



加入语雀，获得更好的阅读体验

注册 或 登录 后可以收藏本文随时阅读，还可以关注作者获得最新文章推送

立即加入



## 006.1-DDD实战领域驱动设计

源视频教程：<https://www.bilibili.com/video/BV1oT4y177Cs>

### 一、DDD初探

#### 1.什么是DDD?

- DDD是一种处理高度复杂领域的设计思想。
- 分离 技术实现的复杂性
- DDD不是架构，是架构设计方法论
- 通过便捷划分，复杂的业务领域简单化，帮助设计出清晰的领域和应用边界。

##### • 1.1-DDD的作用：

- **统一思想**：对各方业务、产品、开发对问题的认知，而不是开发和产品的统一，业务和产品的统一而产生分歧。
- **明确分工**：领域模型需要明确定义来解决方方面面的问题，而针对这些问题形成团队分工的理解。
- **反应变化**：需求是不断变化的，因此我们的模型也是在不断的变化的。领域模型则可以真实的反应这些变化。
- **边界分离**：领域模型与数据模型分离，用领域模型来界定哪些需求在什么地方实现，保持结构清晰。

#### 2.软件的架构模式

- 单机
- 集中式
- 分布式微服务架构

#### 3.微服务到底如何拆分?

- 宏观-战略：业务战略上的重点，从业务角度出发，建立业务领域模型，划分领域边界，建立通用语言的限界上下文。





## 4.学习DDD有什么收获?

- 一套完整的而且系统的设计方法。
- 架构设计能力——>领域专家——>架构师

## 5.名词介绍

- 领域：特定的范围（边界）或领域——>细分——>建立领域模型——>用代码实现问题（领域可以划分为子领域）
  - 子领域：更小的问题/更小的业务范围
    - 核心域（决定产品和公司产品竞争力的子域）
    - 通用域（同时被其他多个子域使用的通用功能的子域）：通用系统、认证、权限、日志，不存在太多的特性，和业务无关；
    - 支撑域（功能子域，必须的，但不包含决定产品核心竞争力的功能，不包含通用功能的子域）；具有企业特性，不具有通用性；
- 限界上下文：
  - 通用语言（定义上下文的含义）和限界上下文（定义领域边界）两个概念（战略设计）。
  - 通过团队交流达成共识、能够简单、清晰、准确描述含义和规则语言。

## 5.名词总结

- 核心域：是业务系统的核心价值所在，系统中的重中之重
- 通用域：是整个业务系统提供通用服务，其他子域都是他的消费者
- 支撑域：专注于业务系统的某一个重要业务，来支撑和完成业务系统

# 二、DDD核心概念

## 1.什么是值对象?

- 并不是编程语言中的值类型，可以看做值和对象，固定不变的；
- 用对象的方式来表述某个值；
- 值对象创建以后不允许修改，破坏地址的业务含义，概念完整性。

## 2.值对象的作用?

物理上独立出来，逻辑上仍然和实体是一体的。

## 3.值对象的优势?

根据场景简化数据库设计，性能提高；



## 5.注意

- 值对象无法单独存在，单独存在没有意义，他只是实体一部分的特征属性。
- 在领域建模阶段，忘掉数据库；

## 6.聚合与聚合根

- 聚合：
  - 强关联关系，整体与部分。
  - 将相关联的领域对象进行显示的分组，表达整体的概念。
- 聚合根：
  - 作用：为了避免由于复杂数据模型缺少统一的业务规则控制，导致聚合和实体之间，数据不一致的问题。

## 7.领域服务和应用服务

- 领域服务：表述领域行为，对应用行为的细化，具体的某一个环节
- 应用服务：表述应用行为，从开始到结束的每一个环节

## 8.聚合根、实体、值对象三者区别？

聚合根：全局的唯一标识、其他的状态信息都是可变

实体：聚合内部才有的本地标识

值对象：没有标识、只读

# 三、DDD分层架构

## 1.两种分层架构

- 严格分层架构
- 松散分层架构

## 2.分层架构优点

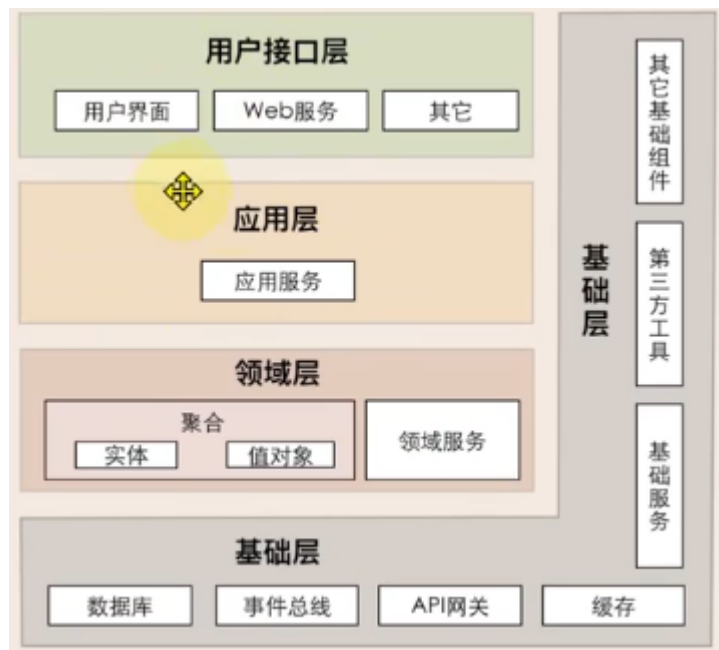
- 让程序结构更加的清晰
- 开发人员可以只关注结构中的某一层
- 很容易替换原有层次的实现
- 降低层与层之间的依赖
- 有利于标准化、各层逻辑的复用



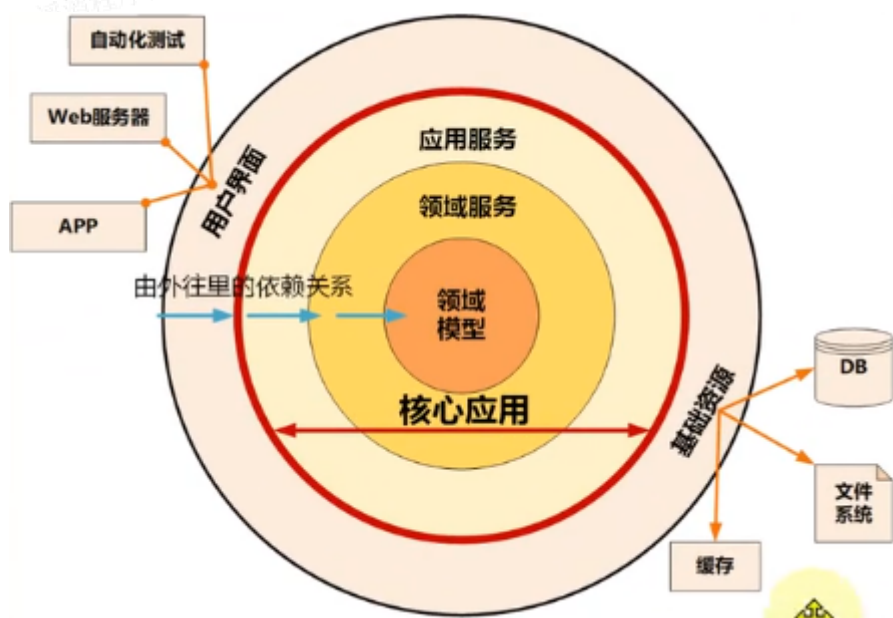
- 性能降低

#### 4.DDD封分层的架构模式

- DDD分层架构（四层架构）

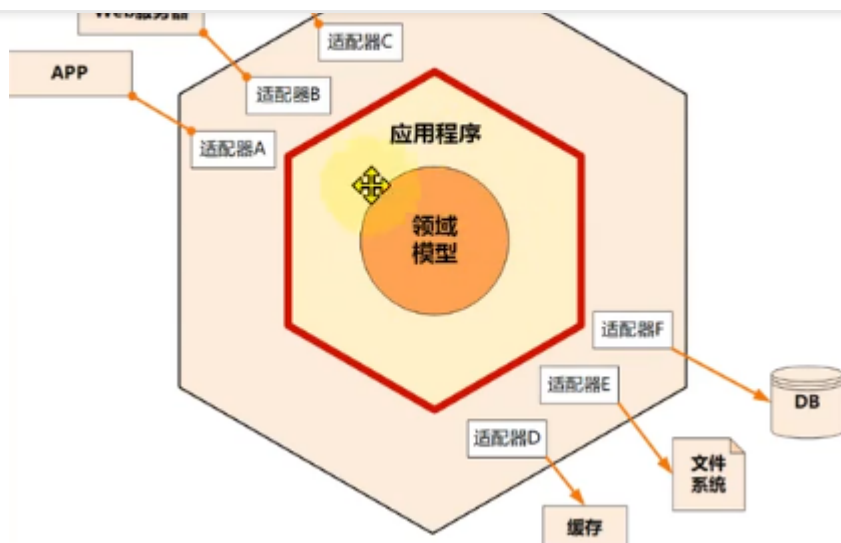


- 洋葱架构（也叫：整洁架构）

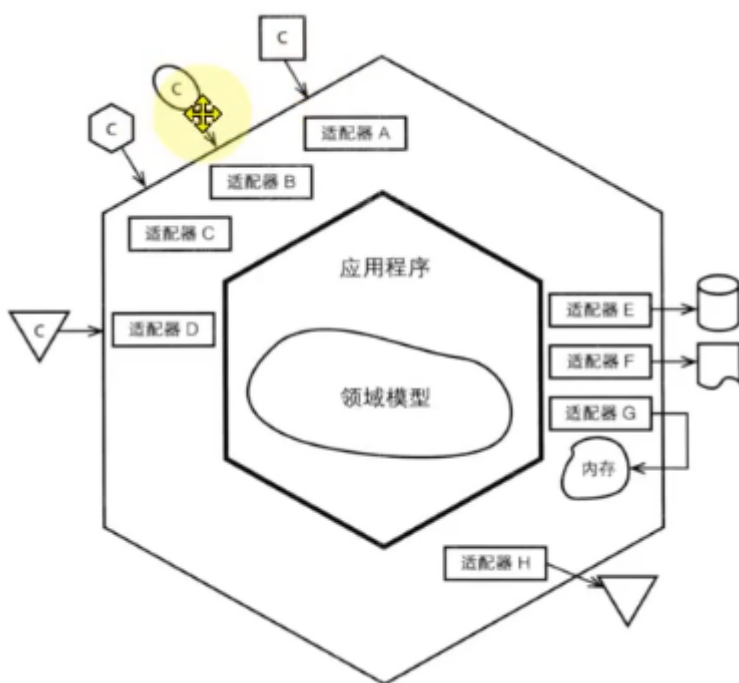


- 洋葱架构在项目中只有三层：核心应用层、用户界面层、基础设施层

- 六边形架构（也叫：端口适配器架构）

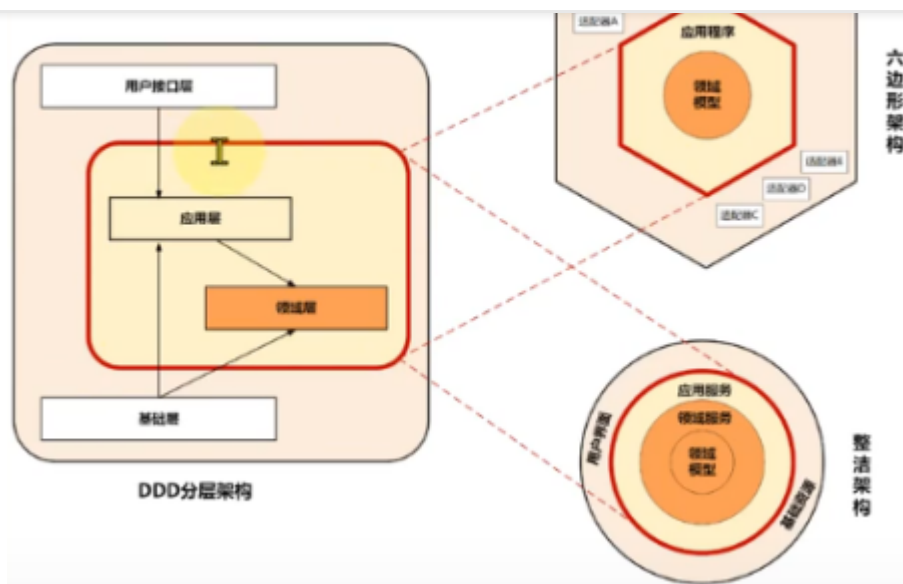


- 内六边形：红圈内实现业务逻辑
- 外六边形：基础资源、API、数据库等外部基础资源
- 什么是端口？
  - 每一条边可以理解为一个端口
  - 要么是输出、要么是输入。
  - 通过应用层、API进行交互
  - 一个边处理相同等不同的东西，比如一条边处理HTTP请求，一条边处理MQ请求。



- 微服务，也是领域模型

- DDD分层架构、洋葱架构（整洁架构）、六边形架构（端口适配器架构）



## 5.DDD变与不变

- 变：前端需求，用户界面
- 不变：领域模型

## 6.领域层

都是最小粒度的服务，应用层对这些服务进行编排，不同的编排有不同的行为。

## 7.什么是代码模型？

DDD设计思想，本身是没有标准的代码模型：快速理解到底什么是DDD，此刻层并不是完整的代码模型。

## 8.完整的DDD架构？

很多框架和组件、CQRS、命令总线、EDA、事件驱动、时间溯源、消息队列等等。

## 9.代码模型 (Web API项目)



- Application（应用层）-类库：存放应用服务，应用层向下就是领域层：用来组合和编排领域服务；向上就是接口层：为接口层提供数据。应用服务和事件相关的代码都会放到这层目录里。



码管理。

- infrastructure（基础设施层）-类库：为其他层提供通用的技术：数据库、身份验证、缓存等等基础设施。
- InterfaceWebAPI（外部调用的用户接口层）-WebAPI：处理用户发送的API请求

## 10.拿到系统开发任务，如何使用DDD思想去展开？

（1）整理各个领域对象，把聚合、实体、命令、事件、业务行文找出来，记录下来。这个过程称为：

### 事件风暴

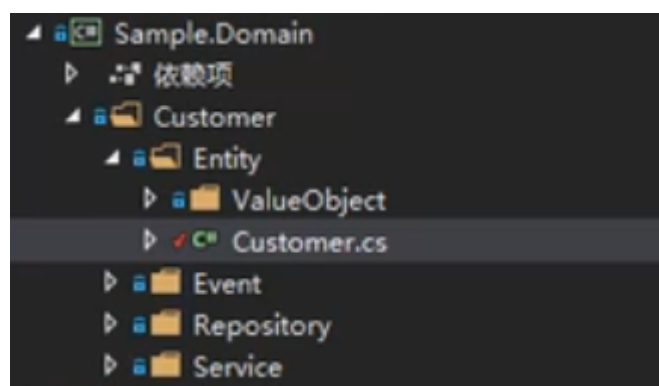
- 事件风暴：
  - 类似于需求调研，需求分析
  - 开一个会，需要各方参加：必须要有一个熟悉领域驱动领域的专家，以他为核心，进行的一项会议互动，所有人在领域专家的带领下熟悉业务、划分边界。

（2）假设经历好了事件风暴，整理好了客户的领域模型，产生这么一张表格。

例如：客户关系系统

- 客户领域
- 客户聚合：客户聚合根

领域模型	聚合	领域对象	领域类型
客户	客户	订单	聚合根
		地址	值对象
		创建客户信息	命令
		修改客户信息	命令
		查询客户信息	命令
		订单已创建	领域事件
	其它	...	...



（3）设计实体



（4）找出聚合根（聚合根来源于领域模型，聚合根通过仓储模式实现聚合类实体和值对象初始化）





## 006.1-DDD实战领域驱动设计 ⓘ

- 代码模型：逻辑上的存在

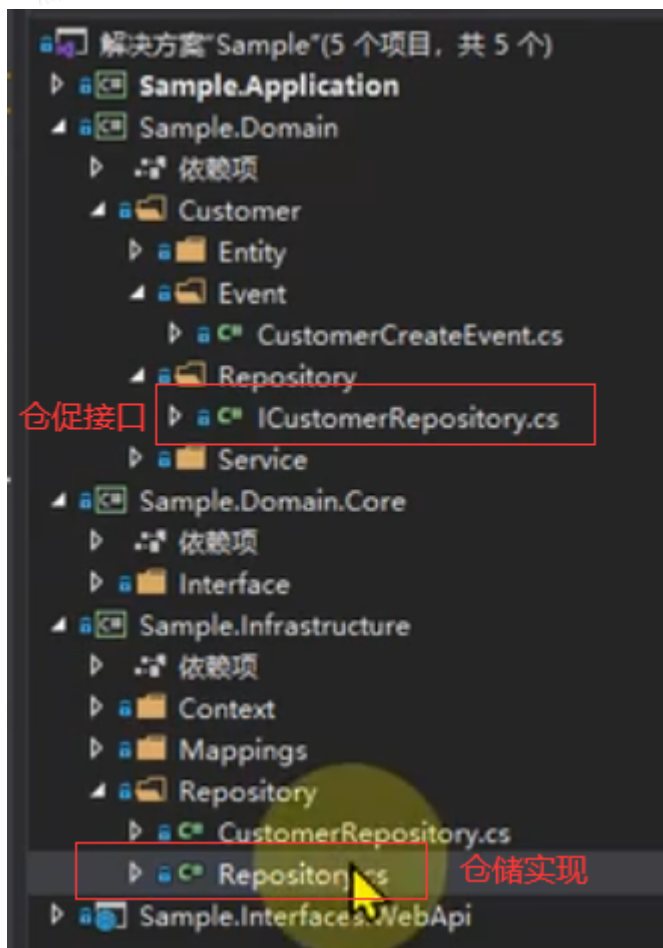
### (6) 事件类

- 内部-单体架构：范式跨聚合的通讯，都是用事件沟通

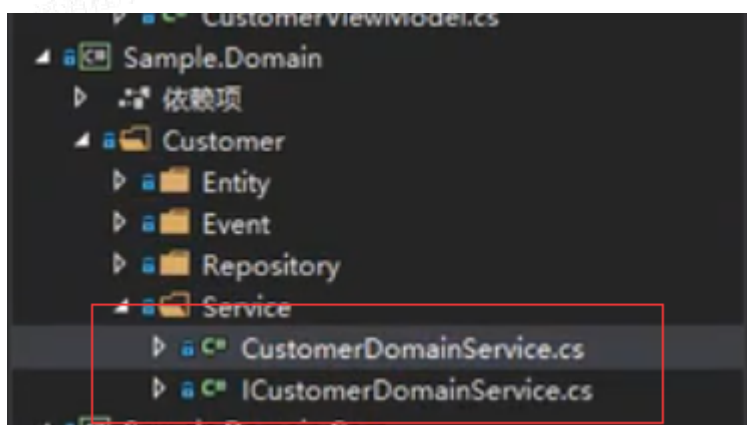


### (7) 仓储类（每一个聚合都对应一个仓储，一个仓储的接口，一个仓储的实现）

- 仓储接口放在【领域层】，仓储的实现放在【基础层】



### (8) 领域服务

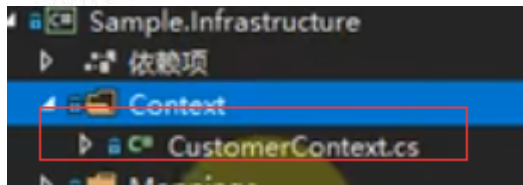


### (9) 基础层





- 有可以很多的上下文



## 11.微服务系统

- N个子域，每个子域对应着一个完整的DDD项目。
- 每个子域：很多聚合。
- 聚合之间通信：内部使用事件及时（进程内），外部使用消息队列。
- 注意：使用微服务，小项目做很累，用例流程大于20个的使用DDD。
  - 什么事流程？购物就是一个流程

## 12.注意

- DDD里面的概念，不一定都会出现在项目中，例如：事件在DDD中没有这个概念。
- 应用程序对外暴露的粗粒度API
- 领域驱动最重要的不是代码，最关键的是：**领域驱动的建模**

## 13.架构师的思维

- 全局



6 人点赞



🔔 滔滔程序猿 ⌚ 06-07 18:12 📄 839 💬 0 | 投诉

关注作者和知识库后续更新



滔滔程序猿

Motto如果说人生是自我...

关注



2\_后端开发笔记

C#、.NET、ASP.NET、.N...

关注





001.2-小学英语基础必备之语法全突破视频教程（更新...

源视频教程：<https://www.bilibili.com/video/BV184411T...>

002.3-NET5零基础到精通实战全集


源视频教程：<https://www.bilibili.com/video/BV1tK4y1j7...>

005.1-ASP.NET MVC5博客项目实战

源视频教程：<https://www.bilibili.com/video/BV1Z4411A...>

< 上一篇

下一篇 >

 加入语雀，参与知识分享与交流

注册 或 登录 语雀进行评论

立即加入

🗨 回复

分享到：  

注册 或 登录 语雀进行评论



语雀

| 关于语雀

使用帮助

数据安全

服务协议

| English

快速注册

