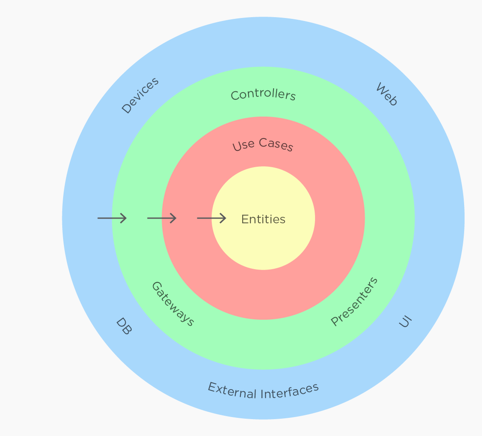
Atitit arch 架构方法总结fix v26

### **3、从分层架构到六边形架构**

#### ****3.1 分层架构****

#### ****3.2 整洁架构****

分层架构中给人的感觉是每一层都同样重要，但如果我们把关注的重点放在领域层，同时把依赖关系按照业务由重到轻形成一个以领域层为中心的环，即演变为一种整洁的架构风格。这里不是说其他层不重要，仅仅是为了凸显承载了业务核心的领域能力。



整洁架构最主要原则是依赖原则，它定义了各层的依赖关系，越往里，依赖越低，代码级别越高。外圆代码依赖只能指向内圆，内圆不知道外圆的任何事情。一般来说，外圆的声明（包括方法、类、变量）不能被内圆引用。同样的，外圆使用的数据格式也不能被内圆使用。

整洁架构各层主要职能如下：

****Entities：****实现领域内核心业务逻辑，它封装了企业级的业务规则。一个 Entity 可以是一个带方法的对象，也可以是一个数据结构和方法集合。一般我们建议创建充血模型。

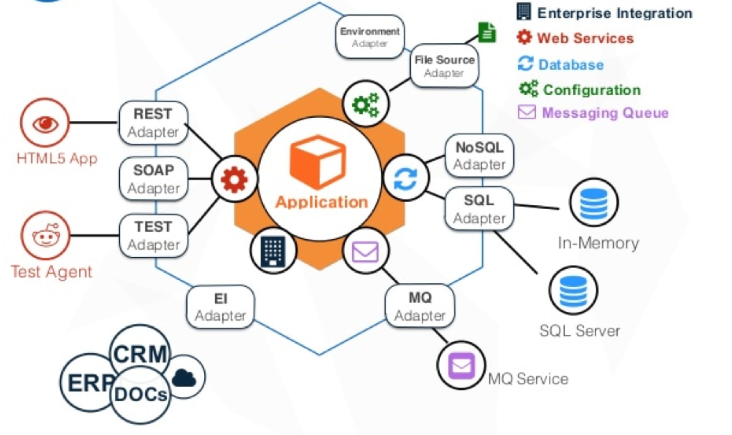
****Use Cases：****实现与用户操作相关的服务组合与编排，它包含了应用特有的业务规则，封装和实现了系统的所有用例。

****Interface Adapters：****它把适用于 Use Cases 和 entities 的数据转换为适用于外部服务的格式，或把外部的数据格式转换为适用于 Use Casess 和 entities 的格式。

****Frameworks and Drivers：****这是实现所有前端业务细节的地方，UI，Tools，Frameworks 等以及数据库等基础设施。

#### ****3.3 六边形架构****

我们把整洁架构的外部依赖按照其输入输出功能、资源类型进行整合。将存储、中间件、与其他系统的集成、http调用分别暴露一个端口。则会演变成下面的架构图。



“Allow an application to equally be driven by users, programs, automated test or batch scripts, and to be developed and tested in isolation from its eventual run-time devices and databases.”“系统能平等地被用户、其他程序、自动化测试或脚本驱动，也可以独立于其最终的运行时设备和数据库进行开发和测试”这是六边形的精髓。

该架构由端口和适配器组成，所谓端口是应用的入口和出口，在许多语言中，它以接口的形式存在。例如以取消订单为例，“发送订单取消通知”可以被认为是一个出口端口，订单取消的业务逻辑决定了何时调用该端口，订单信息决定了端口的输入，而端口为上游的订单相关业务屏蔽了其实现细节。

而适配器分为两种，主适配器（别名Driving Adapter）代表用户如何使用应用，从技术上来说，它们接收用户输入，调用端口并返回输出。Rest API是目前最常见的应用使用方式，以取消订单为例，该适配器实现Rest API的Endpoint，并调用入口端口OrderService，当然service内部可能发送OrderCancelled事件。同一个端口可能被多种适配器调用，本场景的取消订单也可能会被实现消息协议的Driving Adapter调用以便异步取消订单。

次适配器（别名Driven Adapter）实现应用的出口端口，向外部工具执行操作，例如向MySQL执行SQL，存储订单；使用Elasticsearch的API搜索产品；使用邮件/短信发送订单取消通知。有别于传统的分层形象，形成一个六边形，因此也会称作六边形架构。