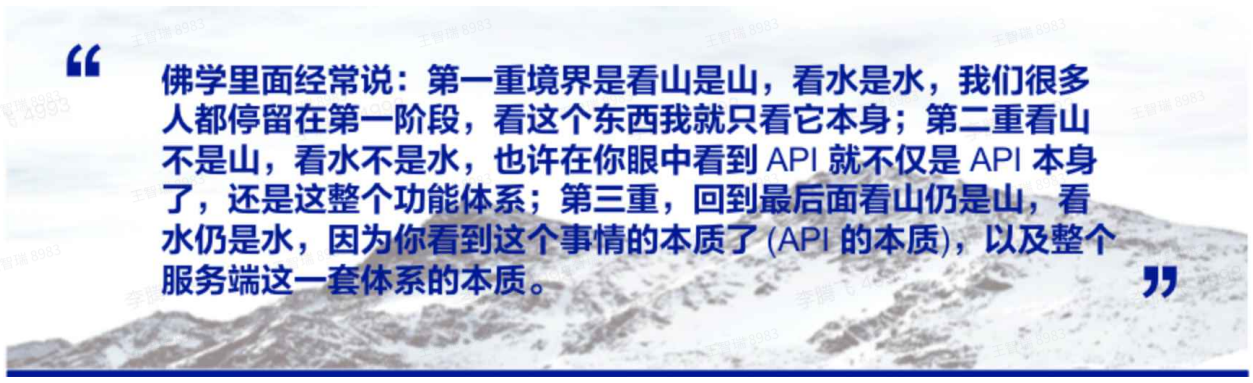
如何做好架构设计

💡 你是否在深夜想过 “天天写 CRUD 再写两年自己就废了”

> 本文整体内容摘自：洪定坤|新人成长 ，架构实战案例解析-王庆友，基于 DDD 的微服务拆分与设计-欧创新

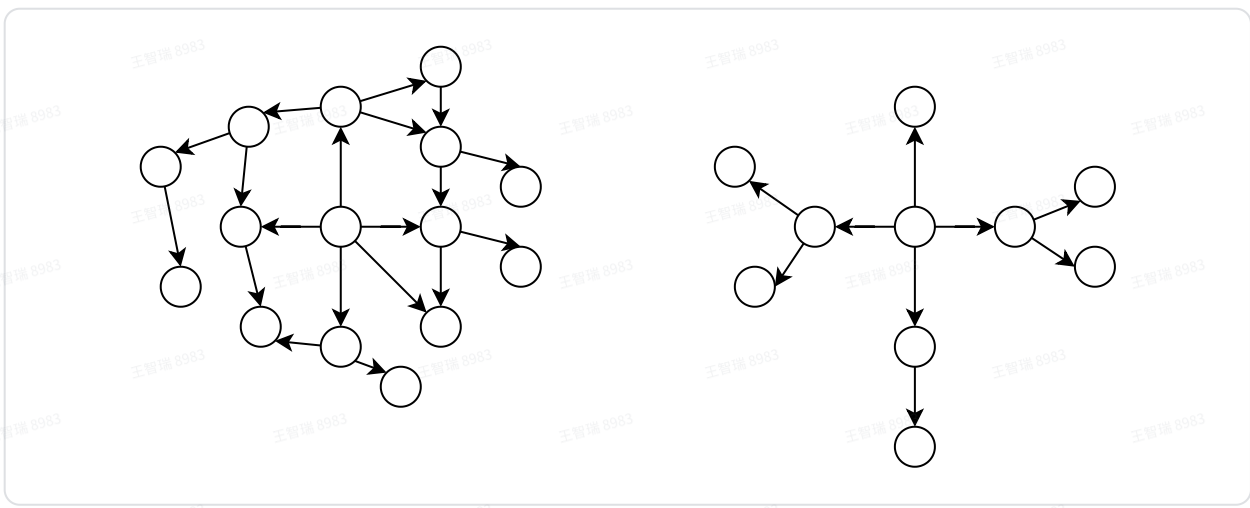


为什么学习架构

> 无架构，不系统。如何画好架构图 这篇文档解决脑子里有东西写不出来

掌握好架构设计，可以让你轻松应对技术和业务的挑战。但是很多技术人员，由于个人项目经验有限，又缺乏很好的学习途径，对架构设计一知半解。在实际工作中，不能把握好架构设计的度，要么设计不足，要么过度设计，导致系统变来变去，严重影响开发效率和质量

软件开发都是伴随着复杂度的产生。为了让系统良好的发展，就需要控制复杂度，进行架构设计。



战术编程 VS. 战略编程

如何学习架构

1. 首先，你需要跳出当前的小模块，站在系统整体的角度来考虑问题。
2. 不仅要从技术的角度考虑问题，也要学会从业务的角度来考虑问题，深入理解系统的挑战在哪里，不要在错误的地方发力。
3. **要做好各方面的平衡，能在现有的各项资源约束下，寻求一个最优解。因地制宜做到“适合”，“适度”**

架构的本质是什么

通过合理的“分”与“合”，系统不是回到了原点，而是把原先铁板一块的系统变成一个富有弹性的结构化系统。这样，系统的复杂性有效地分解了，系统的有序度大幅度地提升

评论（21）

> 如何做好架构设计

李腾飞 2022年12月15日

3,4 年之前团队小，业务发展快，对技术评审不够重视，后面组里规范了关于开发的流程，第一步就是功能开发需要做技术评审，希望提高这方面的意识，但是评审的质量不高，借助本篇文档梳理一些思路。

李腾飞 2022年12月16日

如何做好架构设计 这篇文档解决如何思考问题。

如何画好架构图 这篇文档解决脑子里有东西写不出来

> [流程图]

李腾飞 2022年12月16日

好代码很重要，好架构更重要。

李腾飞 2022年12月16日

代码服务->服务复用
代码解耦->服务解耦
代码简洁->架构简洁

了

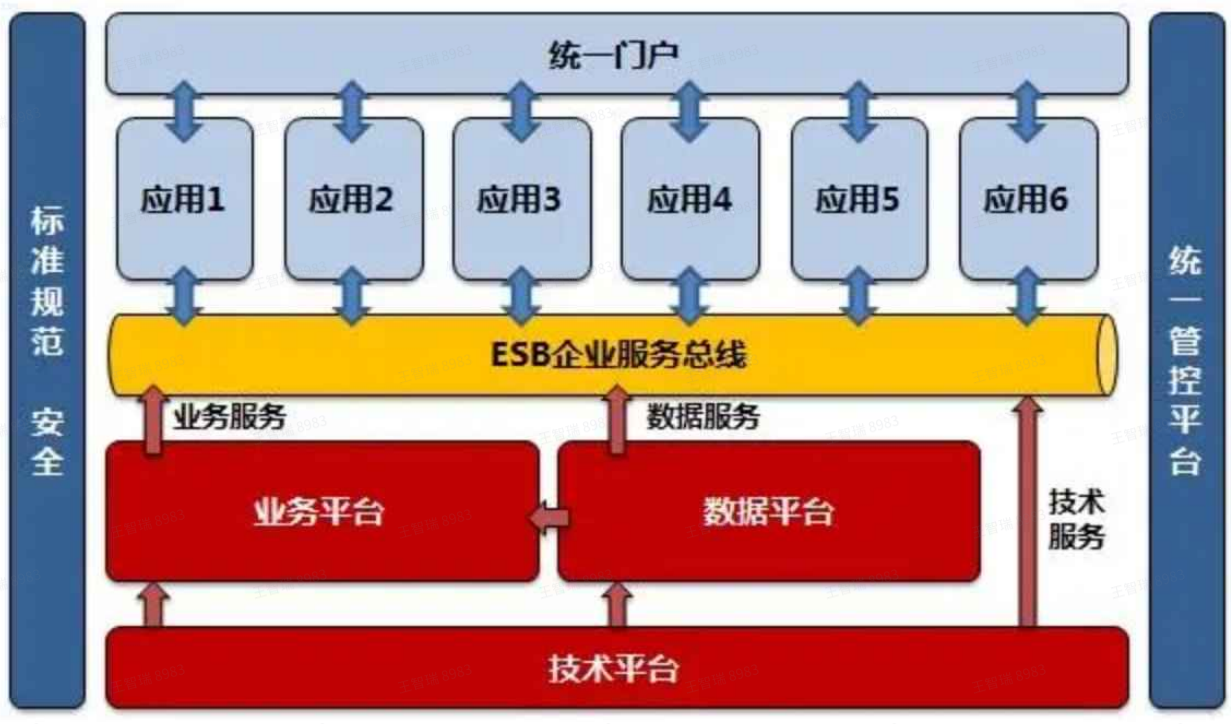
架构的分类

业务架构

业务架构就是讲清楚核心业务的处理过程，定义各个业务模块的相互关系，它从概念层面帮助我们理解系统面临哪些问题以及如何处理

应用架构

应用架构就是讲清楚系统内部是怎么组织的，有哪些应用，相互间是怎么调用的，它从逻辑层面帮助我们理解系统内部是如何分工与协作的



技术架构

业务架构

李腾飞 2022年12月15日

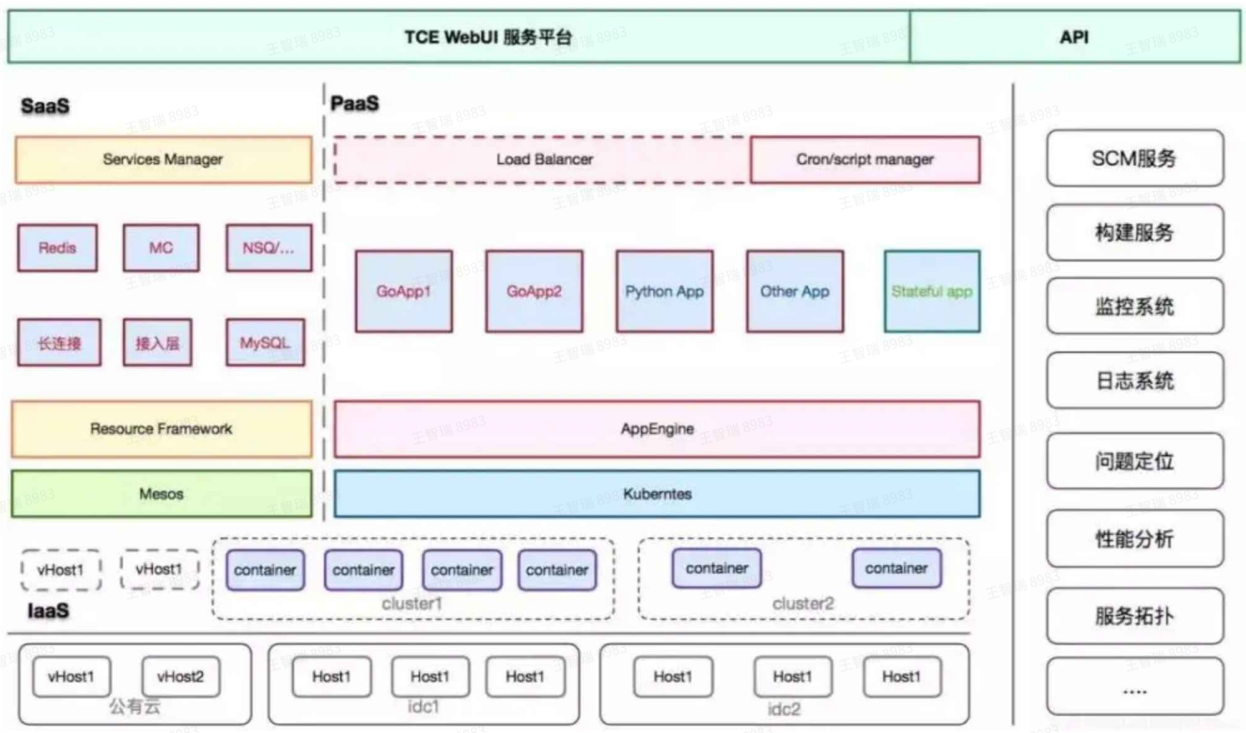
这个是最重要的，团队或者系统大起来以后每个人对业务的理解都不一样，我们需要通过一种方式对齐团队内对业务的理解和划分。日常我们接触到的都是零散的需求，实现这些需求解决用户什么问题？这个功能是支撑哪块业务？时间长了会让 RD 的思考陷入自己常维护的几个服务中缺少全局的视角。

应用架构

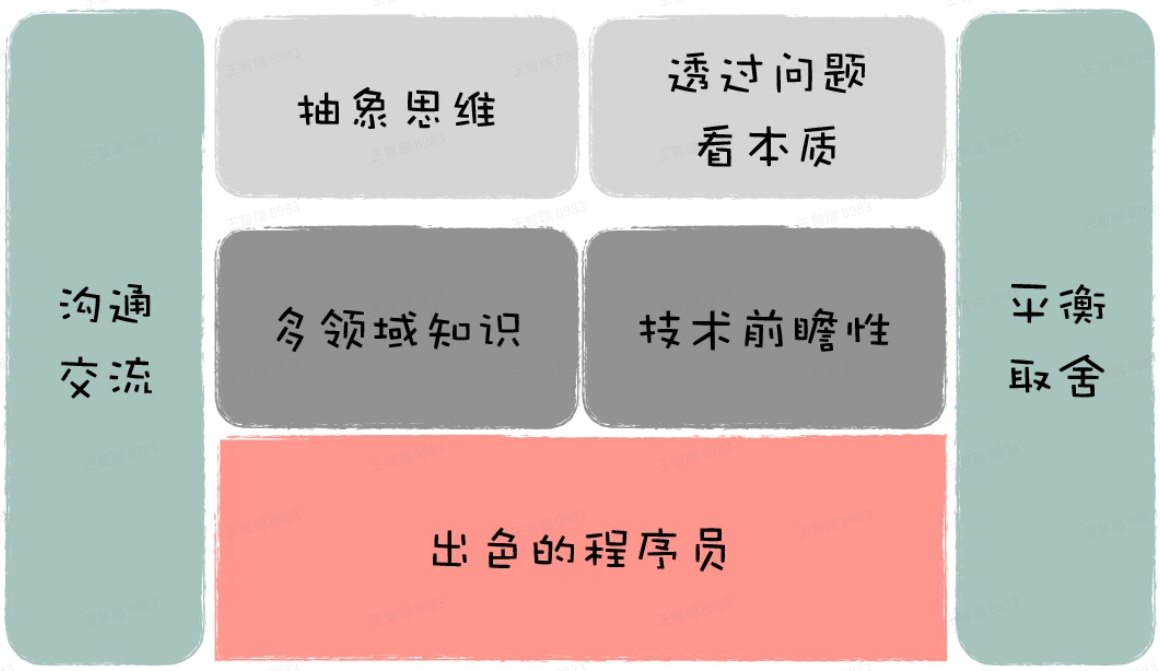
李腾飞 2022年12月15日

有了业务的定位，我们需要一套工程方案解决业务需求支撑业务发展。在实际的日常开发中因为团队内新人对整个系统不熟悉会出现重复开发，也不利于大家在实现技术方案的时候做参考。

技术架构就是讲清楚系统由哪些硬件、操作系统和中间件组成，它们是如何和我们开发的应用一起配合，应对各种异常情况，保持系统的稳定可用



什么是好的架构师



1. TA 必定是一个出色的程序员，写的一手好代码。
2. 架构师要有技术的广度（多领域知识）和**深度（技术前瞻）**
3. 对主流公司的系统设计非常了解，知道优劣长短，碰到实际问题，很快就能**提供多种方案供评估**。
4. 需要有思维的高度，具备**抽象思维能力**。抽象思维是架构师最重要的能力，架构师要善于把实物概念化并归类。比如，面对一个大型的 B2C 网站，能够迅速抽象为采购 -> 运营 -> 前台搜索 -> 下单 -> 履单这几大模块，对系统分而治之。架构师还需要有思维的深度，能够透过问题看本质。
5. **透过问题看本质是由事物的表象到实质，往深层次挖掘**。比如，看到一段 Java 代码，知道它在 JVM（Java Virtual Machine，Java 虚拟机）中如何执行；一个跨网络调用，知道数据是如何通过各种介质（比如网卡端口）到达目标位置。透过问题看本质，可以使架构师能够敏锐地发现底层的真实情况，以端到端闭环的方式去思考问题，能够识别系统的短板并解决它。
6. 能落地的架构才是好架构，**所以架构师还需要具备良好的沟通能力（感性）**，能确保各方对架构达成共识，愿意采取一致的行动；**而良好的平衡取舍能力（理性），可以确保架构在现有资源约束下是最合理的，能让理想最终照进现实**。

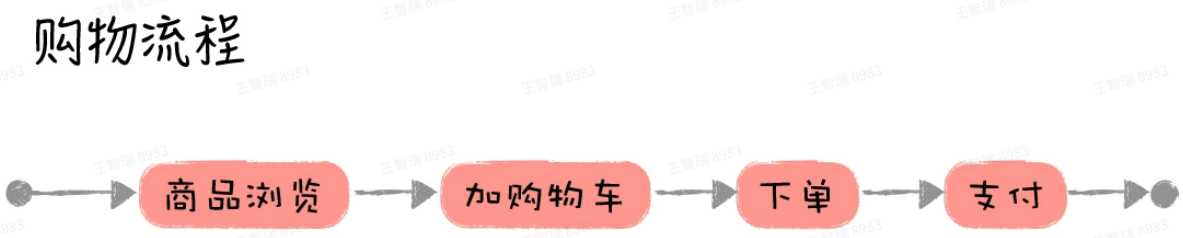
技术架构

李腾飞 2022年12月15日

因为我们团队是偏业务的团队，实际运用中技术架构与应用架构的区分是比较难的。经常出现的是一个图又是应用架构又是技术架构。

架构师跟产品经理的区别

产品经理是站在用户的角度进行需求分析，而业务架构师是站在开发者的角度定义系统内部结构

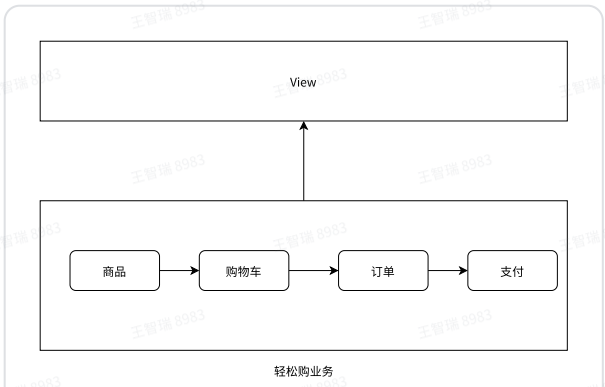


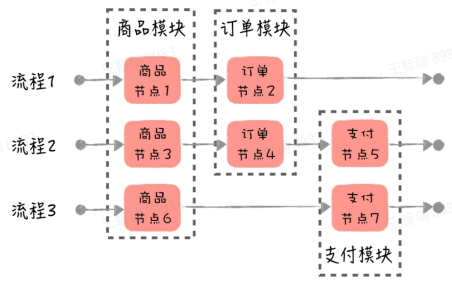
经过产品经理的工作，大量零散的原始需求经过梳理和关联，变成一系列有序的业务流程，以及流程里面的业务步骤（业务步骤也称之为业务节点），然后产品经理把这一系列的业务流程和业务节点以用户界面的方式定义出来，总的来说，产品经理定义了系统的外表。

什么是业务架构

业务架构的职责

对业务架构师来说，TA 的工作，就是把业务流程和节点打散，按照业务域的维度来划分系统模块，并定义这些模块之间的关系，最终形成一个高度结构化的模块体系。这样，开发人员既容易理解业务，又方便落地系统。





产品经理和业务架构师的工作既有区别又有联系，简单地说，产品经理定义了系统的外观，满足了用户；业务架构师在此基础上，进一步定义了系统的内部模块结构，满足了开发人员。

什么是技术架构

技术架构的职责

应用代码怎么组织（比如模块划分和服务分层），那主要是业务架构的事。技术架构就要选择和组合各种软硬件，再结合我们开发的应用代码，来解决系统非功能性需求

系统非功能性需求

系统的高可用

系统的高性能

系统的可伸缩

技术架构与业务架构的区别

因为**业务架构解决的是系统功能性问题**，我们更多的是**从人出发**，去更好地理解系统；而**技术架构解决的是系统非功能性问题**，我们在识别出业务上的性能、可用性等挑战后，更多的是**从软硬件节点的处理能力出发**，通过合理的技术选型和搭配，最终实现系统的高可用、高性能和可伸缩等目标

什么是业务架构 业务架构的职责

李腾飞 2022年12月15日

这个是新人最容易出现的问题，产品说我需要一个功能，流程是这样的 A，B，C，D。结果 RD 就实现了 A，B，C，D 接口，这些接口的编码难度确实很低，他会认为有些技术方案的时间我可能功能都已经上线了，为什么还要形式化的写个文档呢？实际 A 可能是判断用户对某个空间有没有 Admin 权限，RD 就会觉得很简单，空间表读出来看他是否在

我怎样做才能行！

> 系统=模型+关系，不要去理解它，多做练习，在实战中去感受它

领域驱动设计（DDD）

> DDD 是一种设计思想，它可以指导业务建模和微服务设计，通过领域驱动设计方法定义领域模型，从而确定业务和应用边界，保证业务模型与代码模型的一致性

微服务与DDD的关系

DDD 是一种架构设计方法，微服务是一种架构风格，两者从本质上都是为了追求高响应力，而从业务视角去分离应用系统建设复杂度的手段。两者都强调从业务出发，其核心要义是强调根据业务发展，合理划分领域边界，持续调整现有架构，优化现有代码，以保持架构和代码的生命力，也就是我们常说的演进式架构

微服务目标

运行时的进程间通信、容错和故障隔离，实现去中心化数据管理和去中心化服务治理，关注微服务的独立开发、测试、构建和部署

DDD 目标

从业务领域视角划分领域边界，构建通用语言进行高效沟通，通过业务抽象，建立领域模型，维持业务和代码的逻辑一致性

名词解释

DDD 的知识体系提出了很多的名词，像：领域、子域、核心域、通用域、支撑域、限界上下文、聚合、聚合根、实体、值对象等等，非常多。这些名词，都是关键概念，但它们实在有些晦涩难懂，可能导致你还没开始实践 DDD 就打起了退堂鼓

Admin 字段里就行了。那更好的方式应该去访问系统底层的权限模块，而不是读 DB，这种情况主要因为他对整个系统的应用架构不够熟悉，二就是没有意识架构的重要性。
展开

> 我怎样做才能行！

李腾飞 2022年12月16日

技术评审是手段，目的还是期望组内大家能对整个系统的演进实时对齐和了解，类似避免出现读表而不去请求服务的场景。我们需要在组内对齐关于架构实际的一套理论，指引大家朝着一个方向走。

> [图片]

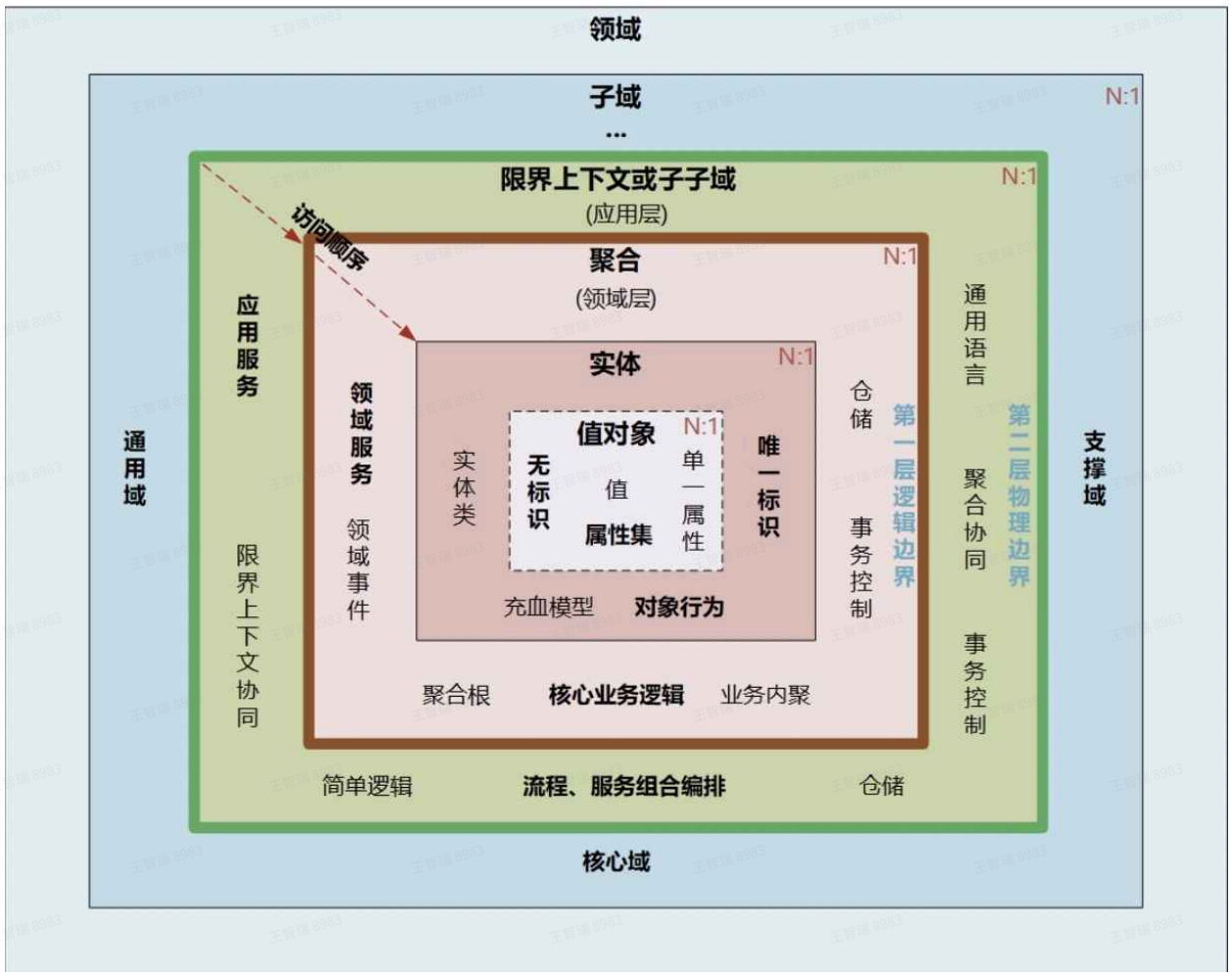
李腾飞 2022年12月16日

无论实现一个 API，又或者实现一个服务或者系统，本质是把一个问题进行梳理找到对应解决方案，但是这个思考的过程和模式是一样的。

> 领域驱动设计（DDD）

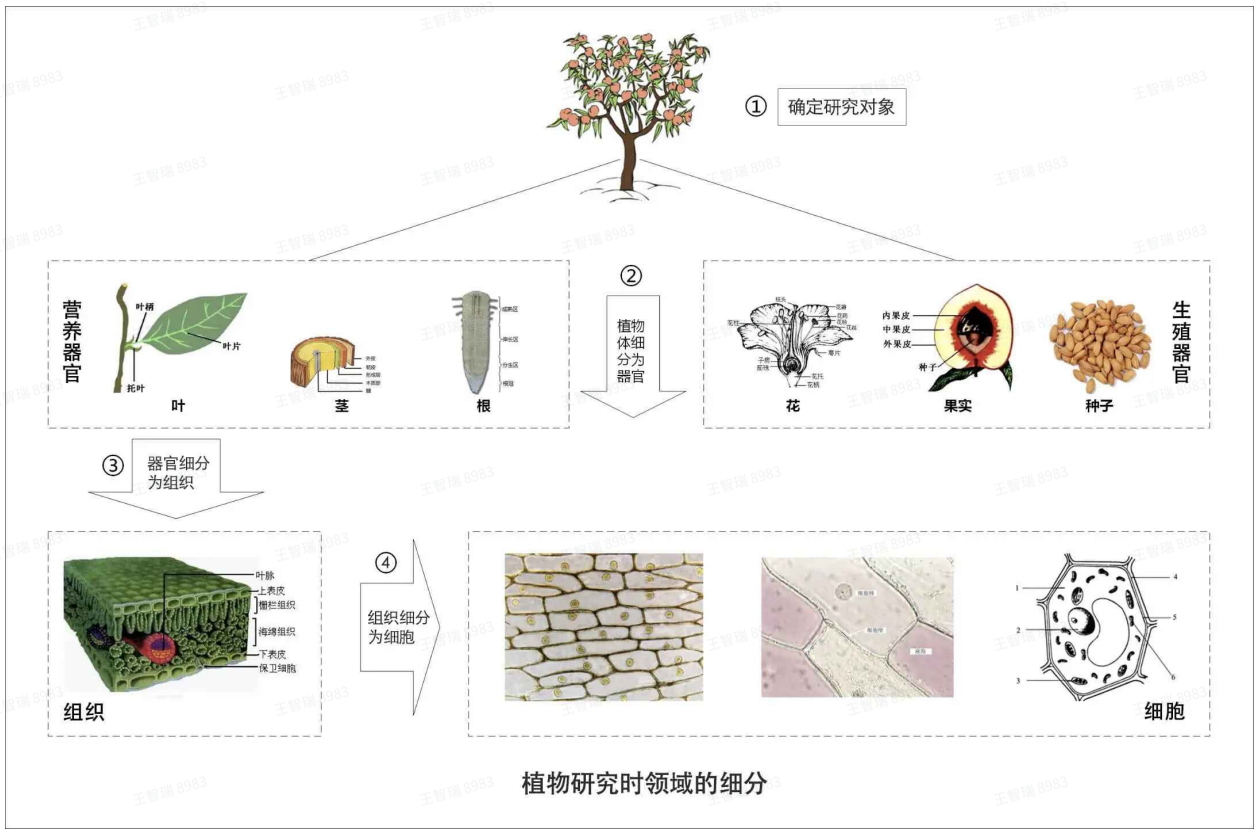
李腾飞 2022年12月16日

实例化： 店铺域架构设计文档（旧版）



领域与子域的区别

领域就是这个边界内要解决的业务问题域，两个解释有一个共同点——范围



植物研究时领域的细分

如何理解核心域、通用域和支撑域?

在领域不断划分的过程中，领域会细分为不同的子域，子域可以根据自身重要性和功能属性划分为三类子域，它们分别是：核心域、通用域和支撑域

核心域

决定产品核心竞争力的子域是核心域，它是业务成功的主要因素和核心竞争力

通用域

被多个子域使用的通用功能子域是通用域

支撑域

不包含产品核心竞争力的功能，也不包含通用功能的子域，它就是支撑域

限界上下文

领域与子域的区别

李腾飞 2022年11月10日

研发流程=热修+多仓+Small MR

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。每个领域模型都有它对应的限界上下文，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

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

实体

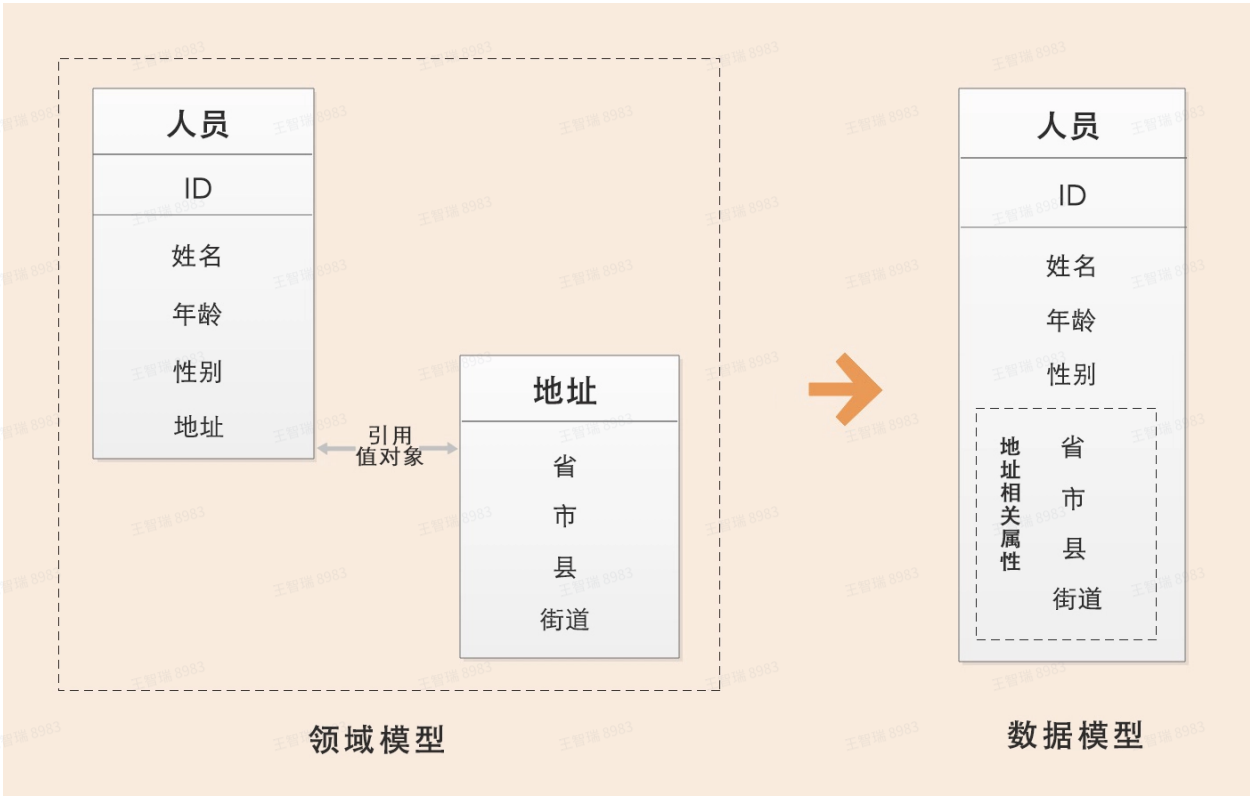
在 DDD 中有这样一类对象，它们拥有唯一标识符，且标识符在历经各种状态变更后仍能保持一致。对这些对象而言，重要的不是其属性，而是其延续性和标识，对象的延续性和标识会跨越甚至超出软件的生命周期。我们把这样的对象称为实体

```go
type OptimusUserFeishuUser struct {
    ID              int64      `gorm:"column:id" json:"id"`
    CreateTime      *time.Time `gorm:"column:create_time" json:"create_time"`
    UpdateTime      *time.Time `gorm:"column:update_time" json:"update_time"`
    Name            string     `gorm:"column:name" json:"name"`
    Email           string     `gorm:"column:email" json:"email"`
    NamePinyin      string     `gorm:"column:name_pinyin" json:"name_pinyin"`
    LarkUserID      string     `gorm:"column:user_id" json:"user_id"`
    Avatar          string     `gorm:"column:avatar_url" json:"avatar_url"`
    OptimusBotOpenID string    `gorm:"column:optimus_bot_open_id" json:"optimus_bot_open_id"`
}

func (u OptimusUserFeishuUser) TableName() string {
    return OptimusUserFeishuUserTableName
}
```

值对象

值对象描述了领域中的一件东西，这个东西是不可变的，它将不同的相关属性组合成了一个概念整体。当度量和描述改变时，可以用另外一个值对象予以替换



> 决定产品核心竞争力的子域是核心域，它…
>
> 李腾飞 2022年11月10日
> optimus

> 被多个子域使用的通用功能子域是通用域
>
> 李腾飞 2022年11月10日
> meta
>
> 李腾飞 2022年12月18日
> bytedance.bits.meta

> 不包含产品核心竞争力的功能，也不包含…
>
> 李腾飞 2022年11月10日
> cronjob
>
> 李腾飞 2022年12月18日
> bytedance.bits.cronjob

> 我们可以将限界上下文拆解为两个词：限…
>
> 李腾飞 2022年11月10日
> optimus 领域里定义了 mr 模型，bits mr 模型

> 在 DDD 中有这样一类对象，它们拥有唯…
>
> 李腾飞 2022年11月10日
> 在 meta 域中请求了 user 模型数据，传递到任何域或系统中

```go
type Department struct {
    Id   string `json:"id"`
    Name string `json:"name"`
}

type QueryEmployeesResponse struct {
    Employees []*Employee `json:"employees"`
    Success   bool        `json:"success"`
}
```

简单来说，值对象本质上就是一个集合

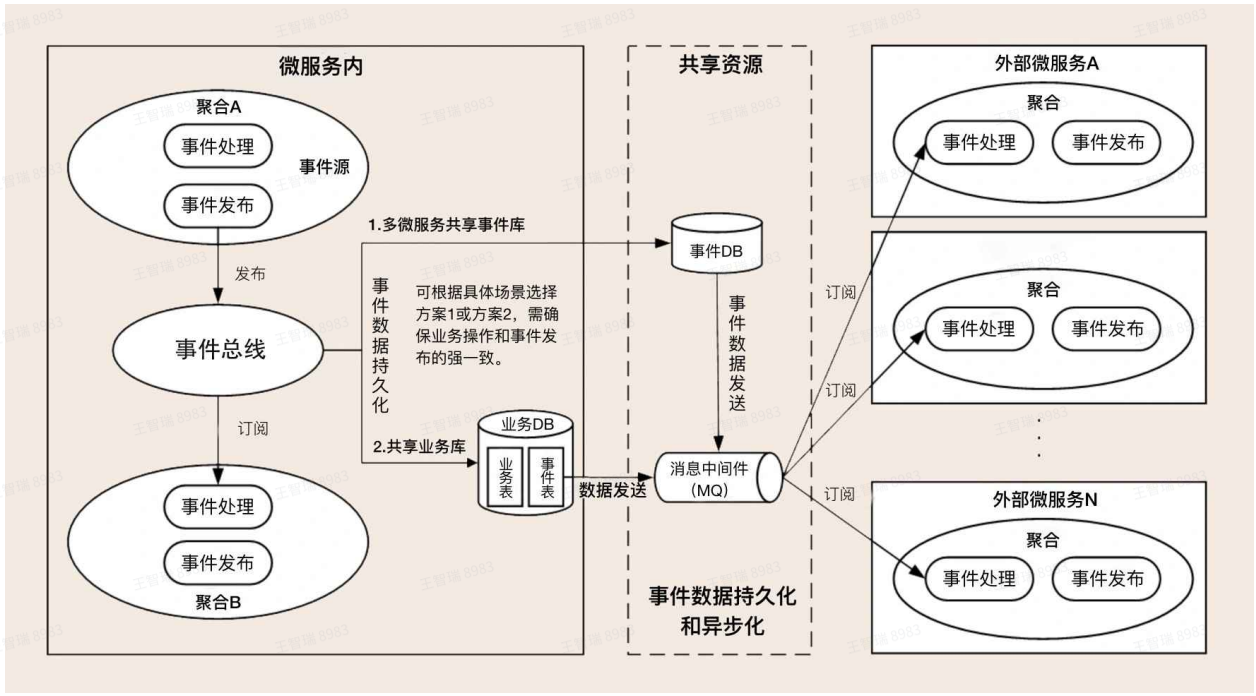
领域模型内的实体和值对象就好比个体，而能让实体和值对象协同工作的组织就是聚合，它用来确保这些领域对象在实现共同的业务逻辑时，能保证数据的一致性

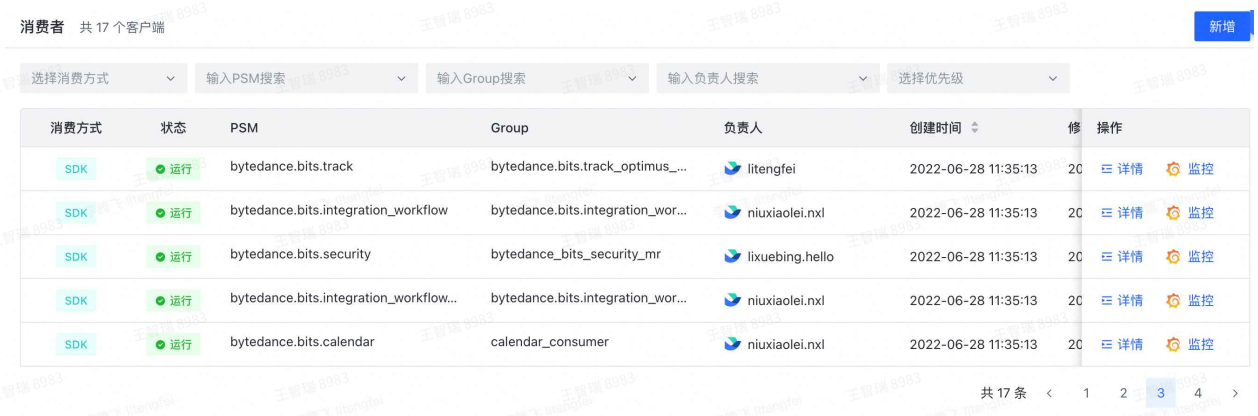
聚合根

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。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

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。





分层架构

id 不会变

[图片]

李腾飞 2022年11月10日

这就是实体的代码形态，对应的数据库表就是实体的数据库形态

[图片]

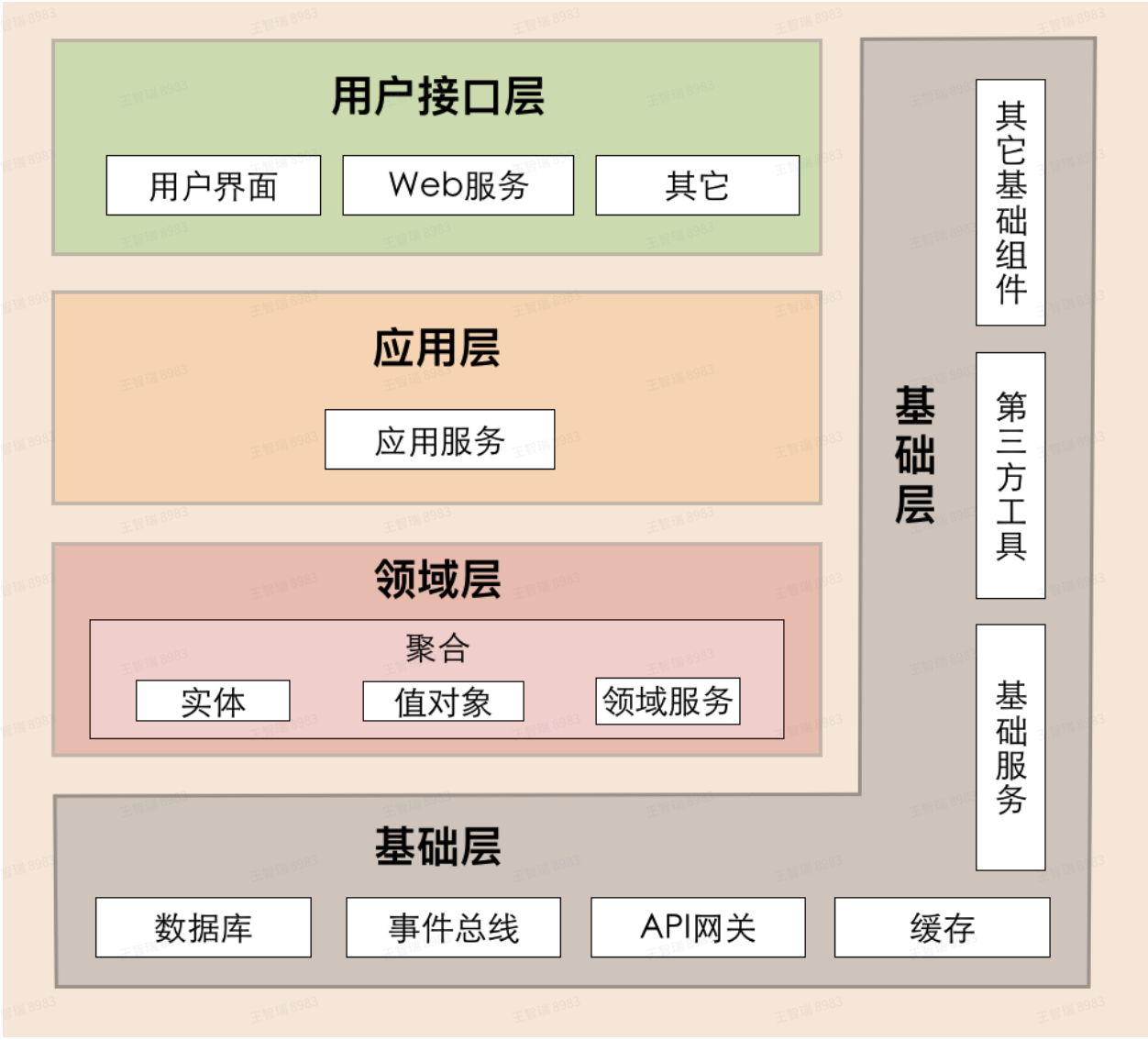
李腾飞 2022年11月10日

人员是实体，地址是值对象

[图片]

李腾飞 2022年11月10日

queryEmployeesResponse 就是值对象的代码形态



用户接口层

用户接口层负责向用户显示信息和解释用户指令。这里的用户可能是：用户、程序、自动化测试和批处理脚本等等。

应用层

应用层是很薄的一层，理论上不应该有业务规则或逻辑，主要面向用例和流程相关的操作。但应用层又位于领域层之上，因为领域层包含多个聚合，所以它可以协调多个聚合的服务和领域对象完成服务编排和组合，协作完成业务操作。此外，应用层也是微服务之间交互的通道，它可以调用其它微服务的应用服务，完成微服务之间的服务组合和编排。

领域层

领域层的作用是实现企业核心业务逻辑，通过各种校验手段保证业务的正确性。领域层主要体现领域模型的业务能力，它用来表达业务概念、业务状态和业务规则。领域层包含聚合根、实体、值对象、领域服务等领域模型中的领域对象。这里我要特别解释一下其中几个领域对象的关系，以便你在设计领域层的时候能更加清楚。首先，领域模型的业务逻辑主要是由实体和领域服务来实现的，其中实体会采用充血模型来实现所有与之相关的业务功能。其次，你要知道，实体和领域服务在实现业务逻辑上不是同级的，当领域中的某些功能，单一实体（或者值对象）不能实现时，领域服务就会出马，它可以组合聚合内的多个实体（或者值对象），实现复杂的业务逻辑。

基础层

基础层是贯穿所有层的，它的作用就是为其它各层提供通用的技术和基础服务，包括第三方工具、驱动、消息中间件、网关、文件、缓存以及数据库等。比较常见的功能还是提供数据库持久化。基础层包含基础服务，它采用依赖倒置设计，封装基础资源服务，实现应用层、领域层与基础层的解耦，降低外部资源变化对应用的影响

微服务架构模型

整洁架构

> 领域模型内的实体和值对象就好比个体，…

李腾飞 2022年12月16日

从业务架构->应用架构->技术架构->领域划分->领域模型 整个过程都是自上而下的一套处理流程，比如需要实现一个用户登录注册的功能。现在从业务架构上看为什么需要用户登录注册，支撑哪块业务。再到应用架构，注册登录应该写到哪一层，通用吗，写到哪个服务里。技术架构考虑用户数据存储方案是什么？redis or mysql or hdfs 并发写怎么解决。领域划分确认我们需要一个人员的域，需要一个 user 模型作为这个与的领域模型，user 应该有 username, passsword ,nickname, avatar 等字段。等我们梳理到领域模型后就会发现整个功能基本已经实现了，剩下的就真的是 CRUD 了。对 user 模型实现对应的增删改查接口

展开

> 聚合根

李腾飞 2022年11月10日

Optimus 域下的 mrinfo 聚合根。整个 bits 系统间只能通过 optimus 与获取 mr 模型的数据

> [图片]

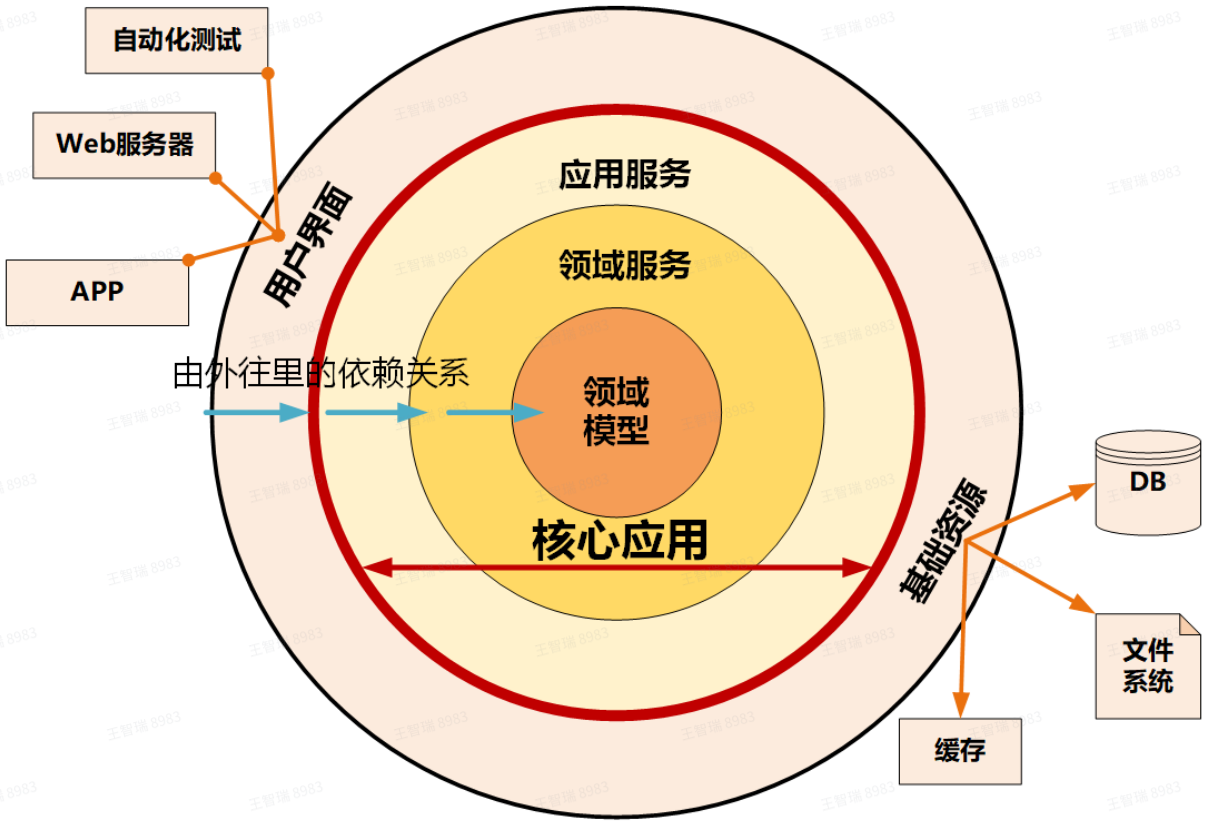
李腾飞 2022年12月15日 （编辑过）

Bits 系统的事件总线

李腾飞 2022年12月15日

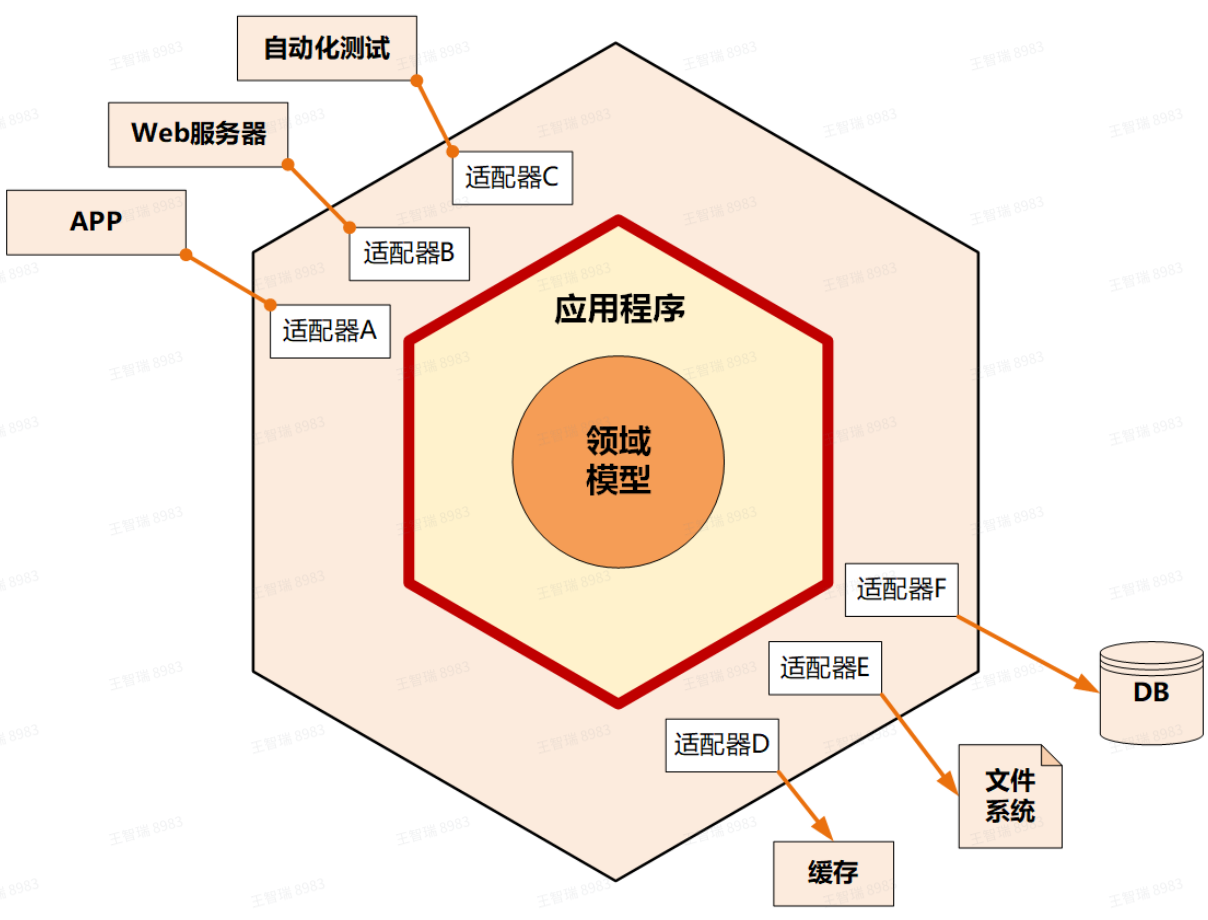
Bits 核心事件总线

整洁架构又名“洋葱架构”。为什么叫它洋葱架构？看看下面这张图你就明白了。整洁架构的层就像洋葱片一样，它体现了分层的设计思想



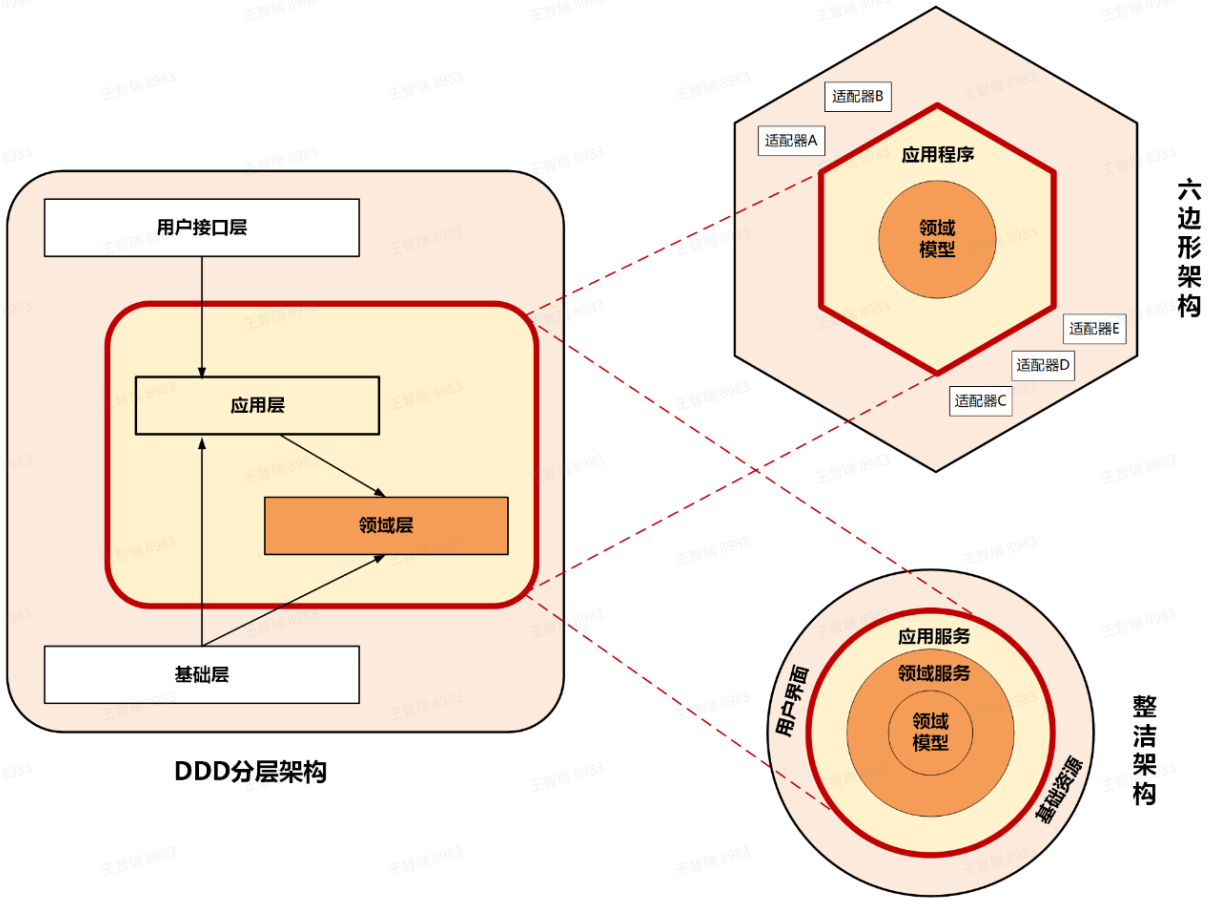
六边形架构

六边形架构又名“端口适配器架构”。追溯微服务架构的渊源，一般都会涉及到六边形架构。六边形架构的核心理念是：应用是通过端口与外部进行交互的。我想这也是微服务架构下 API 网关盛行的主要原因吧。



差异

虽然 DDD 分层架构、整洁架构、六边形架构的架构模型表现形式不一样，但你不要被它们的表象所迷惑，这三种架构模型的设计思想正是微服务架构高内聚低耦合原则的完美体现，而它们身上闪耀的正是以领域模型为中心的设计思想



Q&A

1. 技术架构设计的目标是什么
   a. 安全性
   b. 可复用
   c. 高性能
   d. 高可用

2. 哪些架构设计原则和系统的高可用没有直接关系
   a. 计算可并行
   b. 无状态
   c. 可虚拟化部署
   d. 使用成熟技术

3. 从服务的划分层次看，以下哪个不属于基础业务服务
   a. 用户服务
   b. 商品服务
   c. 库存服务
   d. 下单服务

4. 哪些是消息系统典型的应用场景
   a. 流量消峰
   b. 系统解耦
   c. 业务降级
   d. 异步处理

5. 哪个选项不属于网关的典型功能
   a. 限流
   b. 安全验证

c. 业务编排

d. 路由

6. 在领域不断划分的过程中，领域会被细分为不同的子域，根据子域自身的重要性和功能属性你可以将它们划分为哪几类子域？

a. 核心子域

b. 通用子域

c. 普通子域

d. 支撑子域

7. 在领域建模时，一般会分析出哪些领域对象

a. 领域事件

b. 实体

c. 值对象

d. 聚合根