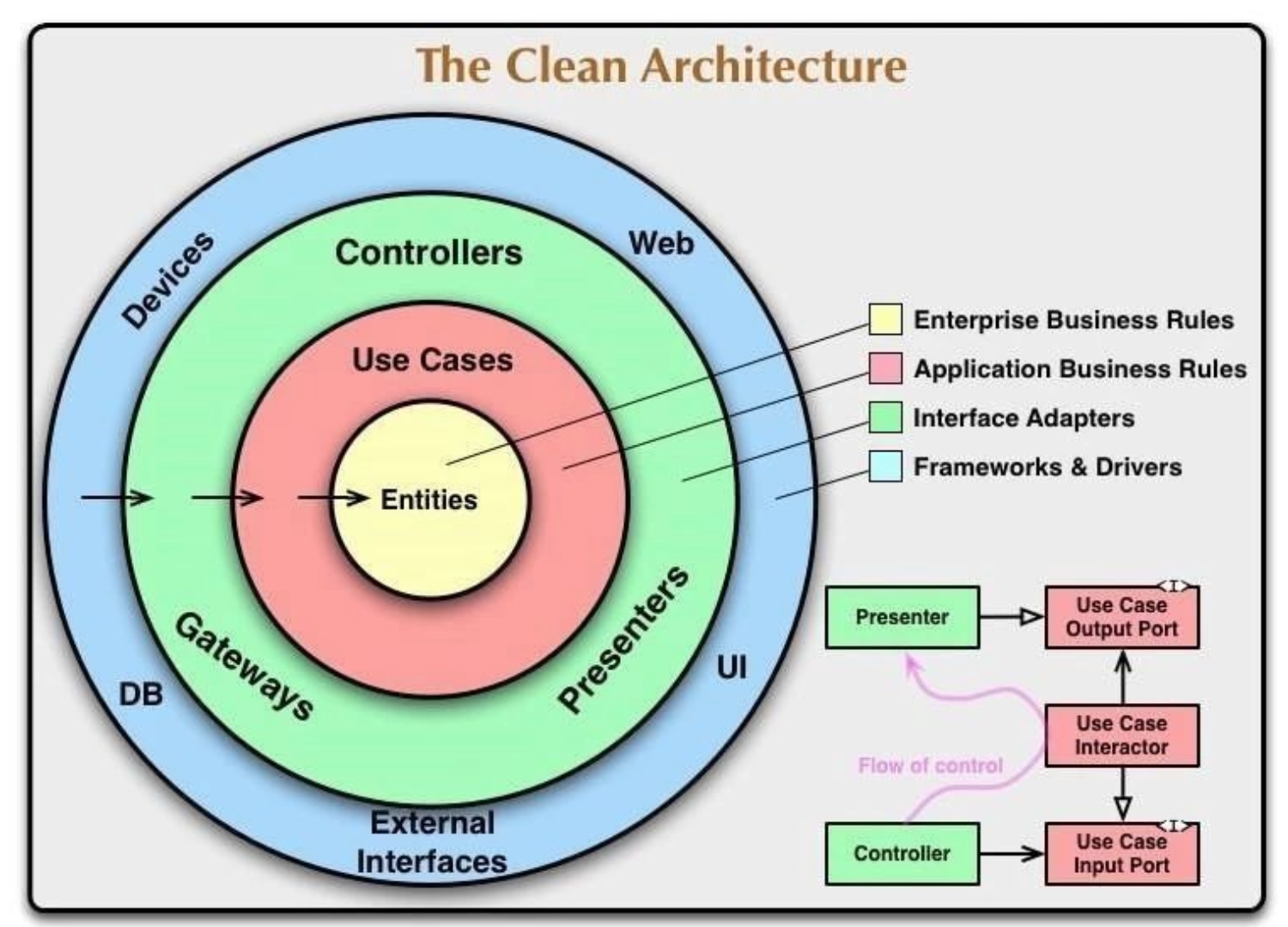
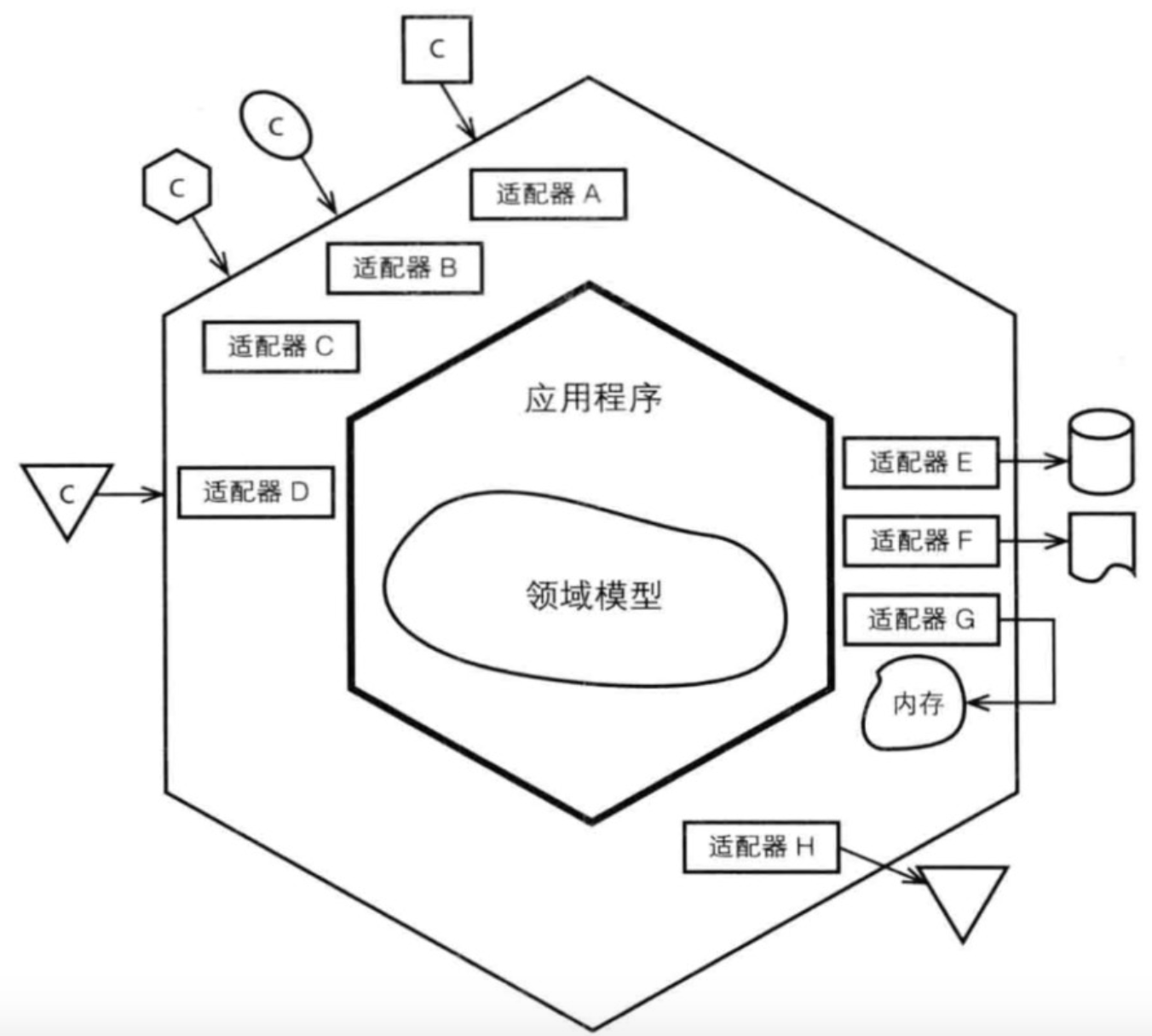
* 微服务的本质就是单一职责：高内聚低耦合
* 微服务架构模型：

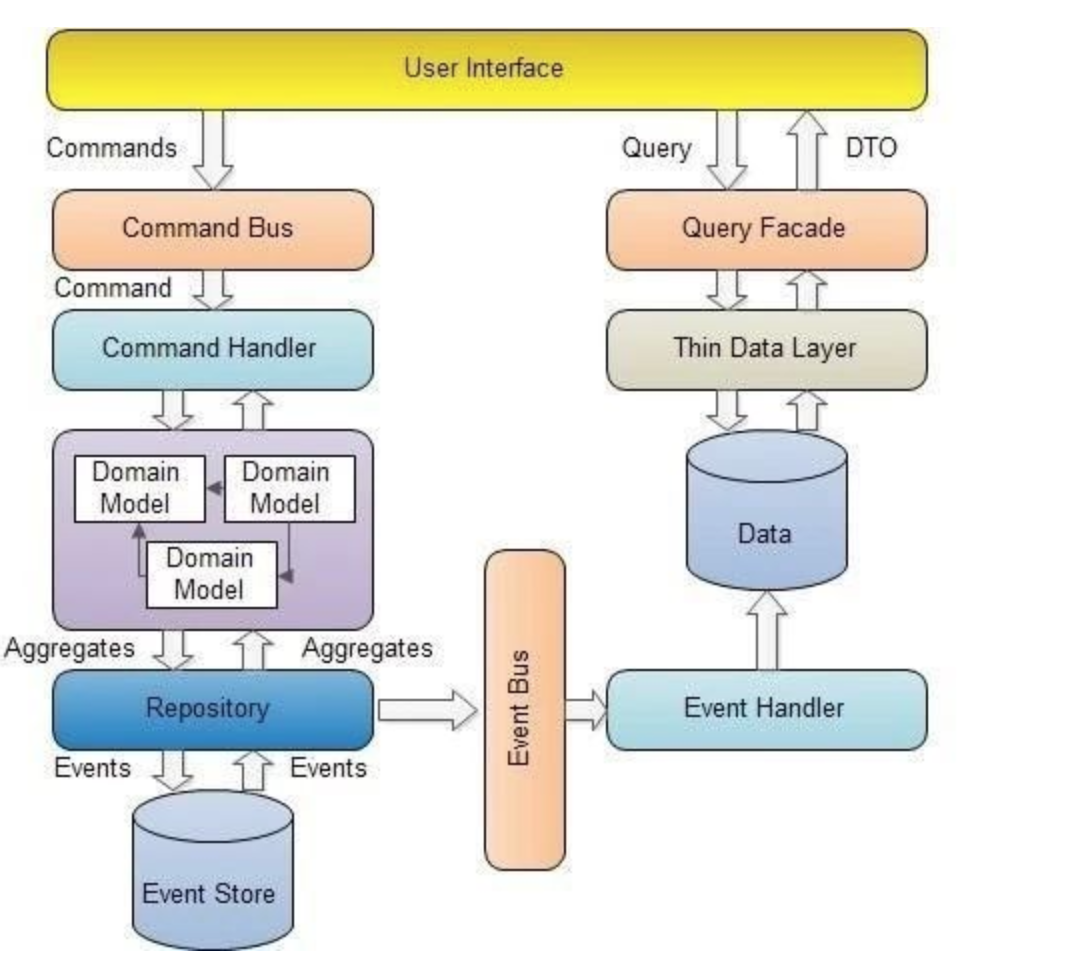
1. 整洁架构：特点：同心圆，层次依赖，从里向外：领域模型、领域服务、应用服务、最外围容易变化的内容；



1. 六边形架构：特点：业务逻辑与外部资源完全隔离；实现良好的前后端分离；

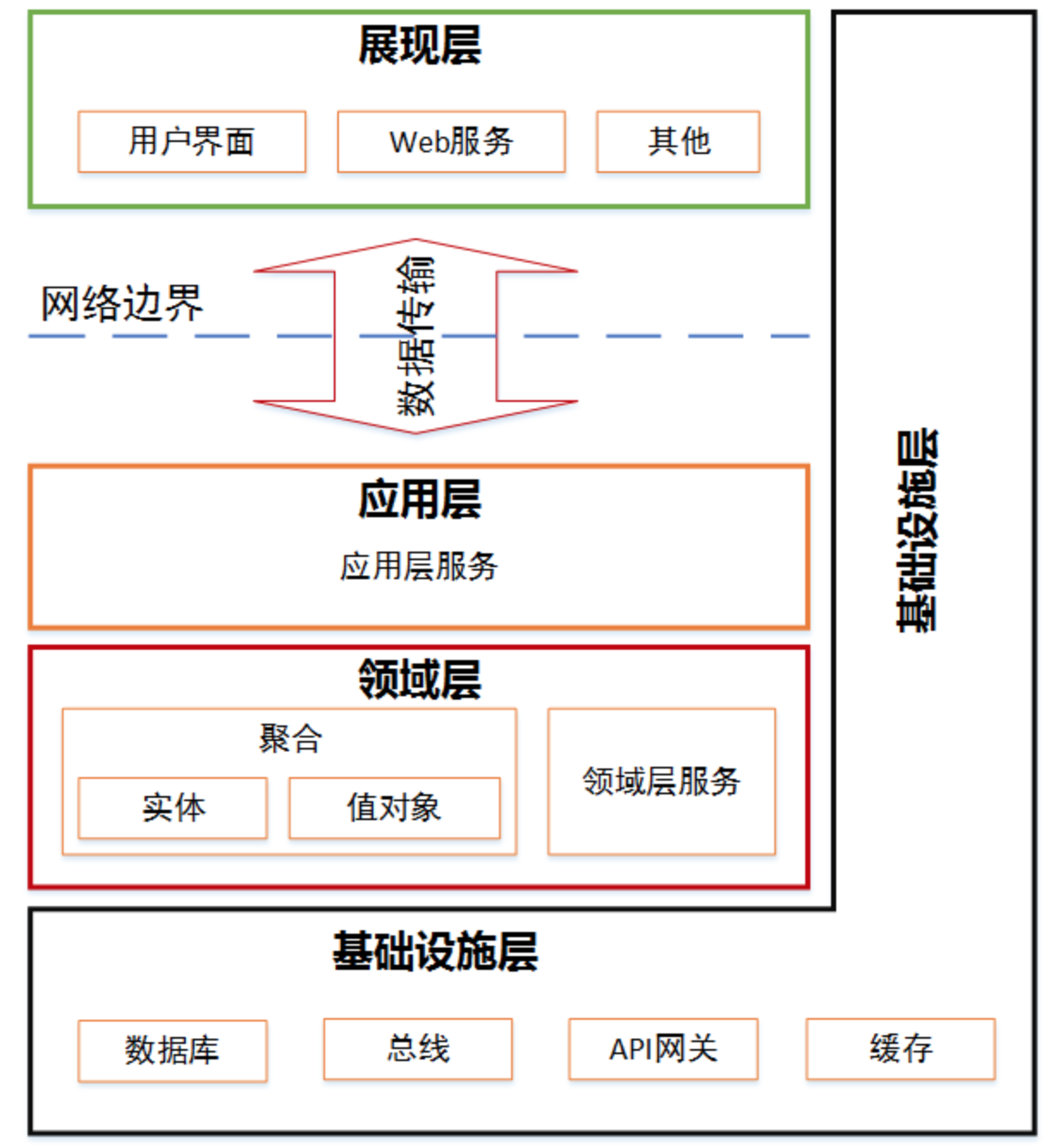


1. CQRS（命令与查询职责分离）：特点：由命令模型与查询模型组成，查询模型主要是为了提高查询效率，命令模型是领域模型，主要是为了写入数据。通过消息的发布订阅机制完成通知。



* 领域驱动设计分层架构：

分层架构的优势：开发人员只需要关注架构中的某一层，很容易用新的方法替换掉原有层次的方法，降低层与层之间的依赖，有利于标准化，利于各层逻辑的复用。



设计准则：划分可能的层次，设计层次的接口，保持层间的隔离，仅使用下层提供的接口；

DDD分层架构：各层定义与职能

展现层：展示信息完成用户命令；

应用层：负责展现层与领域层之间的协调，与其他系统引用层进行交互，简单，负责调度领域层，不包含业务逻辑规则；

领域层：软件业务的核心，领域模型；

基础设施层：提供通用的技术能力，为应用层传递消息，为领域层提供持久话机制。

* 架构模型对比与分析

核心业务逻辑由应用层与领域层完成，领域层完成了最核心的业务逻辑，对外提供细粒度的领域服务，应用层通过编排提供粗粒度的服务；领域层实现业务逻辑能保证后台稳定，当前台变化时，后台不需要调整；

* 从几种架构模型看如何进行中台及微服务设计（关键在于合理的分层与领域模型的设计）

1.聚焦领域模型，领域内业务逻辑实现，实现公共需求

2.合理的架构划分，不要把与领域无关的业务逻辑放在领域层，代码模型的逻辑分层与微服务拆分思路：

* 项目级单应用：分层架构模型，分为领域层与应用层，逻辑简单这2个层可以部署到一起；
* 企业级多中台应用：

3.服务的管理：基础设施层、应用层与领域层；

4.资源的适配与解耦：提供依赖反转也就是接口，屏蔽资源的变更对业务逻辑的影响；

5.前台应用：

* 中台、领域驱动设计及微服务

DDD提出的领域模型概念，统一了分析、设计与开发语言和过程，使得软件能够更灵活的快速响应需求变化

软件分析与设计方法的3个阶段：

1. 单机架构时代cs架构围绕数据库驱动设计与开发；
2. 集中式架构时代，面向对象的设计方法，经典的3层架构，使得服务臃肿维护拓展差；
3. 分布式架构，领域驱动设计方法；

领域驱动设计主要的优势：业务导向，业务逻辑内聚，应用边界清晰，建立领域模型优先，分析设计代码和数据有机结合，代码即设计，扩展性好；

# 领域驱动设计概述

设计思想：通过领域驱动设计方法定义领域模型，确定业务与应用边界保证业务模型与代码模型的一致性。

领域驱动设计与微服务架构天生匹配；

领域驱动设计分为2个阶段：

1. 以通用语言作为交流的工具，在交流的过程中发现领域的概念，然后设计成领域模型；
2. 由领域模型驱动软件设计，用代码实现领域模型；

领域驱动设计业务架构与系统架构形成绑定关系，分为战略设计（领域模型定义）与战术设计（领域设计落地）

领域驱动设计的收获：1.提供设计思路（方法论）；2.可以处理比较复杂的业务逻辑，形成稳定的领域模型内核；3.善于处理系统架构的演进；4.有助于提高团队成员的架构设计能力；5.与微服务架构天生匹配。

* 为什么领域驱动设计是微服务架构的最佳设计方法

2者都是从业务视角去离复杂度的手段，2者都强调从业务出发，领域驱动设计是根据业务发展合力划分领域边界，微服务架构师持续调整现有架构，优化现有代码，以保持架构和代码的生命力；

* 中台、DDD与微服务

中台来自于阿里巴巴，方便一线业务敏捷开发，提炼共同的需求，避免冗余；实现公共需求的共享；

领域的定义：领域就是一个问题域，同一个领域，问题域相同；根据重要性与功能属性分为三类子域：核心子域（决定产品和企业独特竞争力）、支撑子域（必须的）、通用子域（就是通用的）；

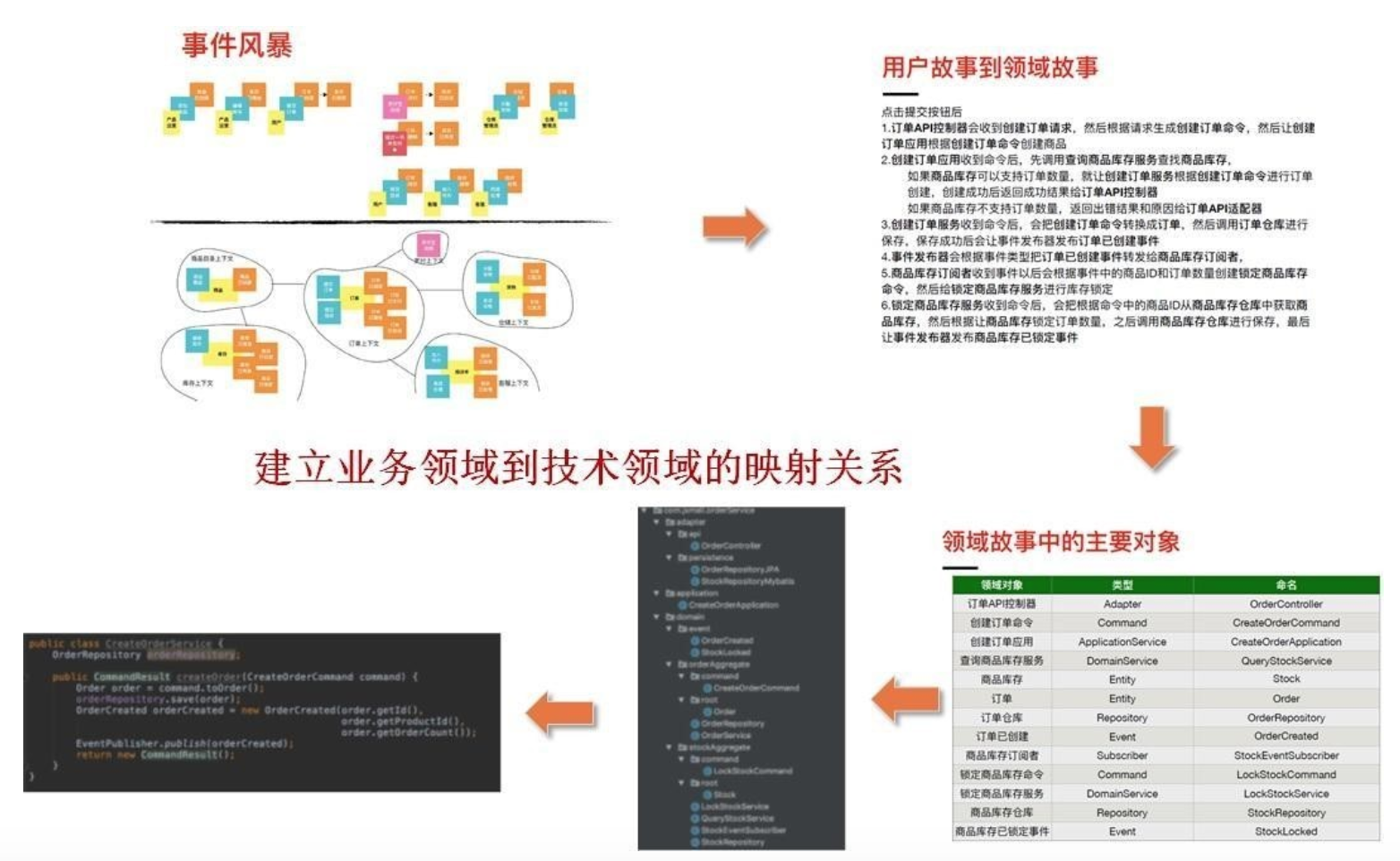
微服务是技术实现域部署的范畴，实现领域与中台的业务逻辑。

* 基于DDD的微服务设计方法

战略设计阶段主要完成领域建模与服务地图，战术阶段主要完成微服务的建设和实施。通过DDD可以保证业务模型、系统模型、架构模型与代码模型的一致。

DDD设计过程：

1. 产品愿景：顶层价值设计，产品要设计成啥样；
2. 场景分析：
3. 领域建模：通过对业务或问题域进行分析，建立领域模型；
4. 服务地图：



* DDD分层架构中的服务

1.应用层服务，表述应用与用户行为，隐藏领域层；2.领域层服务，所有的业务逻辑；3.基础设施层服务，封装的基础资源服务；

* 微服务的边界设计

限界上下文与限界上下文之间或者聚合与聚合之间的边界是逻辑边界，微服务与微服务的边界是物理边界；逻辑边界强调业务逻辑与代码分层的隔离，物理边界强调部署与运行的隔离。

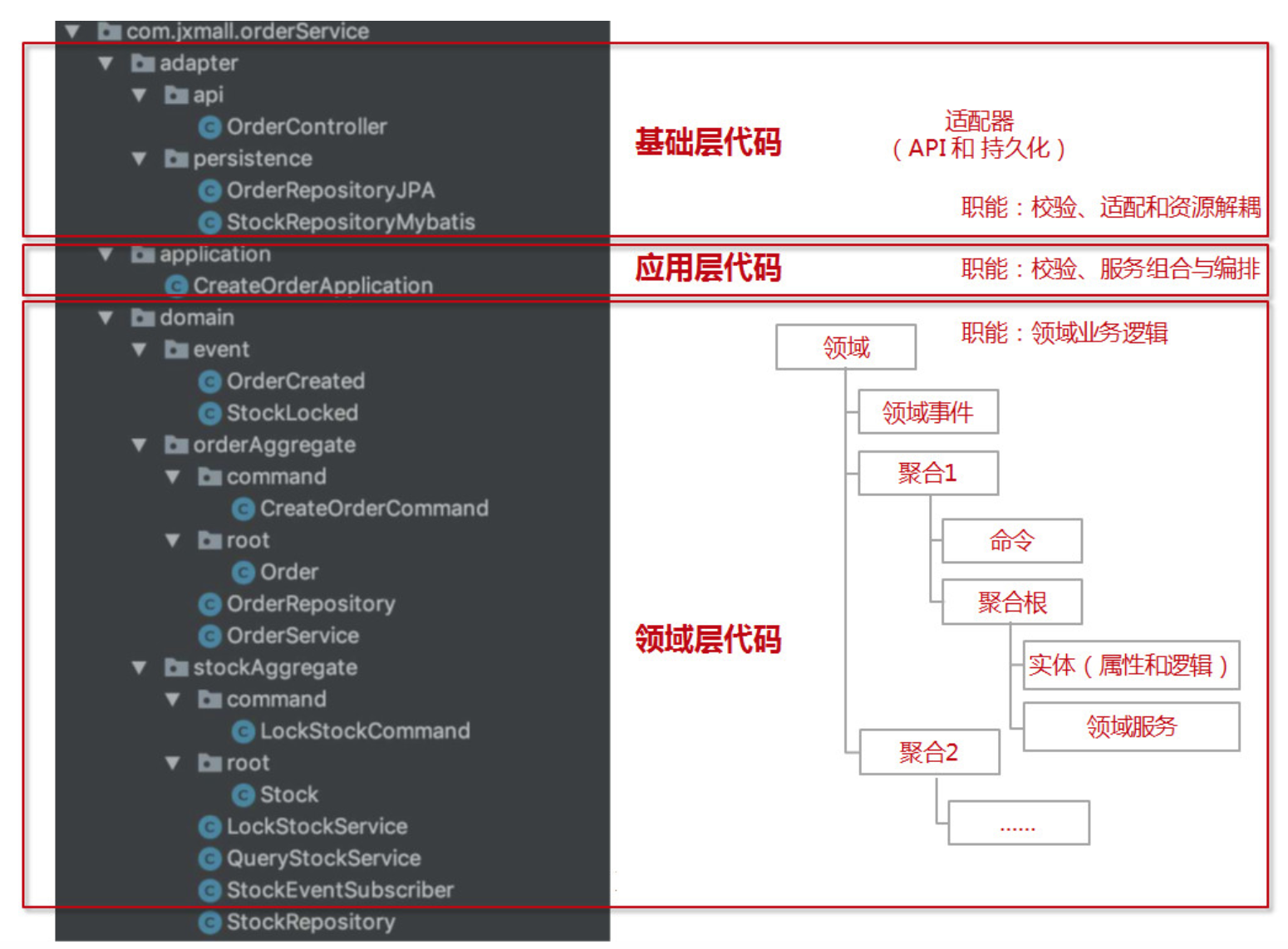
* 要做微服务而不是小单体

微服务设计是否合理的一个简单标准是：微服务在随着业务发展而不断拆分或者重组过程中不会过度增加软件维护成本，这个过程是轻松且简单的。

* 微服务代码逻辑分层和结构

微服务代码合理分层与逻辑隔离

1. 基础层代码：适配代码，主动适配与被动适配；
2. 应用层代码：
3. 领域层代码：



* 微服务的设计与拆分

微服务拆分方法：绞杀者模式（开发新的逐步替换旧的），修缮者模式（剥离新业务与功能）；

微服务拆分原则：高内聚，低耦合。

# 第一部分 运用领域模型

模型是一种简化，是对现实世界的解释；把与解决问题密切相关的方面抽象出来而忽略无关的细节。

领域就是软件所需要解决的问题。业务规则是很多的知识，理解起来麻烦，根据业务规则知识建立简单的模型，概括的反应业务知识。

模型是总体设计是实现的基石；模型容易理解，方便团队交流；模型使用容易理解的方式把业务规则知识组织起来，

## 消化知识

根据需求画图建立简单的模型；根据业务知识的增加丰富模型；简单实现模型；反馈给业务人员；忽略无关的小细节。

模型与实现的绑定，建立了一种基于模型的语言，模型含有很多的业务知识，模型不断被丰富完善。

模型需要不断尝试，找到一个最能反应业务知识规则的；