03 | 限界上下文：定义领域边界的利器

2019-10-18 欧创新

DDD实战课 进入课程 >

讲述：欧创新
时长 14:15　大小 13.06M

你好，我是欧创新。今天我们重点学习“限界上下文”。

在 DDD 领域建模和系统建设过程中，有很多的参与者，包括领域专家、产品经理、项目经理、架构师、开发经理和测试经理等。对同样的领域知识，不同的参与角色可能会有不同的理解，那大家交流起来就会有障碍，怎么办呢？因此，在 DDD 中就出现了“通用语言”和“限界上下文”这两个重要的概念。

这两者相辅相成，通用语言定义上下文含义，限界上下文则定义领域边界，以确保每个上下文含义在它特定的边界内都具有唯一的含义，领域模型则存在于这个边界之内。你是不是感觉这么描述很抽象？没关系，接下来我会给你一一详细讲解。

在这之前，我想请你先看这样两个问题，这也是今天内容的核心。

1. 为什么要提出限界上下文的概念（也就是说除了解决交流障碍这个广义的原因，还有更具体的吗）？
2. 限界上下文在微服务设计中的作用和意义是什么？

什么是通用语言？

为了更好地理解限界上下文，回答这两个问题，我们先从通用语言讲起。

怎么理解通用语言这个概念呢？在事件风暴过程中，通过团队交流达成共识的，能够简单、清晰、准确描述业务涵义和规则的语言就是通用语言。也就是说，通用语言是团队统一的语言，不管你在团队中承担什么角色，在同一个领域的软件生命周期里都使用统一的语言进行交流。

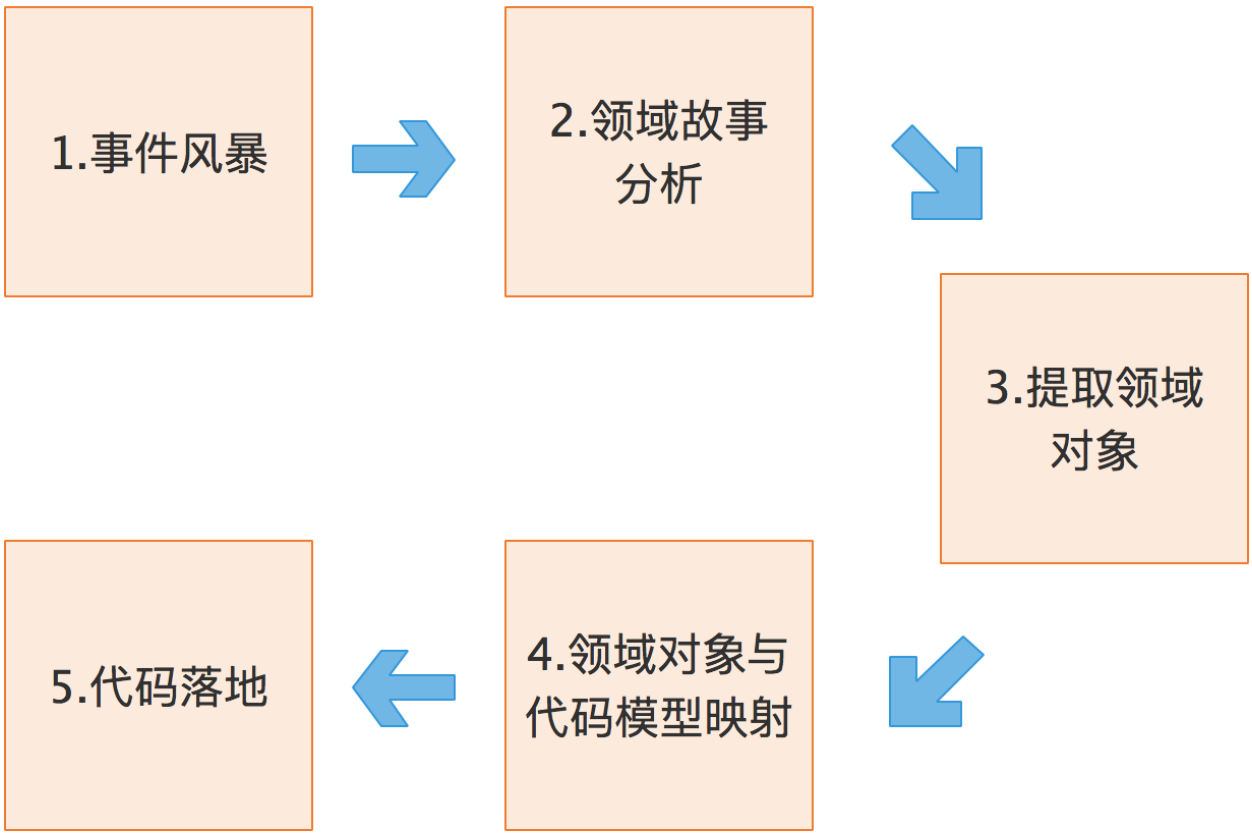
那么，通用语言的价值也就很明了了，它可以解决交流障碍这个问题，使领域专家和开发人员能够协同合作，从而确保业务需求的正确表达。

但是，对这个概念的理解，到这里还不够。

通用语言包含术语和用例场景，并且能够直接反映在代码中。通用语言中的名词可以给领域对象命名，如商品、订单等，对应实体对象；而动词则表示一个动作或事件，如商品已下单、订单已付款等，对应领域事件或者命令。

通用语言贯穿 DDD 的整个设计过程。作为项目团队沟通和协商形成的统一语言，基于它，你就能够开发出可读性更好的代码，将业务需求准确转化为代码设计。

下面我带你看一张图，这张图描述了从事件风暴建立通用语言到领域对象设计和代码落地的完整过程。



1. 在事件风暴的过程中，领域专家会和设计、开发人员一起建立领域模型，在领域建模的过程中会形成通用的业务术语和用户故事。事件风暴也是一个项目团队统一语言的过程。
2. 通过用户故事分析会形成一个个的领域对象，这些领域对象对应领域模型的业务对象，每一个业务对象和领域对象都有通用的名词术语，并且一一映射。
3. 微服务代码模型来源于领域模型，每个代码模型的代码对象跟领域对象一一对应。

这里我再给你分享一条经验，我自己经常用，特别有效。设计过程中我们可以用一些表格，来记录事件风暴和微服务设计过程中产生的领域对象及其属性。比如，领域对象在 DDD 分层架构中的位置、属性、依赖关系以及与代码模型对象的映射关系等。

下面是一个微服务设计实例的部分数据，表格中的这些名词术语就是项目团队在事件风暴过程中达成一致、可用于团队内部交流的通用语言。在这个表格里面我们可以看到，DDD 分析过程中所有的领域对象以及它们的属性都被记录下来了，除了 DDD 的领域对象，我们还记录了在微服务设计过程中领域对象所对应的代码对象，并将它们一一映射。

层	聚合	领域对象名称	领域类型	依赖的领域对象	包名	类名	方法名
应用层	/	创建请假信息应用服务	应用服务	创建请假信息领域服务	*.leave.application.service	CreateLeaveInfoAppService	CreateLeaveInfoAppService
	/	请假审批已通过	事件发布	请假审批（审核）	*.leave.application.event.publish	SendApprovalEventInfo	SendApprovalEventInfo
领域层	请假	请假单	聚合根		*.leave.domain.leave.entity	Leave	
		创建请假信息	命令		*.leave.domain.leave.entity	Leave	CreateLeaveInfo
		审批轨迹	值对象		*.leave.domain.leave.entity	ApprovalTrace	
		创建审批轨迹信息	命令		*.leave.domain.leave.entity	ApprovalTrace	CreateApprovalTrace
		创建请假信息	领域服务	创建请假信息	*.leave.domain.leave.service	CreateLeaveInfoDomService	CreateLeaveInfoDomService
		创建审批轨迹信息	领域服务	创建审批轨迹信息	*.leave.domain.leave.service	CreateApprovalTraceDomService	CreateApprovalTraceDomService
	人员	人员	聚合根		*.leave.domain.person.entity	Person	
		创建人员信息	命令		*.leave.domain.person.entity	Person	CreatePersonInfo
		组织关系	值对象		*.leave.domain.person.entity	PersonRelationship	
		创建组织关系	命令		*.leave.domain.person.entity	PersonRelationship	CreatePersonRelationship
		创建人员信息	领域服务	创建人员信息	*.leave.domain.person.service	CreatePersonInfoDomService	CreatePersonInfoDomService
		创建组织关系	领域服务	创建组织关系	*.leave.domain.person.service	CreatePersonRelationshipDomService	CreatePersonRelationshipDomService
基础层	请假	请假仓储接口	仓储接口		*.domain.leave.repository.facade	LeaveRepositoryInterface	LeaveRepositoryInterface
		请假仓储实现	仓储实现		*.domain.leave.repository.persistence	LeaveRepositoryImpl	LeaveRepositoryImpl
	人员	人员仓储接口	仓储接口		*.domain.person.repository.facade	PersonRepositoryInterface	PersonRepositoryInterface
		人员仓储实现	仓储实现		*.domain.person.repository.persistence	PersonRepositoryImpl	PersonRepositoryImpl

到这里，我要再强调一次。DDD 分析和设计过程中的每一个环节都需要保证限界上下文内术语的统一，在代码模型设计的时侯就要建立领域对象和代码对象的一一映射，从而保证业务模型和代码模型的一致，实现业务语言与代码语言的统一。

如果你做到了这一点，也就是建立了领域对象和代码对象的映射关系，那就可以指导软件开发人员准确无误地按照设计文档完成微服务开发了。即使是不熟悉代码的业务人员，也可以很快找到代码的位置。

什么是限界上下文?

那刚刚提到的限界上下文又是用来做什么的呢?

我们知道语言都有它的语义环境，同样，通用语言也有它的上下文环境。为了避免同样的概念或语义在不同的上下文环境中产生歧义，DDD 在战略设计上提出了“限界上下文”这个概念，用来确定语义所在的领域边界。

我们可以将限界上下文拆解为两个词：限界和上下文。限界就是领域的边界，而上下文则是语义环境。通过领域的限界上下文，我们就可以在统一的领域边界内用统一的语言进行交流。

综合一下，我认为限界上下文的定义就是：用来封装通用语言和领域对象，提供上下文环境，保证在领域之内的一些术语、业务相关对象等（通用语言）有一个确切的含义，没有二义性。这个边界定义了模型的适用范围，使团队所有成员能够明确地知道什么应该在模型中实现，什么不应该在模型中实现。

进一步理解限界上下文

我们可以通过一些例子进一步理解一下这个概念，不要小看它，彻底弄懂会给你后面实践 DDD 打下一个坚实的基础。

都说中文这门语言非常丰富，在不同的时空和背景下，同样的一句话会有不同的涵义。有一个例子你应该听说过。

在一个明媚的早晨，孩子起床问妈妈：“今天应该穿几件衣服呀？”妈妈回答：“能穿多少就穿多少！”

那到底是穿多还是穿少呢?

如果没有具体的语义环境，还真不太好理解。但是，如果你已经知道了这句话的语义环境，比如是寒冬腊月或者是炎炎夏日，那理解这句话的涵义就会很容易了。

所以语言离不开它的语义环境。

而业务的通用语言就有它的业务边界，我们不大可能用一个简单的术语没有歧义地去描述一个复杂的业务领域。限界上下文就是用来细分领域，从而定义通用语言所在的边界。

现在我们用一个保险领域的例子来说明下术语的边界。保险业务领域有投保单、保单、批单、赔案等保险术语，它们分别应用于保险的不同业务流程。

1. 客户投保时，业务人员记录投保信息，系统对应有投保单实体对象。
2. 缴费完成后，业务人员将投保单转为保单，系统对应有保单实体对象，保单实体与投保单实体关联。
3. 如客户需要修改保单信息，保单变为批单，系统对应有批单实体对象，批单实体与保单实体关联。
4. 如果客户发生理赔，生成赔案，系统对应有报案实体对象，报案实体对象与保单或者批单实体关联。

投保单、保单、批单、赔案等，这些术语虽然都跟保单有关，但不能将保单这个术语作用在保险全业务领域。因为术语有它的边界，超出了边界理解上就会出现问题。

如果你对我从事的保险业不大了解也没关系，电商肯定再熟悉不过了吧?

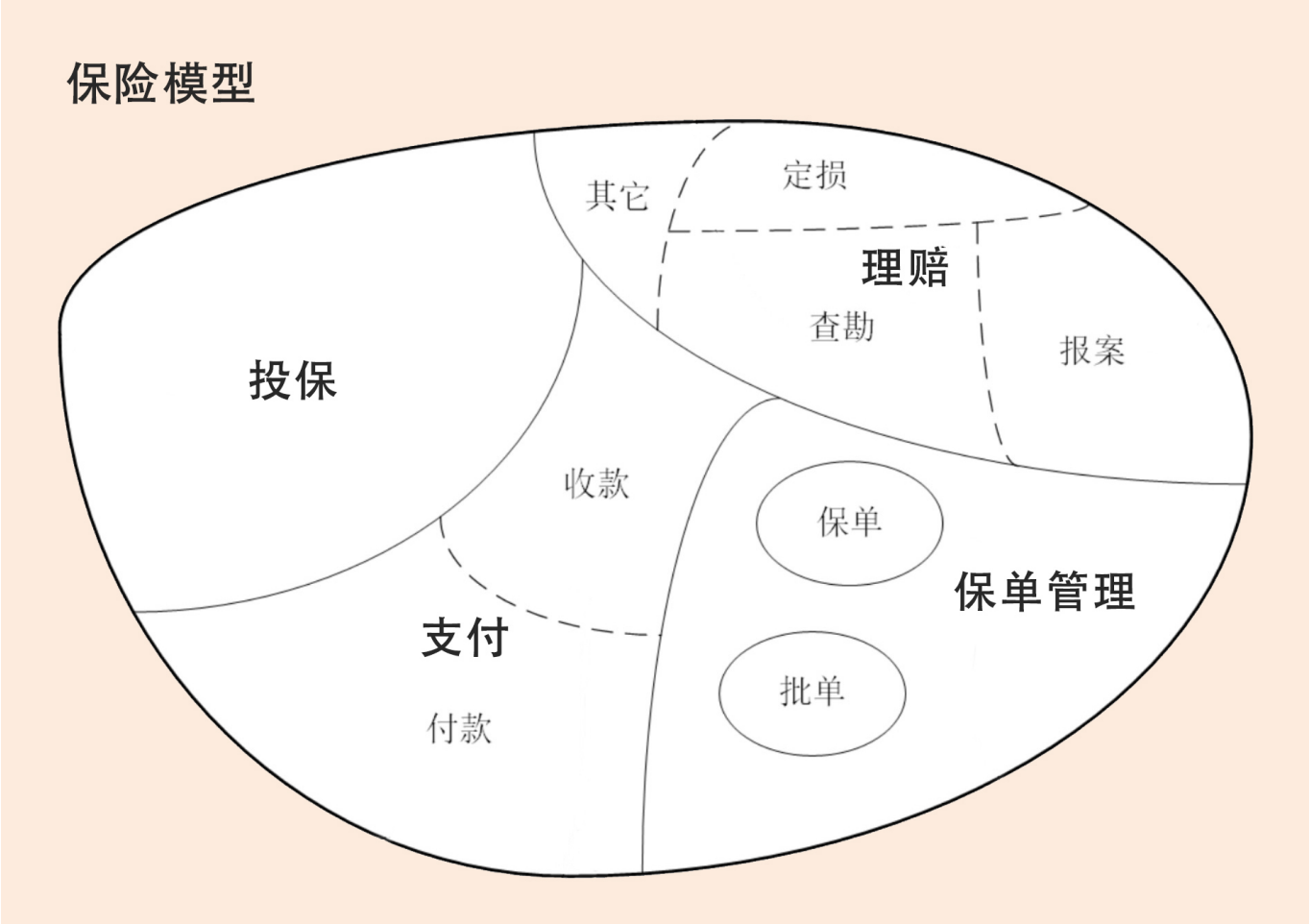
正如电商领域的商品一样，商品在不同的阶段有不同的术语，在销售阶段是商品，而在运输阶段则变成了货物。同样的一个东西，由于业务领域的不同，赋予了这些术语不同的涵义和职责边界，这个边界就可能会成为未来微服务设计的边界。看到这，我想你应该非常清楚了，领域边界就是通过限界上下文来定义的。

限界上下文和微服务的关系

接下来，我们对这个概念做进一步的延伸。看看限界上下文和微服务具体存在怎样的关系。

我想你买过车险吧，或者听过吧。车险承保的流程包含了投保、缴费、出单等几个主要流程。如果出险了还会有报案、查勘、定损、理算等理赔流程。

保险领域还是很复杂的，在这里我用一个简化的保险模型来说明下限界上下文和微服务的关系。这里还会用到我们在 [第 02 讲] 学到一些基础知识，比如领域和子域。



首先，领域可以拆分为多个子领域。一个领域相当于一个问题域，领域拆分为子域的过程就是大问题拆分为小问题的过程。在这个图里面保险领域被拆分为：投保、支付、保单管理和理赔四个子域。

子域还可根据需要进一步拆分为子子域，比如，支付子域可继续拆分为收款和付款子子域。拆到一定程度后，有些子子域的领域边界就可能变成限界上下文的边界了。

子域可能会包含多个限界上下文，如理赔子域就包括报案、查勘和定损等多个限界上下文（限界上下文与理赔的子子域领域边界重合）。也有可能子域本身的边界就是限界上下文边界，如投保子域。

每个领域模型都有它对应的限界上下文，团队在限界上下文内用通用语言交流。领域内所有限界上下文的领域模型构成整个领域的领域模型。

理论上限界上下文就是微服务的边界。我们将限界上下文内的领域模型映射到微服务，就完成了从问题域到软件的解决方案。

可以说，限界上下文是微服务设计和拆分的主要依据。在领域模型中，如果不考虑技术异构、团队沟通等其它外部因素，一个限界上下文理论上就可以设计为一个微服务。

不过，这里还是要提示一下：除了理论，微服务的拆分还是有很多限制因素的，在设计中不宜过度拆分。那这个度怎么把握好呢？有关微服务设计和具体的拆分方法，我会在实战篇中详细讲解。

总结

通用语言确定了项目团队内部交流的统一语言，而这个语言所在的语义环境则是由限界上下文来限定的，以确保语义的唯一性。

而领域专家、架构师和开发人员的主要工作就是通过事件风暴来划分限界上下文。限界上下文确定了微服务的设计和拆分方向，是微服务设计和拆分的主要依据。如果不考虑技术异构、团队沟通等其它外部因素，一个限界上下文理论上就可以设计为一个微服务。

可以说，限界上下文在微服务设计中具有很重要的意义，如果限界上下文的方向偏离，那微服务的设计结果也就可想而知了。因此，我们只有理解了限界上下文的真正涵义以及它在微服务设计中的作用，才能真正发挥 DDD 的价值，这是基础也是前提。

思考题

现在，不妨回头看看，开头的两个问题你能回答了吗？你可以尝试用自己的话来总结一下。

最后再给你留一个作业，你能找一找自己工作中的通用语言和限界上下文吗？可以把你的答案和感受写下来，分享到留言区，与我一起讨论。也欢迎你把今天的内容分享给同事和朋友，邀请他一起学习。

上一篇　02 | 领域、子域、核心域、通用域和支撑域：傻傻分不清？

下一篇　04 | 实体和值对象：从领域模型的基础单元看系统设计

精选留言 (22)

写留言

TH

2019-10-18

以前基于模块的编程方式，会将不同业务中属于同一个对象的功能都写到这个对象的类中，导致这个类非常庞大，从逻辑上来说也将复杂的业务耦合到了一起。如果使用DDD的方式来设计系统，比如文中所举的保单的例子，应该不同的业务线或者说不同的限界上下文内都应当实现自己的保单对象，对应到微服务就是以限界上下文来划分服务，而不是以对象或功能集来划分服务，因此不存在一个单独的“保单”服务，是这样的吗？

展开

作者回复: 不同产品线如果领域模型差异太大的话，建议还是分开建设。这样不同产品之间的影响就很小了，也减少了很多兼容带来的开销，用户体验也不会太好。所以可以针对不同场景的产品设计出不同的微服务，但是在前端设计的时候需要注意，这些不同类产品的业务流程和前端界面是可以很好的融合在一起的。前端的设计我在后面也会有一节来讲解。注意一下，业务差异不是太大，不会带来太大复杂度的话，还是尽量建立一个领域模型。

4

小美
2019-10-20

限界上下文大概是直译过来的一个晦涩的术语，理解成本较高。
英文是bounded context，应该叫上下文边界更合适。

展开

作者回复: 是的，赞一个。

2

碧落惊鸿LY
2019-10-18

老师，我有一个问题，对于你们保险行业来说，投保单、保单、批单、赔案，这些领悟概念都是在不同的限界上下文中，那么就是在不同的微服务中，拆分开以后，如果管理后台有一个需求，需要查出一个列表，列表的字段信息需要这些所有不同类型的订单的组合。
1这种查询你们会放在哪个微服务里做呢
2对于组合查询这种情况你们是连表查询，或者是不同服务通过id查询来提供属于它自己...

展开

作者回复: 这种跨多个微服务的数据查询。对于准实时的查询，你可以考虑数据中台。在数据中台会归拢所有的微服务的数据，在数据中台提供统一的各个维度数据的集中查询。对于实时性要求高的，你可以考虑一定的数据冗余，上游的业务数据做完后，实时写入到需要联表查询的库中，写入时只写入必要的数据就可以了。

3 2

sqyao
2019-10-20

请问欧老师，事件风暴指的是什么？

展开

作者回复: 事件风暴跟头脑风暴类似，项目团队跟领域专家一起，尽可能全面的根据领域事件梳理业务的各种可能，找出产生这些事件的实体，进行聚合，划分限界上下文建立领域模型的过程。后面会有专门一章来讲解如何做事件风暴，请耐心等待。

1 1

波波
2019-10-18

一、限界上下文的作用
1、主要是为了消除通用语言在不同领域中的歧义或者说是限制通用语言的使用范围。
2、是划分领域的重要依据
3、通用语言必须与限界上下文配合使用才有意义
...
展开

作者回复: 是的。

1

m5jun
2019-10-18

文章有个地方说的有问题，领域模型和代码模型肯定不是一一对应的吧?

作者回复: 用事件风暴建立领域模型后，我们还会进一步分析，确定更细的实体和值对象，以及各层的服务，这些都是领域对象。然后这些领域对象会跟代码模型中的数据实体，服务等代码对象建立映射关系。这个分析过程与传统的研发模式不太一样。

3 1

Geek_dcf0fc
2019-10-21

欧老师，文中提到“理赔子域就包括报案、查勘和定损等多个界限上下文(界限上下文与理赔的子子域边界重合)”这句话边界重合怎么理解?是否理赔应该作为一个微服务?另外，如果理赔流程在技术上引入了工作流引擎(如flowable)，那么这个工作流引擎又处于什么位置?
展开

作者回复: 理赔相对保险领域来说是一个比较大的子域。由于子域过大不太容易做事件风暴，因此还需要继续细分子域。当子域细分到一定程度后，对这个子域的分析就比较容易了，很有可能这个子域就是一个限界上下文，所以这时候子子域的边界与限界上下文的边界是一致的。理赔不是一个微服务，需要根据不同子域事件风暴建立领域模型后，它其中的某个业务子域就是一个微服务，比如报案之类的。你说的这个工作流引擎是不是跨了很多的微服务，如果做统一协调，按照职能划分它应该是一个工作流微服务。另外，我简单分析一下，有的时候DDD的事件驱动可以替代工作流的功能，通过事件驱动推动业务在不同的领域模型中的流转。后面的章节会讲到。

1

瓜瓜
2019-10-21

我在拆分微服务的时候，一般是按照从属关系来划分的。
软件就是在表达这个世界的人和人的关系，人和物的关系，物和物之间的关系。他们之间的关系要么是从属，要么是分类。
在我之前的一个项目中，就遇到了这个给问题，他是一个人的对象，按照从属关系，他属于其中一个微服务，可是在实际操作中，发现它和我们的用户权限微服务关系更紧密。...
展开

作者回复: 就我了解来说还没有量化的划分方法。主要依赖项目团队和领域专家的判断。
在事件风暴时候，子域不能太大，要不你hold不住。其实你说的那个人的问题，做微服务设计时会有一些技巧的，尤其是分布式架构下。如果一个对象同时存在于两个微服务中，你可以考虑一定的数据冗余。其中的一个微服务这个对象是主要的业务环节，它可以被设计为实体，而在另外一个微服务中这个数据是冗余数据，它可以设计为值对象或者关联实体。

密码123456
2019-10-21

上节课说，域的划分。这节课说怎么划分。首先要对业务标准化，使用通用语言来描述，所有的业务，每个业务必须清晰不能存在二义性。这样就能确保业务流程转化为代码。然后使用界限上下文，确定每个业务的上限和下限，不多做，也不少做。比如请假事件，就不能直接使用加班的时长来抵扣。必须调用加班的接口
展开

作者回复: 理解的很对。在一个限界上下文就做这个限界上下文内的事情，请假就做请假的事，不要把加班的事情掺和进来。

Randy Liu
2019-10-21

微服务设计实例表格中的关系描述有很好的指导与参考价值
限界上下文 = 限界 + 上下文的 释义描述得比较准确。

展开

作者回复: 我后面的章节还会有从领域模型到微服务设计方面的详细介绍，希望能帮你更好的理解这个表。请耐心等待。

sky
2019-10-20

老师您好，请问文章中保险模型的限界上下文的图是用什么工具画的

作者回复: 用visio画的。用不规则曲线一段一段连出来的。

robert
2019-10-20

老师，请问子域和限界上下文的关系是怎么样的？我可以通过问题域就可以拆成报案等几个子子域，然后通过限界上下文（封装通用语言和领域对象）做好边界隔离。那问题域是不是也可以划分微服务？对新系统可以通过问题域来划分微服务，对于老系统则通过限界上下文？

展开

作者回复: 子域和限界上下文可以这么理解。对于新系统，如果领域过大，我们就需要将它细分为子域，一直分解到适合做事件风暴的大小。事件风暴后会根据限界上下文来建立领域模型，根据领域模型来设计微服务。如果是老系统，只是重建的话，我们可以以老系统对应的业务域来做事件风暴，划分限界上下文，建立领域模型，设计微服务。

TwoPunchMan
2019-10-19

老师讲的很详细,不过建议老师专有名词可以在旁边放入原英文对照, 这样我们要多了解可以

直接搜寻国外更多资料

展开

作者回复: 好的，下次注意一下。

。

2019-10-19

mvc的设计思想也是分层，传统的后台分层是controller、service、dao三层。DDD里是分为四层。具体这两者有什么不同呢？为什么DDD比三层分层要好呢？

展开

作者回复: 我在分层架构那一节里有对比和分析，ddd分层架构领域层突出领域模型，应用层负责服务组合和编排。敬请等待。

江河顺水

2019-10-19

通用语言：团队内部能够清晰、准确的描述业务模型的语言就是通用语言。
限界上下文：为通用语言划定边界，并提供语义上下文。领域内所有界限上下文的模型构成了整个领域模型。
理论上，某些条件下，限界上下文的划分也最终确定了微服务的边界。
在视频领域内，cp专指视频内容提供商，做的事情也只能是视频发行，而不能做其他的...

展开

作者回复: 是这样的。

陈华应

2019-10-18

现在的部门没有领域专家这个概念，产品是由产品经理规划，许多时候研发产品的意见会有分歧
1。这种情况下有没有什么好点的方式来确定限界上下文呢？
2。事件风暴没有尝试过，一个服务一般就是研发侧觉得功能能够内聚，从产品中抽取一些核心功能以及范围来作为边界，是不是有更好的方法论呢？

展开

作者回复: 最好让领域专家和项目团队变成一个团队，全流程参与，统一思想和语言。限界上下文主要由业务的语义边界来确定，特定环境只说这个环境的事。这个很难定量分析，靠项目团队根据公司的业务整体把握。

陈华应
2019-10-18

老师，一直以为是先从业务来确定限界上下文，看完这篇有点疑惑，
1。限界上下文是通过细化到最后的子域的边界来决定的吗？
2。这些边界怎么组成一个完整的限界上下文呢()可以通过微服务落地的)？
3。软件是变化的，那限界上下文是不是也是会变化的呢？比如会根据这个变化来进行微服务拆分？...
展开

作者回复: 限界上下文是从业务端分析得出的。由于一般的领域过大，不太好下手去做事件风暴，所以在划分到合适的子域后做事件风暴就没那么复杂了。限界上下文是由若干个聚合构成的，聚合具有一定的业务内聚性。在依据限界上下文完成微服务设计后，以后还是可以很容易根据领域模型的变化来演进的。微服务演进过程主要是微服务之间聚合的重构。所以设计时要做好聚合的代码边界。

二两豆腐
2019-10-18

针对保险，财务，银行这些从传统行业转型过来的行业，他们都有已经非常完整的业务模型，有自己的领域专家。老师我的问题是：针对现在很多互联网公司，产品需求都是有产品经理驱动，但是产品经理也未必是领域专家，所以和领域专家深度交流进行统一语言这个可能会有一些困难，这个如何破局。
另外老师说限界上下文和统一语言是通过事件风暴形式产生出来的，具体事件风暴是怎...
展开

作者回复: 传统行业有不少应用是单体系统，而且互金和核心之间重复建设的情况不少，因此在做中台的时候需要进行业务模型的重构。中台领域模型的重构和事件风暴详细介绍我会在后续的章节讲到。

吃饭饭
2019-10-18

感觉这个跟传统的 Model -> Dao -> Service -> Controller 有点不一样，边界内的领域模型是承载了真正的面向对象这里理念吗？感觉某些具有相同点的子域的集合有可能就是一个微服务，不知道理解的是否有偏差，望老师指正：）

展开

作者回复: 没错，就是这样的。跟三层架构的对比分析，我会在分层架构那一节详细讲解。


杨杰
2019-10-18

关于时间风暴后面会详细讲么？

展开

作者回复: 会有一节专门介绍。