06 | 领域事件：解耦微服务的关键

2019-10-25 欧创新

DDD实战课 进入课程 >

讲述：欧创新
时长 17:36　大小 14.11M

你好，我是欧创新。今天我们来聊一聊“领域事件（Domain Event）”。

在事件风暴（Event Storming）时，我们发现除了命令和操作等业务行为以外，还有一种非常重要的事件，这种事件发生后通常会导致进一步的业务操作，在 DDD 中这种事件被称为领域事件。

这只是最简单的定义，并不能让我们真正理解它。那到底什么是领域事件？领域事件的技术实现机制是怎样的？这一讲，我们就重点解决这两个大的问题。

领域事件

领域事件是领域模型中非常重要的一部分，用来表示领域中发生的事件。一个领域事件将导致进一步的业务操作，在实现业务解耦的同时，还有助于形成完整的业务闭环。

举例来说的话，领域事件可以是业务流程的一个步骤，比如投保业务缴费完成后，触发投保单转保单的动作；也可能是定时批处理过程中发生的事件，比如批处理生成季缴保费通知单，触发发送缴费邮件通知操作；或者一个事件发生后触发的后续动作，比如密码连续输错三次，触发锁定账户的动作。

那如何识别领域事件呢？

很简单，和刚才讲的定义是强关联的。在做用户旅程或者场景分析时，我们要捕捉业务、需求人员或领域专家口中的关键词：“如果发生……，则……”“当做完……的时候，请通知……”“发生……时，则……”等。在这些场景中，如果发生某种事件后，会触发进一步的操作，那么这个事件很可能就是领域事件。

那领域事件为什么要用最终一致性，而不是传统 SOA 的直接调用的方式呢？

我们一起回顾一下 [第 05 讲] 讲到的聚合的一个设计原则：在边界之外使用最终一致性。一次事务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，应采用领域事件的最终一致性。

领域事件驱动设计可以切断领域模型之间的强依赖关系，事件发布完成后，发布方不必关心后续订阅方事件处理是否成功，这样可以实现领域模型的解耦，维护领域模型的独立性和数据的一致性。在领域模型映射到微服务系统架构时，领域事件可以解耦微服务，微服务之间的数据不必要求强一致性，而是基于事件的最终一致性。

回到具体的业务场景，我们发现有的领域事件发生在微服务内的聚合之间，有的则发生在微服务之间，还有两者皆有的场景，一般来说跨微服务的领域事件处理居多。在微服务设计时不同领域事件的处理方式会不一样。

1. 微服务内的领域事件

当领域事件发生在微服务内的聚合之间，领域事件发生后完成事件实体构建和事件数据持久化，发布方聚合将事件发布到事件总线，订阅方接收事件数据完成后续业务操作。

微服务内大部分事件的集成，都发生在同一个进程内，进程自身可以很好地控制事务，因此不一定需要引入消息中间件。但一个事件如果同时更新多个聚合，按照 DDD“一次事务只更新一个聚合”的原则，你就要考虑是否引入事件总线。但微服务内的事件总线，可能会增加开发的复杂度，因此你需要结合应用复杂度和收益进行综合考虑。

微服务内应用服务，可以通过跨聚合的服务编排和组合，以服务调用的方式完成跨聚合的访问，这种方式通常应用于实时性和数据一致性要求高的场景。这个过程会用到分布式事务，以保证发布方和订阅方的数据同时更新成功。

2. 微服务之间的领域事件

跨微服务的领域事件会在不同的限界上下文或领域模型之间实现业务协作，其主要目的是实现微服务解耦，减轻微服务之间实时服务访问的压力。

领域事件发生在微服务之间的场景比较多，事件处理的机制也更加复杂。跨微服务的事件可以推动业务流程或者数据在不同的子域或微服务间直接流转。

跨微服务的事件机制要总体考虑事件构建、发布和订阅、事件数据持久化、消息中间件，甚至事件数据持久化时还可能需要考虑引入分布式事务机制等。

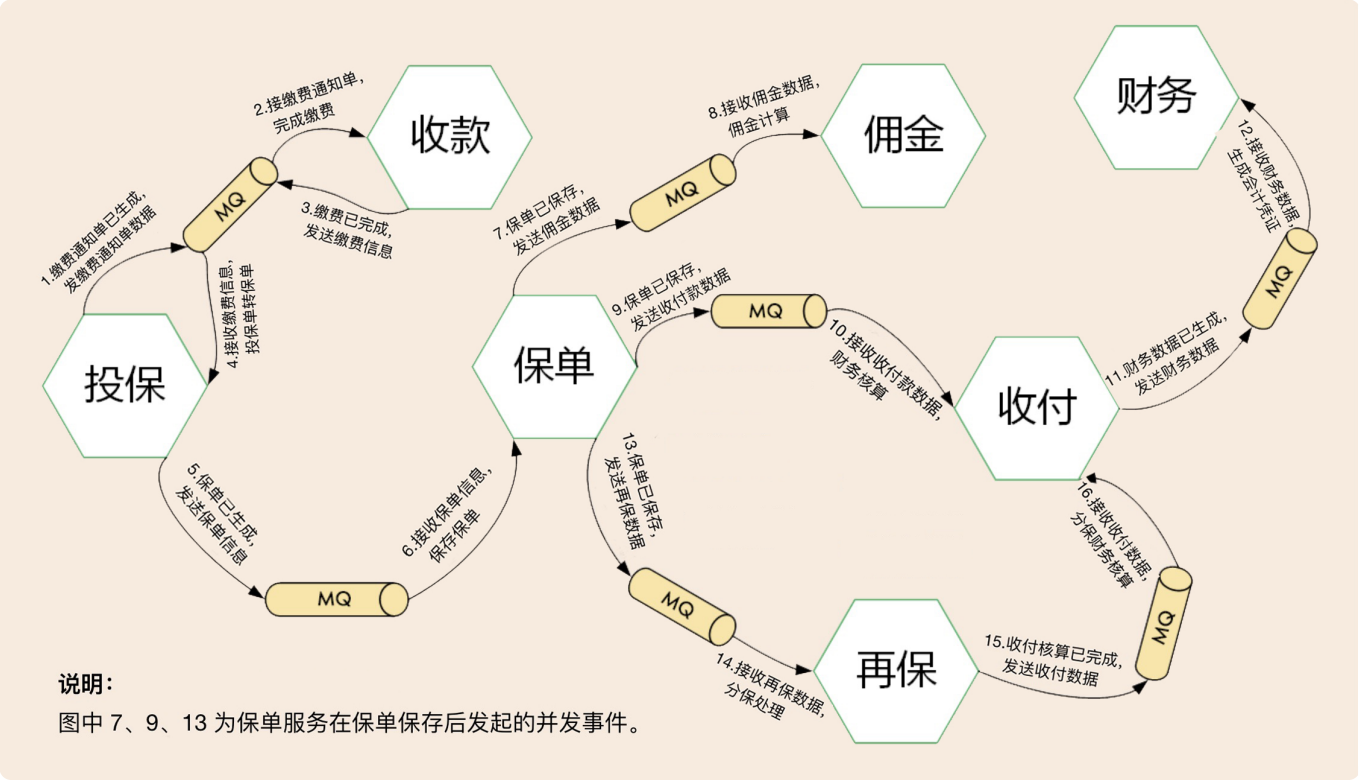
微服务之间的访问也可以采用应用服务直接调用的方式，实现数据和服务的实时访问，弊端就是跨微服务的数据同时变更需要引入分布式事务，以确保数据的一致性。分布式事务机制会影响系统性能，增加微服务之间的耦合，所以我们还是要尽量避免使用分布式事务。

领域事件相关案例

我来给你介绍一个保险承保业务过程中有关领域事件的案例。

一个保单的生成，经历了很多子域、业务状态变更和跨微服务业务数据的传递。这个过程会产生很多的领域事件，这些领域事件促成了保险业务数据、对象在不同的微服务和子域之间的流转和角色转换。

在下面这张图中，我列出了几个关键流程，用来说明如何用领域事件驱动设计来驱动承保业务流程。



事件起点：客户购买保险 - 业务人员完成保单录入 - 生成投保单 - 启动缴费动作。

1. 投保微服务生成缴费通知单，发布第一个事件：缴费通知单已生成，将缴费通知单数据发布到消息中间件。收款微服务订阅缴费通知单事件，完成缴费操作。缴费通知单已生成，领域事件结束。

2. 收款微服务缴费完成后，发布第二个领域事件：缴费已完成，将缴费数据发布到消息中间件。原来的订阅方收款微服务这时则变成了发布方。原来的事件发布方投保微服务转换为订阅方。投保微服务在收到缴费信息并确认缴费完成后，完成投保单转成保单的操作。缴费已完成，领域事件结束。

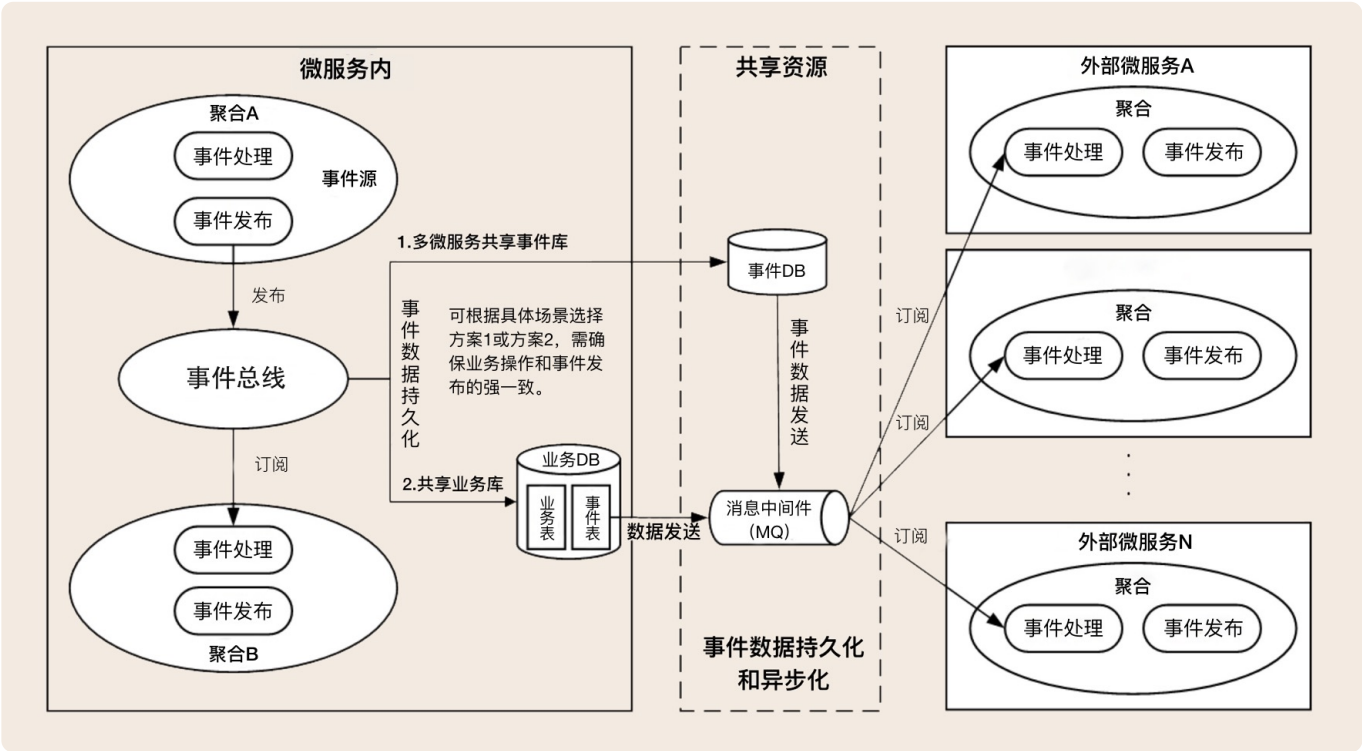
3. 投保微服务在投保单转保单完成后，发布第三个领域事件：保单已生成，将保单数据发布到消息中间件。保单微服务接收到保单数据后，完成保单数据保存操作。保单已生成，领域事件结束。

4. 保单微服务完成保单数据保存后，后面还会发生一系列的领域事件，以并发的方式将保单数据通过消息中间件发送到佣金、收付费和再保等微服务，一直到财务，完后保单后续所有业务流程。这里就不详细说了。

总之，通过领域事件驱动的异步化机制，可以推动业务流程和数据在各个不同微服务之间的流转，实现微服务的解耦，减轻微服务之间服务调用的压力，提升用户体验。

领域事件总体架构

领域事件的执行需要一系列的组件和技术来支撑。我们来看一下这个领域事件总体技术架构图，领域事件处理包括：事件构建和发布、事件数据持久化、事件总线、消息中间件、事件接收和处理等。下面我们逐一讲一下。



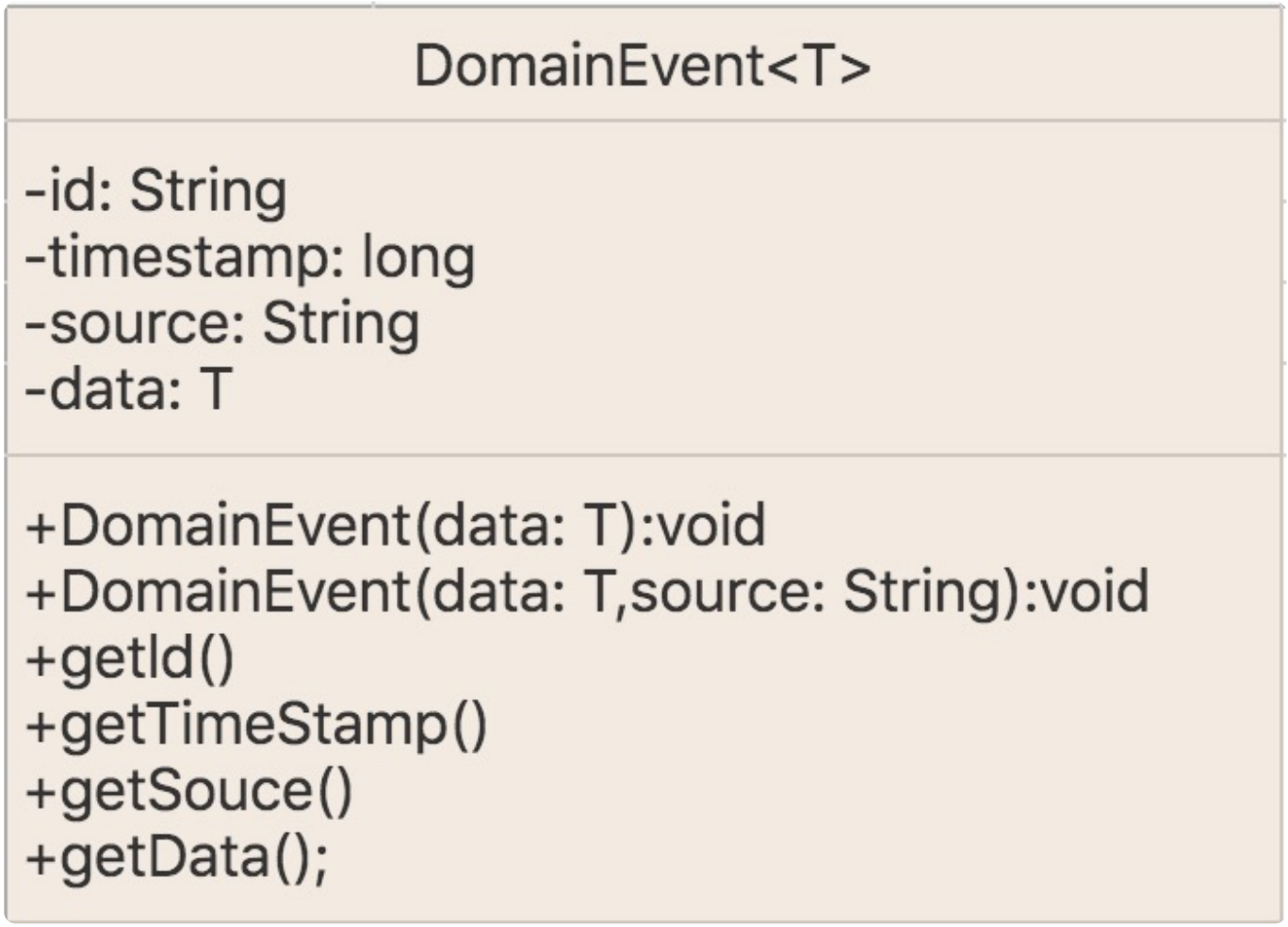
1. 事件构建和发布

事件基本属性至少包括：事件唯一标识、发生时间、事件类型和事件源，其中事件唯一标识应该是全局唯一的，以便事件能够无歧义地在多个限界上下文中传递。事件基本属性主要记录事件自身以及事件发生背景的数据。

另外事件中还有一项更重要，那就是业务属性，用于记录事件发生那一刻的业务数据，这些数据会随事件传输到订阅方，以开展下一步的业务操作。

事件基本属性和业务属性一起构成事件实体，事件实体依赖聚合根。领域事件发生后，事件中的业务数据不再修改，因此业务数据可以以序列化值对象的形式保存，这种存储格式在消息中间件中也比较容易解析和获取。

为了保证事件结构的统一，我们还会创建事件基类 DomainEvent（参考下图），子类可以扩充属性和方法。由于事件没有太多的业务行为，实现方法一般比较简单。



事件发布之前需要先构建事件实体并持久化。事件发布的方式有很多种，你可以通过应用服务或者领域服务发布到事件总线或者消息中间件，也可以从事件表中利用定时程序或数据库日志捕获技术获取增量事件数据，发布到消息中间件。

2. 事件数据持久化

事件数据持久化可用于系统之间的数据对账，或者实现发布方和订阅方事件数据的审计。当遇到消息中间件、订阅方系统宕机或者网络中断，在问题解决后仍可继续后续业务流转，保证数据的一致性。

事件数据持久化有两种方案，在实施过程中你可以根据自己的业务场景进行选择。

持久化到本地业务数据库的事件表中，利用本地事务保证业务和事件数据的一致性。

持久化到共享的事件数据库中。这里需要注意的是：业务数据库和事件数据库不在一个数据库中，它们的数据持久化操作会跨数据库，因此需要分布式事务机制来保证业务和事件数据的强一致性，结果就是会对系统性能造成一定的影响。

3. 事件总线 (EventBus)

事件总线是实现微服务内聚合之间领域事件的重要组件，它提供事件分发和接收等服务。事件总线是进程内模型，它会在微服务内聚合之间遍历订阅者列表，采取同步或异步的模式传递数据。事件分发流程大致如下：

如果是微服务内的订阅者（其它聚合），则直接分发到指定订阅者；

如果是微服务外的订阅者，将事件数据保存到事件库（表）并异步发送到消息中间件；

如果同时存在微服务内和外订阅者，则先分发到内部订阅者，将事件消息保存到事件库（表），再异步发送到消息中间件。

4. 消息中间件

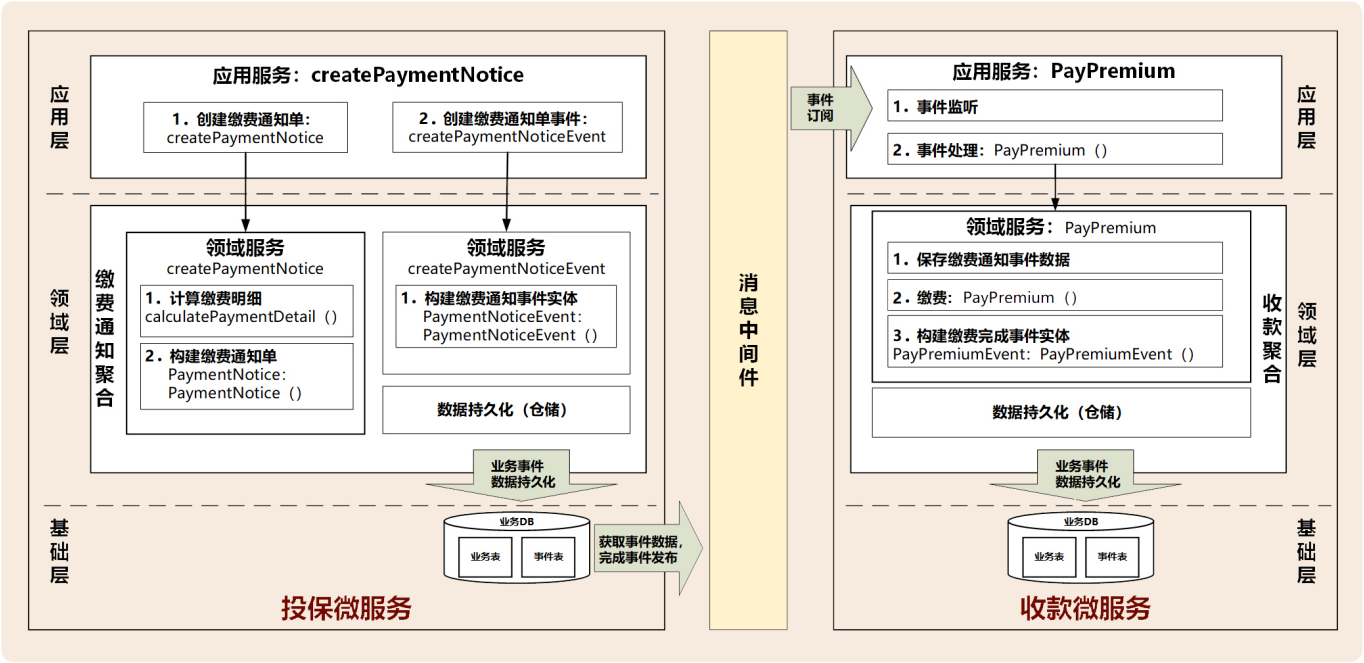
跨微服务的领域事件大多会用到消息中间件，实现跨微服务的事件发布和订阅。消息中间件的产品非常成熟，市场上可选的技术也非常多，比如 Kafka，RabbitMQ 等。

5. 事件接收和处理

微服务订阅方在应用层采用监听机制，接收消息队列中的事件数据，完成事件数据的持久化后，就可以开始进一步的业务处理。领域事件处理可在领域服务中实现。

领域事件运行机制相关案例

这里我用承保业务流程的缴费通知单事件，来给你解释一下领域事件的运行机制。这个领域事件发生在投保和收款微服务之间。发生的领域事件是：缴费通知单已生成。下一步的业务操作是：缴费。



事件起点：出单员生成投保单，核保通过后，发起生成缴费通知单的操作。

1. 投保微服务应用服务，调用聚合中的领域服务 createPaymentNotice 和 createPaymentNoticeEvent，分别创建缴费通知单、缴费通知单事件。其中缴费通知单事件类 PaymentNoticeEvent 继承基类 DomainEvent。

2. 利用仓储服务持久化缴费通知单相关的业务和事件数据。为了避免分布式事务，这些业务和事件数据都持久化到本地投保微服务数据库中。

3. 通过数据库日志捕获技术或者定时程序，从数据库事件表中获取事件增量数据，发布到消息中间件。这里说明：事件发布也可以通过应用服务或者领域服务完成发布。

4. 收款微服务在应用层从消息中间件订阅缴费通知单事件消息主题，监听并获取事件数据后，应用服务调用领域层的领域服务将事件数据持久化到本地数据库中。

5. 收款微服务调用领域层的领域服务 PayPremium，完成缴费。

6. 事件结束。

提示：缴费完成后，后续流程的微服务还会产生很多新的领域事件，比如缴费已完成、保单已保存等等。这些后续的事件处理基本上跟 1 ~ 6 的处理机制类似。

总结

今天我们主要讲了领域事件以及领域事件的处理机制。领域事件驱动是很成熟的技术，在很多分布式架构中得到了大量的使用。领域事件是 DDD 的一个重要概念，在设计时我们要重点关注领域事件，用领域事件来驱动业务的流转，尽量采用基于事件的最终一致，降低微服务之间直接访问的压力，实现微服务之间的解耦，维护领域模型的独立性和数据一致性。

除此之外，领域事件驱动机制可以实现一个发布方 N 个订阅方的模式，这在传统的直接服务调用设计中基本是不可能做到的。

思考题

思考一下你公司有哪些业务场景可以采用领域事件驱动的设计方式？

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，共同进步。

上一篇　05 | 聚合和聚合根：怎样设计聚合？

下一篇　07 | DDD分层架构：有效降低层与层之间的依赖

精选留言 (24)

写留言

阿神
2019-10-25

微服务内的领域事件我建议少用，增加复杂性了

展开

作者回复: 我跟你的观点一样。

2

杨杰
2019-10-28

关于领域事件有几个问题：
1、如果采用了主流的消息队列（如rabbitmq，kafka，rocketmq），是否领域事件还需要持久化?
2、对于领域事件的内容，是否需要把所有变化（或绝大多数）的内容都保存到事件里面（比如保单里面的所有内容）？这样的话这个这个领域事件会不会比较大？...

展开

作者回复: 第一个问题：虽然这些消息队列自身有持久化的功能，但是中间过程，或者在订阅到数据后，在处理之前出问题，需要进行数据对账，这样就没法找到发布时和处理后的数据版本。关键的业务数据我建议还是落库比较好。
第二个问题：事件实体的业务数据还是按需发布比较好，避免不必要的业务信息泄露。
第三个问题：CDC的设计方式比较简单，属于数据库底层的技术，不会对上层应用产生太多的影响。

1

杨杰
2019-10-28

关于领域事件还有一个问题请教：对于大型系统来说，领域事件肯定有很多，在整个领域事件的管理方面有没有什么经验可以分享的?

展开

作者回复: 我感觉，第一是要做好源端和目的端数据的对账处理，发现并识别处理过程中的异常数据。第二是，发现异常数据后的，要有相应的处理机制。第三是，选择适合自己场景的技术，保证数据正确传输。

。
2019-10-28

欧老师，你好。有两个问题请教一下
1.微服务内的领域事件。
我看你介绍的，需要引入事件总线和分布式事务。同一个服务内，应该是同一个进程，数据库一般也
是同一个。采用直接调用是不是就可以？如：在a领域中 b.method();...

展开

作者回复: 微服务内的领域事件主要是避免一次事务同时修改多个聚合的数据，解耦聚合和保证数据的一致性。但聚合内采用事件总线会增加开发的复杂度，个人不建议多用。
2、这种异步的方式一般都有源端和目的端定期对账的机制。我见得最多的是采用类似财务冲正的方式。如果在发布和订阅之间事件表的数据发现异步数据有问题，需要回退，会有相应的代码进行数据处理，不过不同的场景，业务逻辑会不一样，处理的方式会不一样。有的甚至还需要转人工处理。

Geek_b5c4eb
2019-10-28

老师好，最后一个缴费例子，每个领域服务都对应一个应用服务么，而且图上事件的触发和订阅都由相应的应用服务来做，那应用服务是不是很重，而且领域服务显得不内聚了。实践中应用服务我们是做成了面向某一类用户的应用服务，如面向c端一个应用服务，面向内部运营系统一个应用服务，请问老师这样理解应用服务是否正确，谢谢！

展开

作者回复: 这里碰巧是领域服务对应一个应用服务。应用服务主要是做组合和编排，不具有太多的逻辑，所以不会太重。作为事件服务的发布和订阅，在应用服务中也只是完成发布和接受，不做业务逻辑处理，事件处理是在领域层去做的，所有也不会太重。
应用层主要是做服务组合、转换和传递的作用。不会有太多的业务逻辑处理。这样设计也相当于是将部分与业务无关的逻辑剥离出领域层。

陈勇
2019-10-27

老师能否用springboot和mybatis技术栈开源一个实战项目呢？比如对电商的商品域和营销域进行建模？通过一个项目把DDD的各个设计元素串起来？不然大家还是会看起来懂了，实践起来还是不会。谢谢

作者回复: 等后面有时间的时候我试试看哈。不过DDD主要还是在于设计思想，你先从这个课件里了解一下DDD是怎样进行战略设计和战术设计的。咱们可以多保持交流。

约书亚
2019-10-27

看了最后的例子我有很大的疑惑，似乎领域事件并不和触发其的服务等，在同一个聚合里？而是分开两个聚合，在应用层一起调用？
按照直观感受把事件本身和事件源放在一起不是更内聚？

展开

作者回复: 这些领域事件分成两大类，一个是微服务内跨聚合的领域事件，另一个是微服务之间的领域事件。一个聚合内一般都不用领域事件驱动的方式设计，它们在同一个进程内，不需要采用异步的方式。
微服务内和微服务之间都可以采用应用服务调用的方式来实现。

1

约书亚
2019-10-27

请问您，领域事件要由领域服务产生，而不是由相关的实体实现产生领域事件的方法？

作者回复: 领域事件由外部命令触发。触发命令可以是领域服务，也可以是实体的某一个方法或者行为。

密码123456
2019-10-27

领域事件。通过领域事件，将各个分域后的组件形成联系，对内形成一种弱关联，对整个领域形成强关联。

Jxin
2019-10-26

赞同老师触发事件的用法。走canal增量同步数据库数据，通过监听特定表的数据变更来触发生成事件的调用。如此有利于主流业务的解耦，提高维护和可读性。（具体生成事件的操作当然还是放在对应领域的微服务中，canal监听消费端可以理解为一个任务调度平台）

展开

作者回复: 是的。这样的实现逻辑相对简单。

你的美
2019-10-26

用聚合内的领域事件服务（事件总线），是为了减轻微服务之间调用，消息中间件的压力，不用也行；用的话就可以将聚合内的服务调用设置成同步做实时提醒，在限界上下文的微服务之间设置成异步，实现最终的数据一致性。
老师这样理解对吗？

展开

作者回复: 聚合之间的领域事件是为了保证同一个微服务内聚合之间的数据一致性，记住：是在同一个微服务内。
它会增加开发的复杂度，应尽量少用，通过应用层的应用服务也能实现。
微服务之间的领域事件，可以采用异步方式来实现，减轻微服务之间直接调用的压力。

夙梦流尘
2019-10-26

有两个问题想问下
1 服务内的事件我觉得也是要落库吧。因为要保证最终一致性
2 外部服务关心的消息可以不走事件总线直接在应用服务里发消息吗？

展开

作者回复: 因为在微服务内部在同一个进程，事件总线相对好配置，它可以配置为异步的也可以配置为同步的。如果是同步就不需要落库。我还是建议尽量少用微服务内聚合之间的领域事件，它会增加你的开发复杂度。
微服务之间的事件，在事件数据落库后，通过应用服务直接发布到消息中间件的。

李赐麟
2019-10-25

有点cqrs的味道了。希望能有新的出路

展开

作者回复: 领域事件跟CQRS有那么一点差异。
CQRS主要是想读写分离，将没有领域模型的查询功能，从命令中分离出来。领域事件主要目的还是为了微服务解耦，在连续的业务处理过程中，以异步化的方式完成下一步的业务处理，降低微服务之间的直连。
它们的共同点就是通过消息中间件实现从源端数据到目的端数据的交互和分离。


心浮天空
2019-10-25

个人理解：
同一微服务中的多个聚合之间的协作可以在应用服务层处理, 如果多个聚合根共享同一数据库时, 可以直接利用本地事务达到数据的一致性, 但这样的协作方式不利于之后聚合之间的拆分和独立演化，不过这种协作方式实现成本最低，效率也高, 在前期是个很好的选择。
跨服务间领域事件, 需要持久化存储, 防止事件丢失，当领域事件通知下游服务失败时, 可...

展开

作者回复: 第一个问题，发送失败后是可以通过定时程序重新发送的，如果系统不能自动处理，还可以设计成转人工处理。
第二个问题，我们会定义一个领域事件的实体，这是一个事件对象，用于存储事件数据。它在代码目录结构里有一个专门的目录，你可以继承事件基类，定义事件实体。


2


嘉嘉
2019-10-25

老师好，
请问，事件数据，发布方和订阅方都要做持久化吗？

作者回复: 建议做，如果你的数据不重要，不需要对账或者丢了也没关系。那就不需要。


蜗牛慢慢爬

2019-10-25

建议把源码发出来好理解些，光这些流程有些抽象。

展开

作者回复: 不好意思哈，不太方便放出来。有疑问咱们可以多沟通。

wstr

2019-10-25

太抽象了看不懂!

展开

编辑回复: 可以把问题具体化向作者提问，希望能帮到你!


1

夙梦流尘

2019-10-25

那如果少用领域事件，聚合之间应该怎么交互呢

展开

作者回复: 可以通过应用层来协调和交互。应用服务是所有聚合之上的服务，负责服务的组合和编排，以及聚合之间的协调。


3

TH

2019-10-25

课表中好像没有涉及到领域服务和应用服务，老师可以讲一讲吗？本文案例中领域服务与应用服务同名，那么他们的概念和区别是什么呢？

另外，投保单转保单这个功能的实现位于投保微服务还是保单微服务？对应到上下文设计，又应当属于投保上下文还是保单上下文呢？从上下文到具体的实现，会不会出现上...

展开

作者回复: 我在分层架构和其它章节会详细讲解这两个服务的。同名的话一般会通过后缀名来区分，比如*appservice是应用服务，*domainservice是领域服务。

投保转保单是在投保微服务完成的，它是投保微服务的最后一个业务操作。保单数据发送到保单微服务后，就到了保单管理的限界上下文了。上下文与实现一般都是一致的。

朱振光
2019-10-25

内容很充实，但还是有两个问题。
1. 在采取最终一致性的情况下，event消费端如果出现错误，消费失败，但是之前的业务都成功了，虽然记录了event dB，但是后续如何处理，是需要人工介入解决吗？如果人工介入再解决，前端用户会不会看到数据不一致，体验不好？
2. 因为event都是异步发送，当最后一个event消费成功后，如何有效的通知前端界面，...
展开

作者回复: 第一个问题，失败的情况应该比例是很少的。失败的信息可以采用多次重发的方式，如果这个还解决不了，只能将有问题的数据放到一个问题数据区，人工解决。当然要确保一个前提，要保证数据的时序性，不能覆盖已经产生的数据。
第二个问题，一般来说发布方不会等待订阅方反馈结果。发布方有发布的事件表，订阅方有消费事件表，你可以采用每日对账的方式来发现问题数据。