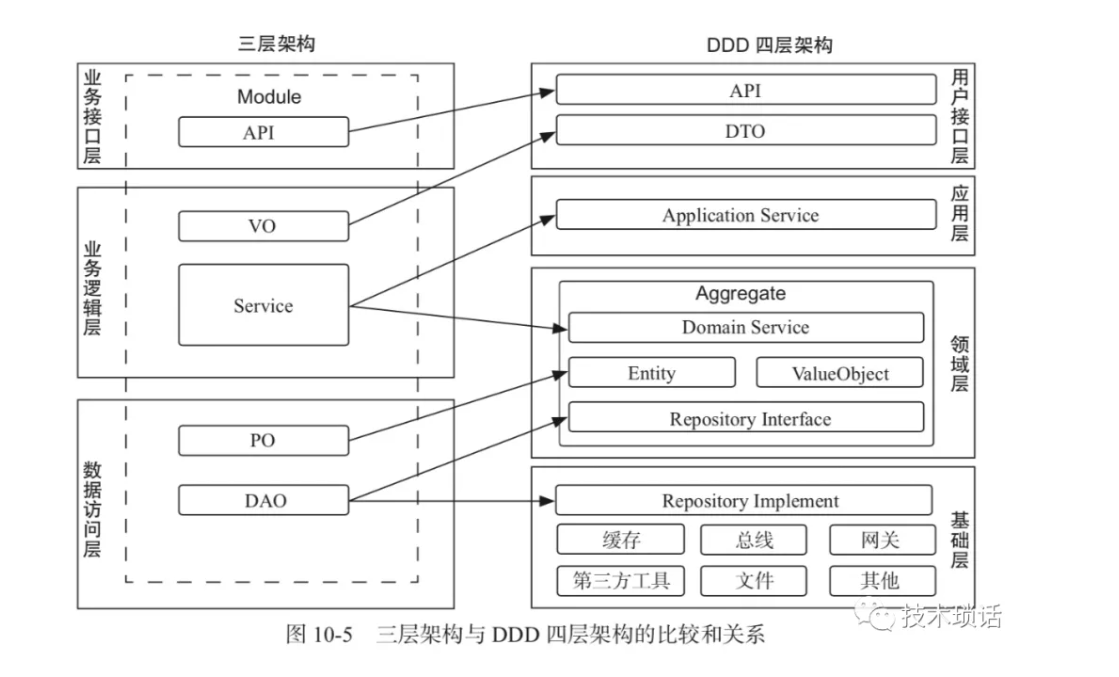
# 微服务ddd详解

## ddd模型的结构

1. 

感觉很复杂，东西很多，下面来一层一层地解释对应的作用,采用实习的时候的业务cms-center来解释对应的部分。

­这里一般是采用四层的结构，该结构包括

1. User Interface：这一般是前端的活
2. Application:定义软件要完成的任务，应用层要尽可能地简单，让其下的domain层更好地协调任务、分配工作、互相协助

比如一个updateDeviceInfo()的方法，你就应该在app层只实现到接受对应的传入参数（如果是rpc，那么对应的参数设置应该写在go kit endpoint/typedef里）+调用下层方法的程度

1. Domain层，包括：
2. Entity:

领域实体是domain的核心成员。domain entity具有如下三个特征：

1.**唯一业务标识**

**2.持有自己的业务属性和业务行为**

**3.属性可变，有着自己的生命周期**

例如在这个项目中，对应的连接的设备就是一个entity

1. ValueObject:

领域值对象。value object是相对于domain entity来讲的，对照起来value object有如下特征：

* **可以有唯一业务标识***【区别于domain entity】*
* **持有自己的业务属性和业务行为***【同domain entity】*
* **一旦定义，他是不可变的，它通常是短暂的**

例如在这个项目中，我们就把设备的持续时间作为我们的ValueObject

1. domain service

按道理来说，只要是没有办法将其归属于对应的实体的方法，都叫做domain service

例如这个项目中，你对设备的一些操作就归结于domain service

1. domain event

这个就是对应的domain使用的消息队列，以及一些事件类的东西

在这个项目中，就是对应的ebus文件夹，我们订阅了对应的消息队列（rocketMQ）

1. domain factory

这个是用于domain层的重构的，一般根据需求添加

1. repository：

仓库。我们将仓库的接口定义归类在domain层，因为他和domain entity联系紧密。仓库接口定义了和基础实施的持久化层交互契约，完成领域对应的增删改查操作。domain层的repository只是定义契约的接口，实际实现仍然由infrastructure完成。

　仓库的实际实现根据不同的存储介质而不同，可以是redis、oracle、mongodb等。具体仓库的实现会讲给infrastructure层完成，我们会在下一篇blog中详细阐述repository的实现。

对于repository的接口定义，建议规范接口名命名，比如：查询都叫着query等等，减小沟通成本。

在这个项目中，对应的仓库里会放一些和infra层交互的操作，一般只定义，在infra里实现

1. aggregate

那么什么是聚合呢？聚合就是把对应的objvalue和entity进行结合，再将这些在对应的上下文中进行界定