链滴

DRY、KISS、YAGNI 三原则的理解

作者：valarchie
原文链接：https://ld246.com/article/1566828680282
来源网站：链滴

软件三原则的个人理解

在软件的设计当中前人已经总结了许多的设计原则和设计模式。例如SOLID，GRASP设计原则，这些则都是基于面向对象设计总结而来的。而GOF23是基于许多常见的场景总结出了一套设计模式，在我遇到类似的场景，都可以套用设计模式。而今天所讲到的软件三原则是适用于在软件设计的各个层面。它不仅适用于面向对象的设计，也适用于面向过程的程序设计；不仅适用于类的设计，也适用于模、子系统的设计。就连在项目架构运维部署中也适用于这一套简单的法则。

DRY - Don't Repeat Yourself

第一条准则是千万不要重复你自身。尽量在项目中减少重复的代码行，重复的方法，重复的模块。其许多设计原则和模式最本质的思想都是在消除重复。我们经常提起的重用性和可维护性其实是基于减重复这一简单的思想。为什么现在微服务盛行呢？正是因为将系统中的服务进行抽取的话，便减少了复。重复冗余在维护代码的时候将是非常困难的。DRY意味着系统内的每一个部件都应该是唯一的并是不模糊的。我们可以通过应用单一职责接口隔离等原则尽量拆分系统，模块，类，方法。使其每一部件都是职责明确的并且可重用的。

KISS - Keep It Simple & Stupid

第二条准则是保持简单易懂。从小到几行代码的写法大到整个系统的架构我们都应该保持简单易懂。手高就高在可以将复杂的东西“简单”的实现出来。刚入行的时候，我总喜欢用三目运算符将复杂的辑用一句冗长的代码行写出来。后面才发现这是非常愚蠢的。到了重构或者需求变更的时候，连我自写的代码我都看着非常费劲难以下手。所以我们应该致力于代码的可理解性。降低复杂度也意味着维变得简单。Martin Flower在《重构》中有一句经典的话："任何一个傻瓜都能写出计算机可以理解的序，只有写出人类容易理解的程序才是优秀的程序员。其实不光是程序，这个原则也可以延伸到产品设计，业务的设计，项目结构的设计上。

YAGNI - You Ain’t Gonna Need It

第三条准则是你将不会需要它。千万不要进行过度设计。我们经常会在开发当中尽可能的迎合未来可的需求。而为了迎合某些产生概率极低的需求而设计的成本是非常高的，这种过度设计的收益非常低可能你深思熟虑的设计花了不少时间成本，却在未来的两三年内这个设计却完全没有派上用场。一些计是否必要，更多的应该基于当前的情况。而不是为了应对未来的各种变化，画蛇添足的设计。如果宝一开始就往日均交易上亿的情况进行设计的话，那么可能就不会有今天的淘宝了。因为创业公司的间是非常宝贵的，比其他公司早一步退出新的服务就能抢占先机。并不是说淘宝不需要考虑以后交易暴增的情况，而是不应该以当前日均交易才几万的情况下去设计编码日均交易上亿的项目。过度设计往会延缓开发迭代的速度。

原文链接：DRY、KISS、YAGNI 三原则的理解