的业务规则运行，实现对象数据的一致性，边界之外的任何东西都与该聚合无关，这就是聚合能实现业务高内聚的原因。

2. 设计小聚合。如果聚合设计得过大，聚合会因为包含过多的实体，导致实体之间的管理过于复杂，高频操作时会出现并发冲突或者数据库锁，最终导致系统可用性变差。而小聚合

设计则可以降低由于业务过大导致聚合重构的可能性，让领域模型更能适应业务的变化。

3. 通过唯一标识引用其它聚合。聚合之间是通过关联外部聚合根 ID 的方式引用，而不是直接对象引用的方式。外部聚合的对象放在聚合边界内管理，容易导致聚合的边界不清晰，也会增加聚合之间的耦合度。

4. 在边界之外使用最终一致性。聚合内数据强一致性，而聚合之间数据最终一致性。在一次事务中，最多只能更改一个聚合的状态。如果一次业务操作涉及多个聚合状态的更改，应采用领域事件的方式异步修改相关的聚合，实现聚合之间的解耦（相关内容我会在领域事件部分详解）。

5. 通过应用层实现跨聚合的服务调用。为实现微服务内聚合之间的解耦，以及未来以聚合为单位的微服务组合和拆分，应避免跨聚合的领域服务调用和跨聚合的数据库表关联。

上面的这些原则是 DDD 的一些通用的设计原则，还是那句话：“适合自己的才是最好的。”在系统设计过程时，你一定要考虑项目的具体情况，如果面临使用的便利性、高性能要求、技术能力缺失和全局事务管理等影响因素，这些原则也并不是不能突破的，总之一切以解决实际问题为出发点。

总结

[第 04 讲] 和 [第 05 讲] 的内容，其实是有强关联的。我们不妨在这里总结下聚合、聚合根、实体和值对象它们之间的联系和区别。

聚合的特点：高内聚、低耦合，它是领域模型中最底层的边界，可以作为拆分微服务的最小单位，但我不建议你对微服务过度拆分。但在对性能有极致要求的场景中，聚合可以独立作为一个微服务，以满足版本的高频发布和极致的弹性伸缩能力。

一个微服务可以包含多个聚合，聚合之间的边界是微服务内天然的逻辑边界。有了这个逻辑边界，在微服务架构演进时就可以以聚合为单位进行拆分和组合了，微服务的架构演进也就不再是一件难事了。

聚合根的特点：聚合根是实体，有实体的特点，具有全局唯一标识，有独立的生命周期。一个聚合只有一个聚合根，聚合根在聚合内对实体和值对象采用直接对象引用的方式进行组织和协调，聚合根与聚合根之间通过 ID 关联的方式实现聚合之间的协同。

实体的特点：有 ID 标识，通过 ID 判断相等性，ID 在聚合内唯一即可。状态可变，它依附于聚合根，其生命周期由聚合根管理。实体一般会持久化，但与数据库持久化对象不一定是一对一的关系。实体可以引用聚合内的聚合根、实体和值对象。

值对象的特点：无 ID，不可变，无生命周期，用完即扔。值对象之间通过属性值判断相等性。它的核心本质是值，是一组概念完整的属性组成的集合，用于描述实体的状态和特征。值对象尽量只引用值对象。

思考题

请你结合公司的某个业务场景，试试能分析出哪些聚合？

欢迎留言和我分享你的思考，你也可以把今天所学分享给身边的朋友，邀请他加入探讨，共同进阶。

上一篇　04 | 实体和值对象：从领域模型的基础单元看系统设计

下一篇　06 | 领域事件：解耦微服务的关键

精选留言 (20)

写留言

陈华应 置顶

2019-10-23

老师，麻烦有空帮忙看一下
场景：电销
根据任务类型的属性创建具体的定期执行任务，调度器把到点的任务放到执行器里去执行。执行完了等待下一次执行，对任务生成的明细可以填写沟通记录（第三方服务）但是本服务要提供此字段查询...

展开

作者回复: 我先理解一下你说的业务场景哈。不知道理解的对不对？不对的地方请你指出。由于不好展示事件风暴的过程，我就口述吧。
你描述的业务场景主要包括这两个部分吧。
流程一：创建任务
1、根据任务规则获取任务基础数据，生成任务。（产生任务已创建事件，领域对象包括：任务生成的基础数据、数据过滤规则、任务、任务类型）
2、自动将任务分配给销售。（销售的数据应该来源于其他系统，这个过程实际上是一个给任务赋值的过程，领域对象包括：销售）
3、销售领取任务。（给任务分配销售）
这个过程领域对象包括：基础数据、数据过滤规则、任务、任务类型、销售。
命令有：创建任务，给任务分配销售。
领域事件有：任务已创建。

流程二、任务执行
1、销售查询并获取任务，执行任务。（不清楚你说的字段在这个过程是什么含义，是查询时勾选类型吗）
2、任务执行完成后，记录执行结果，产生任务执行日志。（产生任务执行日志已创建事件，领域对象有：任务、任务日志）
3、统计日志。这一块不清楚你的业务逻辑和流程。
因为有统计日志，是不是就会去查询任务的执行日志，如果是这样的话，任务日志就需要设计为实体，跟任务关联，这里任务是聚合根。
这个阶段的领域对象有：任务和任务执行日志。
命令有：查询任务，生成任务执行日志。
领域事件有：执行任务日志已创建。

结合这两个流程，我们整体来分析一下。
领域对象包括：基础数据、数据过滤规则、任务、销售、任务类型、任务执行日志。
由于销售人员数据来源于第三方，以值的形式存储在任务中，因此我们可以将它设计为任务的值对象。
任务执行日志依附于任务，但是由于它后续要做查询和统计分析，因此将它设计为被任务引用的

实体。
任务类型是任务的值对象
其它的基础数据、数据过滤规则这两个领域对象很独立，你可以理解他们是独立的实体，或者说一个实体就是一个聚合。
但是这样设计在代码目录设计时会显得比较单薄，一个聚合会有一个仓储和聚合自己的代码目录结构。因此我们可以将这两个实体可以直接放在任务的聚合里，但是他们的生命周期不受任务这个聚合根管理。
这样的话，我们就可以只建立一个任务聚合。这个聚合的聚合根是任务。它引用的实体包括任务执行日志，任务的值对象有：销售、任务类型。还有两个独立实体：基础数据和数据过滤规则。
说明一下：
在不少的数据统计和计算场景中，有很多实体之间相互独立，只参与计算和统计分析，但是这类场景中业务内聚性又很高，你找不出管理这些实体的聚合根。我称这种业务模型是非典型领域模型。虽然有些方面（比如聚合根）不符合DDD的一些原则，但是我们也可以按照DDD方法来完成设计。

4 6

渊虹
2019-10-24

老师，有个问题不明白，麻烦解惑。投保聚合和客户聚合中，投保人和被保人跨聚合引用到客户的id，需求是查询以客户s为被保人的保单。就需要跨过聚合根，直接访问被保人这个值对象。这个是不是和只能通过聚合根访问聚合内其他对象的理论不一致

展开

作者回复: 被保人这个值对象以属性嵌入的方式嵌入保单聚合根中，查询客户保单时你不需要到客户聚合去查询客户信息了，直接根据客户信息在投保聚合查保单就可以了，当然这个客户信息不只是ID。
说明一下：这个跨聚合引用是在生成保单的时候，通过客户聚合根查询获取的客户信息，从客户聚合获取客户信息后，客户的信息就作为值对象的值嵌入到了保单实体中。

1

密码123456
2019-10-23

聚合。用界限上下文把细粒度的实体圈起来当做一个组织，选出组织的董事长。一个组织和另外一个组织交流的时候，只需要通过一个董事长，能够了解该组织的全部非隐私信息

作者回复: 类比的不错。

杨杰
2019-10-23

“一个微服务可以包含多个聚合，聚合之间的边界是微服务内天然的逻辑边界。有了这个逻辑边界，在微服务架构演进时就可以以聚合为单位进行拆分和组合了，微服务的架构演进也就不再是一件难事了“

目前我们在微服务内部，对聚合的要求降低了。也就是说一个微服务内部的数据结构是随便互相访问的，实体也是贫血的模型。主要是考虑：服务层已经分为微服务层，...

展开

作者回复: 因为微服务的架构演进，会有功能和代码的拆分和重构的过程。而一般来讲聚合的内聚性很高，聚合内的功能相对稳定，我们可以聚合功能和代码为单位在不同的微服务之间进行功能和代码的重构。
如果聚合之间代码和业务边界不清晰，聚合之间数据和服务可以随便访问，就会因为耦合度过高，代码很难剥离，最后微服务架构演进时又要在走一遍从单体拆分微服务的过程。后面的章节我会专门讲微服务架构的演进。

蜗牛慢慢爬
2019-10-23

听了这么多节课，总结一下还是太抽象了

展开

作者回复: 基础篇主要讲解DDD的基础概念和设计理念，所以相对抽象一些。后面会有中台业务建模和微服务设计案例，比较好理解。敬请期待。

守候、
2019-10-23

老师，就基础篇。我的理解是基于事件风暴（头脑风暴）定义出实体与值对象，并能识别出根对象，进而得到聚合，并清晰领域边界、松耦合。从而达到服务化确定服务的边界！但是这其中事件风暴如何组织进行是否有策略或者说工具技术。如果团队成员就事件风暴无法达成共识，如何进一步推进？

展开

作者回复: 你好，事件风暴会有专门的一节介绍。

墨名次
2019-10-26

老师，有个问题不明白：值对象的不可变指的是什么？拿上一章的Person跟Address来说，Address是值对象，对于Address来说，它的不可变指的是它的属性名称不可变？还是属性值不可变？或者是它里面的属于数量不可变？

展开

作者回复: 值对象可以整体替换，但不能对里面的属性数据做局部修改。你可以这么理解，值对象是一个字段，但不同的是值对象里面还有有很多其它属性。数据库里面你可修改的最小单元是字段，你不可以对这个字段的具备修改，你只能整体修改字段的内容。

liuchangit
2019-10-26

领域驱动设计里介绍，只为聚合根提供repository，想问下老师，在一个聚合内所有实体和值对象的存取（如文中投保单聚合内的对象），都由这个repository来负责数据库交互吗？感觉常见的orm框架似乎都做不到这一点。在DDD实践中，这个问题一般怎么解决呢？

展开

作者回复: 据我所知，现在的java框架对orm支持得不够好。而nosql的方式可能支持的更好一些。所以这一块的设计尽量结合自己的技术和业务场景，如果不会在聚合内产生数据不一致的情况，在技术不具备的时候，咱们没必要为了DDD而做DDD。

清涧飞鸟
2019-10-25

老师，我有两个问题：
1界限上下文代码中的体现是个包？
2聚合根可以是一个具体的实体吗？

展开

作者回复: 限界上下文如果被被设计为一个微服务的话，它的代码就是一个微服务发布包。理论上一个限界上下文就可以设计为一个微服务，但它还受其它外部因素限制，比如技术异构，团队沟

通成本，版本发布频率，性能等因素。
聚会根就是一个实体。不过它是一个具有管理功能的特殊实体。

夙梦流尘
2019-10-25

A聚合的applicationService可以load B聚合，使用B聚合的值对象吗

作者回复: 应用服务是在聚合之上的，它不是单独属于某个聚合的。在应用服务内聚合根可以引用其它聚合的聚合根，读取或者修改其它聚合的实体。

心浮天空
2019-10-25

本章理解：
聚合是将领域中一组协作关系密切的实体和值对象组织在一起并找出聚合根的过程，换句话说，聚合根就是聚合过程的产物。聚合根是这一组实体和值对象的管理入口，由聚合根通过组织实体和值对象的关系来完成实际业务，而聚合根的业务行为可以看作是最小的事务单元，在这个事务单元内，要求数据的强一致性，不同聚合根之间的协作只能通过应...
展开

作者回复: 1、聚合根主要协调聚合内的实体和值对象，通过引用的方式是从数据的角度来保证数据的一致性。而领域服务主要是从业务行为，通过实体属性和实体方法来进行业务逻辑的组合和编排，多个实体协作完成复杂的业务逻辑。
2、客户信息修改后，是不能影响保单中的客户数据的，这个客户数据是跟单数据，不会随着客户实体数据的变更而变化。举个收货地址的例子，你在维护你个人中心的多个收货地址后，不会修改已经发货的订单上的地址一样的。除非你提出变更收货地址。
3、一般来说应用层不应该有自己的实体对象，它引用领域层的对象，不实现复杂的业务逻辑。通过对不同聚合的领域服务组合和编排，实现跨聚合的业务协作。
4、在微服务内只有一个应用层。在微服务内尽量避免聚合之间的直接交互。聚合之间的交互都通过应用层。如果设计时将某个聚合放在了不合适的限界上下文内，以后在聚合拆分和合并时，由于聚合之间耦合度低，微服务的演进也就容易的多。

约书亚
2019-10-24

这一节解决了我长久的很多困惑，谢谢。比如一致性/不变性这点，在不基于DDD做设计时，一直在思考，在多个实体之上时长会存在一个约束，可这个约束应该放在哪里来保证呢？现在有答案了。
问题：是不是要想实现聚合根，在代码实现时就对orm框架的选型有一定要求了，比如怎么也要满足延迟加载，更新追踪这些特性，而一些轻量级的sql框架则不能使用了，因为...
展开

作者回复: 是的，感觉现在JAVA的ORM框架还不是强大。

江河顺水
2019-10-24

个人总结：聚合指的是能让实体和值对象共同协作的组织，作用是确保实体和值对象通过完成业务逻辑时保证数据一致性。聚合根是聚合内的“管理者”，负责业务规则的指定与协同。产生聚合的过程一般是，事件风暴->领域对象 ->实体、值对象->聚合根->聚合。
展开

作者回复: 是的。

杨杰
2019-10-24

以下情况是不是不太适合用聚合？
1、虽然有相对明确的父子关系，但是子表的数据量很大。在orm框架里面虽然有延迟加载，但是根据聚合根的要求，子表的数据难免都要缓存，这种情况下占用系统内存会很大，对于部分操作来说性能也会比较差。
2、虽然有明确的父子关系，但是对子表的操作比较频繁或者某种程度上是脱离父表而比...
展开

作者回复: 聚合还是需要的，在后面的代码目录结构了，我对聚合专门设计了一个目录结构，主要目的是起到功能和代码聚合的作用，以后微服务架构演进就相对方便。
至于聚合根，如果聚合内数据规则简单，不会产生数据不一致的情况。如果通过聚合根会影响到使用的便利性，比如性能效率等，我个人觉得也可以不设计聚合根，采用传统的设计方法也是没有关系的。一切以解决问题为基本要求，不要为了DDD而做DDD。

一零一零
2019-10-24

老师你好，领域服务和应用服务是怎么定义的？分别有什么区别？从文中看，两者的区别是否是服务粒度的不同？另外，结合微服务的实践，这两种服务在微服务中是怎么设计的？

作者回复: 应用服务主要做服务组合和编排，领域服务实现多个实体的核心业务逻辑。这一块在后面的章节他们是主角，会有详细介绍。请耐心等待。

杨杰
2019-10-23

“”设计小聚合”，我的理解就是对于那些可聚合也可以不聚合的，那么就不要聚合。

作者回复: 聚合也别太小了，一个实体设计成一个聚合，就没啥意义了。还是要考虑业务内聚性，尽量实现实体的归类。以后微服务的架构演进还要以聚合为单位来演进呢。

2

切糕
2019-10-23

个人总结：实体是项目业务所关注的对象，值对象是用来补充描述实体，这些基本的对象是整个DDD项目的基本元素，如果直接修改实体或值对象，可能会影响项目其他的业务逻辑，所以根据DDD的聚合，通过分割原则，将相应的实体和值按照聚合的方式进行分割成一个个独立的聚合，每一个聚合都依附于聚合根。
个人建议：老师讲的很透彻，感谢老师的经验分享。因为我们没有接触过DDD设计的项...
展开

作者回复: 后面章节中的设计会详细到包名，类名和服务等这些微服务代码目录结构。请耐心等待哈。

TH
2019-10-23

在别处看到，持久化只能由聚合根来进行，并且由于聚合是用来保持聚合内部的数据一致性，因此持久化时应当持久化整个聚合。

另外还有一个疑问，在与外部聚合协同时只能通过外部聚合根的ID，这个具体是怎么实现的呢？如果当前聚合不持有外部聚合根实体，要怎么使用它的业务功能呢？通过外部聚...

展开

作者回复: 是的，在一个聚合内是由聚合根来管理和协调所有实体和值对象的生命周期的，所以在一个聚合内可以保证数据的一致性。
由于多个聚合运行在同一个微服务内，在应用层我们是可以拿到其它聚合的聚合根的DO对象的，拿到聚合根后你就可以通过聚合根引用聚合的内部实体和值对象。

乘风
2019-10-23

一直疑惑，订单，商品，库存该怎么划分？分属三个领域吗？

作者回复: 应该是电商领域下的三个不同的子域。
你可以对这三个子域进行事件风暴，找出实体和聚合，划分限界上下文，建立领域模型，完成微服务设计。

三木子
2019-10-23

什么是充血模型

展开

作者回复: 说到充血模型，就离不开贫血模型。
先说一下贫血模型吧，贫血模型是指使用的领域对象中只有setter和getter方法，所有的业务逻辑都不包含在领域对象中而是放在业务逻辑层。
而充血模型将大多数业务逻辑放在领域实体中实现，实体本身包含了属性和它的业务行为，它在领域模型中就是一个具有业务行为和逻辑的基本业务单元。

2