**[四种优秀架构](http://designzum.com/2014/01/28/4-great-programming-architectures-you-should-know/" \o "原始链接" \t "_blank)**

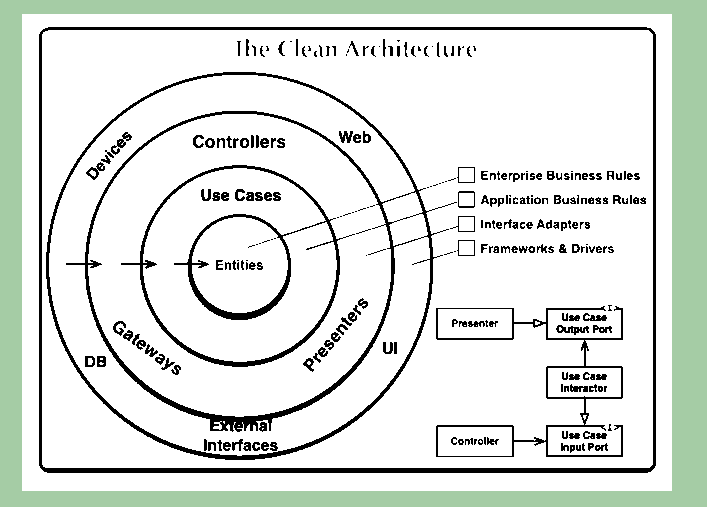
除非你是非常熟悉基础编程的整个世界，否则你很难了解编程[**架构**](https://www.jdon.com/tags/249)到底是什么。所以我们假设你并不太了解编程技术，那么我会说，编程是一种定义逻辑的途径或方法，这种逻辑以代码方式设计，让​​指定的编译器能够理解它，让编译器能够知道如何指挥计算机执行相应的功能。对于一个新手来说，这可能是编程的最简单的定义(banq注：对于缺乏逻辑的新手，这可能是最难懂的定义，因此，对于可以不编程的架构师来说最基本的能力是逻辑能力)。

基本上有三种类型的编程：低层次的编程，中间级编程和高级编程。所有这三种编程类型中通用的事情是：都可以执行相同的功能。只是对于不同编程类型具体执行的方式是不同的。

当一个程序很容易地运行,有可能在同样设备上也有其它程序运行。但是，如果所有正在运行的程序需要内部通讯怎么办？这是由该程序的架构来解决这个问题。一个程序架构是一种设计的结构,在设计时就要考虑相互通讯方案，两个程序通讯也许中间需要经历许多相互等待的阶段，因此，当你使用你喜欢的任何语言编写这种程序时，必须要记住，程序必须遵循的架构，如线性编程(顺序编程)，也就是说所有的步骤要遵循一个接一个的顺序，这当然会付出执行时间的代价。另一方面，当所有的步骤都是并行执行，并在最后一步全部完成时，所需的累积(cumulative )时间是相当少。因此，最好的架构是至少有一个累积等待阶段。

但是，这并不意味着线性编程就没有用而总是使用并行架构。重要的是要知道最好的几种可能的架构，这样您可以轻松地创建一个适合所有要求的优化方案。

下面是四种伟大的程序架构：

1. [Clean架构](http://blog.8thlight.com/uncle-bob/2012/08/13/the-clean-architecture.html)  


外圈的层次可以依赖内层，反之不可以，内圈核心的实体代表业务，不可以依赖其所处的技术环境。

2.[DCI架构](http://www.artima.com/articles/dci_vision.html)，[本站中文DCI架构专题](http://www.jdon.com/dci.html)，DCI代表Data, Context, Interaction。

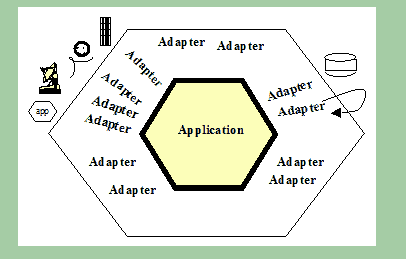
3.[DDD/CQRS](http://cqrsguide.com/ddd) 领域驱动设计，[本站](http://www.jdon.com/ddd.html)[**DDD**](https://www.jdon.com/tags/272)专题

领域驱动设计对于成功交付和维护[**CQRS**](https://www.jdon.com/tags/9958)的系统非常重要。 DDD作为一项战略方针，允许将复杂的问题域划分为单独的块（称为[**有界上下文**](https://www.jdon.com/tags/15977)），虽然有很多方式如：不同的心智Mental模式，组织政治，域语言学等也是这样做，但是DDD建立了一个有界的心智mental模式，这样商务人士也可以理解，程序员也可以很容易地在代码中实现。

CQRS，作为一种战术办法，是实现DDD建模领域的最佳途径之一。事实上，它就是因为这个目标而诞生在这个世界上。

相关资源：[DDD – CQRS Leaven V20](http://prezi.com/akrfq7jyau8w/ddd-cqrs-leaven-v20/)

4.[六边形架构](http://alistair.cockburn.us/Hexagonal+architecture)



允许应用程序都是由用户，程序，自动化测试或批处理脚本驱动的，在事件驱动和数据库环境下被开发和隔离测试。一个事件从外面世界到达一个端口，特定技术的适配器将其转换成可用的程序调用或消息，并将其传递给应用程序。该应用程序是可以无需了解输入设备的性质(调用者是哪个)。当应用程序有结果需要发出时，它会通过一个端口适配器发送它，这个适配器会创建接收技术（人类或自动）所需的相应信号。该应用程序与在它各方面的适配器形成语义良性互动，但是实际上不知道适配器的另一端的谁在处理任务。