20 | 简单和直观，是永恒的解决方案

2019-02-18 范学雷

上一次，我们聊了影响代码效率的两个最重要的因素，需求膨胀和过度设计。简单地说，就是找到要做的事情，做的事情要少。接下来，我们来聊聊怎么做这些事情。其中，我认为最重要的原则就是选择最简单、最直观的做法。反过来说，就是不要把事情做复杂了。

要想简单直观，我们要解决两个问题。 第一个问题是，为什么要简单直观？第二个问题是，该怎么做到简单直观？

为什么需要简单直观？

简单直观，看似是一条每个人都能清楚明白的原则。事实上，这是一个非常容易被忽略的原则。如果我们没有对简单直观这个原则有一个基本的认识，就不太可能遵循这样的原则。

我们都喜欢原创和挑战，来展示我们的聪明才智。而简单直观的事情，显而易见的解决办法，似乎不足以展示我们的智慧和解决问题的能力。

遗憾的是，在软件世界里，一旦我们脱离了简单直接的原则，就会陷入行动迟缓、问题倍出的艰难境地。简洁是智慧的灵魂，我们要充分理解这一点。

简单直观是快速行动的唯一办法

我们真正要的不是简单直观的代码，而是轻松快速的行动。编写简单直观的代码只是我们为了快速行动而不得不采取的手段。有一个说法，如果面条代码能够让我们行动更快，我们就会写出面

条代码，不管是刀削面还是担担面。

我见过的优秀的程序员，无一例外，都对简洁代码有着偏执般的执着。甚至小到缩进空格该使用几个空格这样细枝末节的问题，都会严格地遵守编码的规范。乍一看，纠缠于缩进空格不是浪费时间吗？可是真相是，把小问题解决好，事实上节省了大量的时间。

这些对代码整洁充满热情的工程师，会对整个团队产生积极的、至关重要的影响。这种影响，不仅仅关乎到工程进展的速度，还关系到工程的质量。真正能够使得产品获得成功，甚至扭转科技公司命运的，不是关键时刻能够救火的队员，而是从一开始就消除了火灾隐患的队员。

简单直观减轻沟通成本

简单直观的解决方案，有一个很大的优点，就是容易理解，易于传达。事情越简单，理解的门槛越低，理解的人越多，传达越准确。一个需要多人参与的事情，如果大家都能够清晰地理解这件事情，这就成功了一半。

我们不要忘了，客户也是一个参与者。简单直观的解决方案，降低了用户的参与门槛，减轻了学习压力，能够清晰地传递产品的核心价值，最有可能吸引广泛的用户。

简单直观降低软件风险

软件最大的风险，来源于软件的复杂性。软件的可用性，可靠性，甚至软件的性能，归根到底，都是软件的复杂性带来的副产品。越复杂的软件，我们越难以理解，越难以实现，越难以测量，越难以实施，越难以维护，越难以推广。如果我们能够使用简单直接的解决方案，很多棘手的软件问题都会大幅地得到缓解。

如果代码风格混乱，逻辑模糊，难以理解，我们很难想象，这样的代码会运行可靠。

该怎么做到简单直观？

如果我们达成了共识，要保持软件的简单直观，那么，我们该怎么做到这一点呢？最重要的就是做小事，做简单的事情。

使用小的代码块

做小事的一个最直观的体现，就是代码的块要小，每个代码块都要简单直接、逻辑清晰。整洁的代码读起来像好散文，赏心悦目，不费力气。

如果你玩过乐高积木，或者组装过宜家的家具，可能对“小部件组成大家具”的道理会深有体会。代码也是这样，一小块一小块的代码，组合起来，可以成就大目标。作为软件的设计师，我们要做的事情，就是识别、设计出这些小块。如果有现成的小块代码可以复用，我们就拿来用。如果没有现成的，我们就自己来实现这些代码块。

为了保持代码块的简单，给代码分块的一个重要原则就是，**一个代码块只做一件事情**。前面，我们曾经使用过下面的例子。这个例子中，检查用户名是否符合用户名命名的规范，以及检查用户名是否是注册用户，放在了一个方法里。

```
/**
 * Check if the {@code userName} is a registered name.
 *
 * @return true if the {@code userName}is a registered name.
 * @throws IllegalArgumentException if the {@code userName} is invalid
 */
boolean isRegisteredUser(String userName) {
    // snipped
}
```

如果单纯地从代码分块来看，还有优化的空间。我们可以把上述的两件事情，分别放到一个方法里。这样，我们就有了两个可以独立使用的小部件。每个小部件都目标更清晰，逻辑更直接，实现更简单。

```
/**
 * Check if the {@code userName} is a valid user name.
 *
 * @return true if the {@code userName} is a valid user name.
 */
boolean isValidUserName(String userName) {
    // snipped
}
```

```
/**
 * Check if the {@code userName} is a registered name.
 *
 * @return true if the {@code userName} is a registered name.
 */
boolean isRegisteredUser(String userName) {
    // snipped
}
```

遵守约定的惯例

把代码块做小，背后隐含一个重要的假设：这些小代码块要容易组装。不能进一步组装的代码，如同废柴，没有一点儿价值。

而能够组装的代码，接口规范一定要清晰。越简单、越规范的代码块，越容易复用。这就是我们前面反复强调的编码规范。

花时间做设计

对乐高或者宜家来说，我们只是顾客，他们已经有现成的小部件供我们组合。对于软件工程师而言，我们是软件的设计者，是需要找出识别、设计和实现这些小部件的人。

识别出这些小部件，是一个很花时间的事情。

有的程序员，喜欢拿到一个问题，就开始写代码，通过代码的不断迭代、不断修复来整理思路，完成设计和实现。这种方法的问题是，他们通常非常珍惜自己的劳动成果，一旦有了成型的代码，就会像爱护孩子一般爱护它，不太愿意接受新的建议，更不愿意接受大幅度的修改。结果往往是，补丁摞补丁，代码难看又难懂。

有的程序员，喜欢花时间拆解问题，只有问题拆解清楚了，才开始写代码。这种方法的问题是，没有代码的帮助，我们很难把问题真正地拆解清楚。这样的方法，有时候会导致预料之外的、严重的架构缺陷。

大部分的优秀的程序员，是这两个风格某种程度的折中，早拆解、早验证，边拆解、边验证，就像剥洋葱一样。

拆解和验证，看起来很花时间。是的，这两件事情的确很耗费时间。但是，如果我们从整个软件的开发时间来看，这种方式也是最节省时间的。如果拆解和验证做得好，代码的逻辑就会很清晰，层次会很清楚，缺陷也少。

一个优秀的程序员，可能80%的时间是在设计、拆解和验证，只有20%的时间是在写代码。但是，拿出20%的时间写的代码，可能要比拿出150%时间写的代码，还要多，还要好。这个世界真的不是线性的。

有一句流传的话，说的是“跑得慢，到得早”。这句话不仅适用于健身，还适用于写程序。

借助有效的工具

我自己最常使用的工具，就是圆珠笔和空白纸。大部分问题，一页纸以内，都可以解决掉。当然，这中间的过程，可能需要一打甚至一包纸。

一旦问题有点大，圆珠笔和空白纸就不够用了。这时候，我们需要称手的工具，帮助我们记忆和

思考。

现在我最喜欢的工具有思维导图、时序图和问题清单。在拆解问题时，思维导图可以帮助我厘清思路，防止遗漏。时序图可以帮助我理解关键的用例，勾画清楚各个部件之间的联系。而问题清单，可以记录下要解决和已经解决的问题，帮助我记录状态、追踪进度。

你最顺手的工具是什么？欢迎你分享在留言区，我们一起来学习。

小结

今天，我们主要聊的话题，就是做小事。我们工作生活中，一旦出现两种以上的竞争策略，要记住这个经过实践检验的理念：选择最简单，最直观的解决方案。

当然，我们遇到的不会总是简单的问题。 如果把复杂的问题、大的问题，拆解成简单的问题、小的问题，我们就能够化繁为简，保持代码的整洁和思路的清晰。

一起来动手

通常一个用户登录的设计，需要输入用户名和密码。用户名和密码一起传输到服务器进行校验，授权用户登录。但现在有了更先进的设计。用户先输入用户名，用户名通过服务器检验，才能进一步输入密码，然后授权用户登录。

你愿不愿意分析一下，这种简单的流程变化，带来的收益？ 客户端和服务器端的接口代码，大致应该是什么样子的？你使用了什么工具来分析这些问题？

欢迎你在留言区讨论上面的问题，我们一起来看看这种简单的变化可以带来什么样的好处。

精选留言

北风一叶

1

代码块越小，越容易复用，这个和“重构改善即有代码设计”里的观点一致

2019-04-02