Hello，这里是爱 Coding，爱 Hiphop，爱喝点小酒的 AKA 柏炎。

不知道你有没有这样一种感觉，不管去哪家公司，只要你接触的项目是多人维护且存在了 2 年以上的，这个代码里面总有各种各样的“坏味道”。

俗称：“前人挖坑，后人填坑。”

你仔细想想，这是不是你开发需求的常规操作： 先动手建立表结构，再针对表数据进行 CRUD 组装出功能点。

这种“屎山”代码的特点就是：业务逻辑极其混乱，为了完成一个功能点，代码就是往上堆。

发现没有，你并不是从业务角度为起点去开发整个系统，而是以数据为起点去开发整个系统。数据库表又是一个很敏感的东西，不会动不动就大改表结构。那为了能够满足日益复杂的需求，你只能够加表、加字段、加 if/else 的逻辑嵌套。久而久之，你的代码就开始像积木搭建的房子，层层往上堆，如果其中有一块没搭好，可能瞬间就崩塌了。

那到底是什么原因导致了祖传代码的产生呢?

归根到底就是我们没有 从业务出发自顶向下 地去思考一个系统里面各个功能模块的职责与能力。

业务（领域）驱动设计，这就是 DDD。 它的宗旨就是 内聚与解耦 ，这也正是这本小册将为你输出的核心知识点。

这就带你从业务出发，拿捏架构设计!

我是谁

我花名叫**柏炎**。

干过测试，写过业务系统，现在主要在负责基础架构开发。

工作这些年我做过测试架构、MVC 的业务架构，当然也 落地过 DDD 的六边形架构 。

如果你问我是不是一个资深的技术大佬，那我肯定不是。我只不过是一个“摸着石头过河的瞎子”，但是也正因为如此，我在基础架构落地的路上经过了实战的考验，也经历了业务的打磨，希望能从更加贴合 业务与实践 的角度为大家提供架构落地思路，与大家共同成长。

不过嘛，相比较把自己定义成一个程序员，我还是更喜欢定义自己是个热爱 HipHop 的老酒鬼，哈哈哈。

为什么跟我学?

纵观网上关于 DDD 的资料或者 Github 上的开源项目，不是在疯狂堆叠 DDD 的基础概念，就是直接给你一个演示 Demo。稍微好点的，概念里面配几张图或者演示 Demo 里面多加点注释，总是缺少了一个推导的过程。

就像你妈打你，从不讲道理。现在有 DDD 这个东西，你就照着“我”这么搞就行。

但，为什么这么搞？这么搞的好处是什么？……这样相关的内容少之又少。

我们本小册的行文采用 对比思路 ，从 MVC 三层架构或者我们习惯的开发模式出发，看看会存在什么样的问题，然后再逐步对比看如何使用 DDD 的内聚与解耦的思想来解决这些问题。

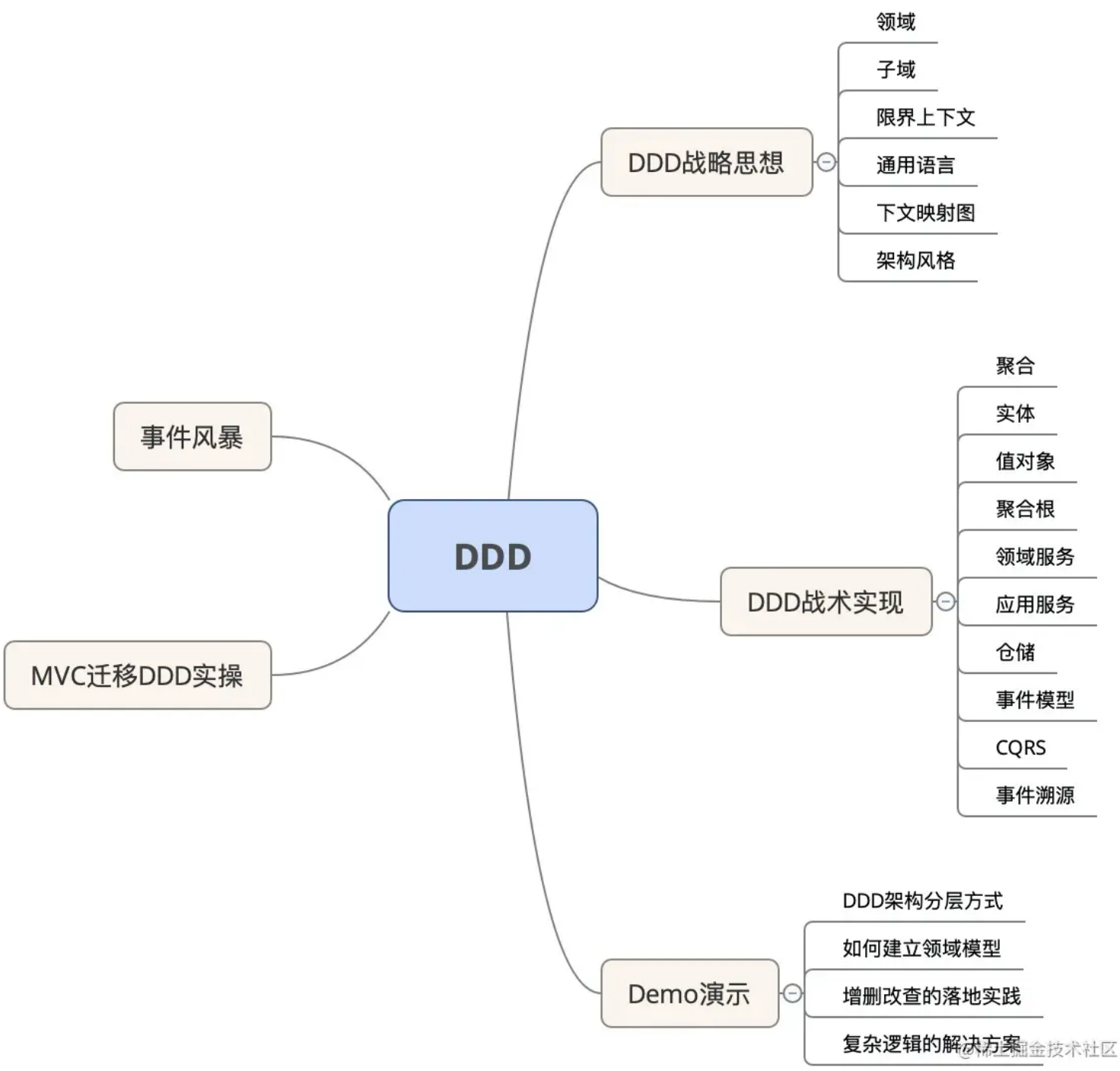
为什么大家对 DDD 都是听过、看过却不认识？那是因为概念没有对应到实际的代码中。因此，本小册在系统地为你分析和讲解完 DDD 的概念与理论之后，还会给出实际的 Demo 讲解，进而让你彻底掌握 DDD 的核心思想与落地操作。

什么？MVC 系统迁移 DDD 成本太大，操作太麻烦？别担心！DDD 本身就是一个系统架构上的战略思想，完全按照其规范去开发系统只是它的战术目标，我们完全可以选择其中好落地、好实现的部分去优化系统。

总之，学完这本小册，铲平“屎山”固然困难，但是给它掘掉点是完全没有问题的。

你能学到什么？

以下就是本小册知识点的思维导图：



可以看到，小册整体的设置思路是这样的：

- 从使用 DDD 的原因出发，逐步为你讲解 DDD 的战略思想与战术实践；
- 理论结束之后，我将为你一步步讲解如何搭建 DDD 的项目，以及如何落地实践 DDD；
- 最后，再手把手指导你如何将系统从 MVC 架构迁移至 DDD 架构。

总之就是，不停留在理论层面，而是系统性地学习 DDD，从根上理解它、攻克它、落地它。

打打鸡血

架构设计这东西听起来就是一个很虚的东西。你实际工作中可能碰不到，但是面试的时候还经常问到，一问你还老答不好。

有的人工作三年已经能够上手系统架构设计，并写出优美的代码了，而你工作五年，却每天只能 CRUD，感觉 35 岁的程序员危机在提前向你招手。

虽然这段话像在贩卖焦虑，但不可否认 DDD 确实是一些中大厂的架构趋势。

我不能保证你学完本小册就一定能够成为架构师或者进入大厂，但至少可以让你写出的代码优雅且好维护，出去面试也能让你的简历有个闪光点，再不济还能在同事那里吹水搞过架构，哈哈哈～

加油吧，少年！

最后希望小册能够在编程思维上给你带来一些正面的影响。 当然，如果你在小册中遇到了任何问题，都欢迎你在评论区与微信群与我分享，我们共同进步。

留言

输入评论（Enter换行，Ctrl + Enter发送）

发表评论

全部评论（19）

火龙果天天写bug 14天前

开始！！

点赞 回复

减肥多喝水 后端 3月前

“先动手建立表结构，再针对表数据进行 CRUD 组装出功能点。” 太真实了

点赞 回复

南方者 要寻找躲在日常的小快... 1年前

学之前：DDD？滴滴滴？爹爹爹？ 学完后：DDD？弟弟弟！拿来吧你！

点赞 回复

雨季不再来 JAVA开发 1年前

开始学习，加油

点赞 回复

EthanWinters 软件开发 @ 后台开发 1年前

跟着学习，语言风趣幽默，不过确实，我妈打我从不需要理由 ，这个是实话

点赞 1

柏炎 （作者） 1年前

多谢支持

点赞 回复

西尔James Python @ 创业公司 1年前

点赞 回复

艺术不过派大星 AI模型训练师 1年前

点赞 回复

用户9020176671... 1年前

点赞 回复

peigo 前端工程师 1年前

目前的DDD大部分是后台系统，有没有前端工具类系统的经验，类似figma，云端建模这样的大型系统，越到后面系统就越复杂。使用DDD一般该怎么设计比较好怎么分领域？

2 2

武码公社 1年前

思维其实是可以相通的，前端业务代码一样可以使用DDD的思想，找到领域边界，领域事件等等来合理拆分代码，实现高内聚低耦合，以及开闭原则，毕竟js尤其是ts也是面向对象的。

点赞 回复

柏炎 （作者） 1年前

前端我接触的并不是很多，不太清楚你们的业务逻辑处理模式。在后端中DDD的思路就是把同一业务领域的逻辑聚合在聚合根（可以理解为vue的一个组件库？）中，你的聚合根表示了这个业务。这样后续复杂的业务需求我只需要不断的调用聚合根里面的方法组合就好了，不需要重复定义逻辑，而且功能点都只是对业务方法的调用，逻辑内聚在聚合中，调用点的代码就是你的功能点体现。我想这上面的逻辑处理方式是不限于前后端的，是一种通用的思路，如果感兴趣的，可以继续阅读一下小册的后文，也许会对你有帮助

点赞 回复

WaterSports 1年前

可是实际工作中，人们都说要实现6个9，但是需求砸下来的时候时间又很紧迫，比如说过两天就要。这种时候从数据出发思考，以及写函数，而不是想怎么按结构体来分类的方式反而是比较快速和简单并且符合人的思路的。重点是不给开发很多时间思考。就是说一下需求然后过几天就要结果

1 2

柏炎 （作者） 1年前

DDD使用是需要成本的，对产品的可持续性要求比较高。初期建模是比较困难的，但是一旦初期模型建立之后，后续需求加进去，迭代其实是很丝滑的。但是如果你本身项目就是类似于那种定制性开发的，一天到晚紧急上线的系统，需求本身就存在不连贯的情况，那使用DDD肯定不合适的。你的这个疑问我在小册中都做了一一的分析与说明，可以继续看看哦~

1 回复

小飞飞66 10月前

同感

点赞 回复

meahu 前端工程师 1年前

说的这么好，怒赞

1 1

柏炎 （作者） 1年前

谢谢支持

点赞 回复

鱼腩875 1年前

优秀，又风趣的技术大拿

点赞 1

柏炎 （作者） 1年前

谢谢支持

点赞 回复