# 加餐 | 你真的了解重构吗?

2019-03-15 郑晔

10x程序员工作法 进入课程 >


讲述：郑晔
时长 10:19 大小 9.45M

今天（3 月 15 日），Martin Fowler 《重构》第二版的中文版正式发布。前不久，人邮的杨海灵老师找到我，让我帮忙给这本书写推荐语，我毫不犹豫地就答应了，有机会为经典之作写推荐语，实属个人荣幸。

不过，我随即想到，在专栏里，我只是在谈 TDD 的时候提到了重构，并没有把它作为一个专门的话题来讲，于是，我决定给我的专栏读者加餐，专门谈谈重构，毕竟重构是几乎每个程序员都会用到的词汇。但你真的了解重构吗?

## 每个程序员都要做的事

作为程序员，我们都希望自己的代码是完美的。但没有代码是完美的，因为只要你的代码还有生命力，一定会有新的需求进来，而新的需求常常是你在编写这段代码之初始料未及的。

很多人直觉的选择是，顺着既有的代码结构继续写下去，这里添一个 if，那里加一个标记位，长此以往，代码便随时间腐坏了。

如果用一个物理学术语描述这种现象，那就是“熵增”，这也就是大名鼎鼎的热力学第二定律。如果没有外部干预，系统会朝着越来越混乱的方向发展。对抗熵增的一个办法就是引入负熵，让系统变得更加有序。而在代码中引入负熵的过程就是“重构”。

调整代码这件事是程序员都会有的习惯，但把这件事做到比较系统，上升为“重构”这个值得推广的实践是从一个小圈子开始的，这个小圈子的核心就是我们在专栏里前面提到过的两位大师级程序员：Ward Cunningham 和 Kent Beck。

而真正让这个概念走出小圈子，来到大众面前的，则是 Martin Fowler 在 1999 年写下那本软件行业的名著《重构：改善既有代码的设计》（Refactoring: Improving the Design of Existing Code）。

Martin Fowler 的本事就在于他极强的阐述能力，很多名词经过他的定义就会成为行业的流行语（Buzzword），重构就是其中之一。

重构这个说法可比“调整代码”听上去高级多了。时至今日，很多人都会把重构这个词挂在嘴边：“这个系统太乱了，需要重构一下。”

**但遗憾的是，很多程序员对重构的理解是错的。**

## 重构是一种微操作

你理解的重构是什么呢？就以前面那句话为例：这个系统太乱了，需要重构一下。如果我们接着问，你打算怎么重构呢？一些人就会告诉你，他们打算另立门户，重新实现这套系统。对不起，**你打算做的事叫重写（rewrite），而不是重构（refactoring）。**

《重构》是一本畅销书，但以我的了解，很少有人真正读完它，因为 Martin Fowler 是按照两本书（Duplex Book）来写的，这是他常用写书的风格，前半部分是内容讲解，后半部分是手册。

让这本书真正声名鹊起的就是前半部分，这部分写出了重构这件事的意义，而后半部分的重构手册很少有人会看完。很多人以为看了前半部分就懂了重构，所以，在他们看来，重构就是调整代码。调整代码的方法我有很多啊，重写也是其中之一。

如果真的花时间去看这本书的后半部分，你多半会觉得很无聊，因为每个重构手法都是非常细微的，比如，变量改名，提取方法等等。尤其是在今天，这些手法已经成了 IDE 中的菜单。估计这也是很多人就此把书放下，觉得重构不过如此的原因。

所以，行业里流传着各种关于重构的误解，多半是没有理解这些重构手法的含义。

**重构，本质上就是一个“微操作”的实践。**如果你不能理解“微操作”的含义，自然是无法理解重构的真正含义，也就不能理解为什么说“大开大合”的重写并不在重构的范畴之内。

我在《大师级程序员的工作秘笈》这篇文章中曾经给你介绍过“微操作”，每一步都很小，小到甚至在很多人眼里它都是微不足道的。

重构，也属于微操作的行列，与我们介绍的任务分解结合起来，你就能很好地理解那些重构手法的含义了：**你需要把做的代码调整分解成若干可以单独进行的“重构”小动作，然后，一步一步完成它。**

比如，服务类中有一个通用的方法，它并不适合在这个有业务含义的类里面，所以，我们打算把它挪到一个通用的类里面。你会怎么做呢？

大刀阔斧的做法一定是创建一个新的通用类，然后把这个方法复制过去，修复各种编译错误。而重构的手法就会把它做一个分解：

添加一个新的通用类，用以放置这个方法；

在业务类中，添加一个字段，其类型是新添加的通用类；

搬移实例方法，将这个方法移动到新的类里面。

得益于现在的 IDE 能力的增强，最后一步，按下快捷键，它就可以帮我们完成搬移和修改各处调用的工作。

**在这个分解出来的步骤里，每一步都可以很快完成，而且，每做完一步都是可以停下来的，这才是微操作真正的含义。**这是大刀阔斧做法做不到的，你修改编译错误的时候，你不知道自己需要修改多少地方，什么时候是一个头。

当然，这是一个很简单的例子，大刀阔斧的改过去也无伤大雅。但事实上，很多稍有规模的修改，如果不能以重构的方式进行，常常很快就不知道自己改到哪了，这也是很多所谓“重

写”项目面临的最大风险，一旦开始，不能停止。

你现在理解了，重构不仅仅是一堆重构手法，更重要的是，**你需要有的是“把调整代码的动作分解成一个个重构小动作”的能力。**

## 重构地图

下面我准备给你提供一张关于重构的知识地图，帮你了解它与周边诸多知识之间的关系，辅助你更好地理解重构。

学习重构，先要知道重构的定义。关于这点，Martin Fowler 给出了两个定义，一个名词和一个动词。

> 重构（名词）：对软件内部结构的一种调整，目的是在不改变软件可观察行为的前提下，提高其可理解性，降低其修改成本。

> 重构（动词）：使用一系列重构手法，在不改变软件可观察行为的前提下，调整其结构。

之所以要了解重构的定义，因为重构的知识地图就是围绕着这个定义展开的。

首先，我们要对软件的内部结构进行调整，第一个要回答的问题是，我们为什么要调整。Martin Fowler 对于这个问题的回答是：代码的坏味道。

代码的坏味道，在我看来，是这本书给行业最重要的启发。很多人常常是无法嗅到代码坏味道的，因此，他们会任由代码腐坏，那种随便添加 if 或标记的做法就是嗅不出坏味道的表现。

我经常给人推荐《重构》这本书，但我也常常会补上一句，如果你实在没有时间，就去看它的第三章《代码的坏味道》。

顺便说一下，对比两版的《重构》，你会发现它们在坏味道的定义上有所差异，在新版的《重构》中，可变数据（Mutable Data）、循环语句（Loops）都定义成了坏味道，如果你不曾关注近些年的编程发展趋势，这样的定义着实会让人为之震惊。但只要了解了函数式编程的趋势，就不难理解它们的由来了。

换句话说，**函数式编程已然成为时代的主流**。如果你还不了解，赶紧去了解。

我们接着回到重构的定义上，重构是要不改变软件的可观察行为。我们怎么知道是不是改变了可观察行为，最常见的方式就是测试。

关于测试，我在“任务分解”模块已经讲了很多，你现在已经可以更好地理解重构、TDD 这些概念是怎样相互配合一起的了吧！

再来，重构是要提高可理解性，那重构到什么程度算是一个头呢？当年重构讨论最火热的时候，有人给出了一个答案：重构成模式（Refactoring to Patterns）。当然，这也是一本书的名字，有兴趣的话，可以找来读一读。

我个人有个猜想，如果这个讨论可以延续到 2008 年，等到 Robert Martin 的《Clean Code》出版，也许有人会提“重构成 Clean Code”也未可知。所以，无论是设计模式，亦或是 Clean Code，都是推荐你去学习的。

至此，我把重构的周边知识整理了一番，让你在学习重构时，可以做到不仅仅是只见树木，也可看见森林。当然，重构的具体知识，还是去看 Martin Fowler 的书吧！

## 总结时刻

总结一下今天的内容。今天我介绍了一个大家耳熟能详的概念：重构。不过，这实在是一个让人误解太多的概念，大家经常认为调整代码就是在做重构。

重构，本质上就是一堆微操作。重构这个实践的核心，就是将调整代码的动作分解成一个一个的小动作，如果不能理解这一点，你就很难理解重构本身的价值。

不过，对于我们专栏的读者而言，因为大家已经学过了“任务分解”模块，理解起这个概念，难度应该降低了很多。

既然重构的核心也是分解，它就需要大量的锤炼。就像之前提到任务分解原则一样，我在重构上也下了很大的功夫做了专门的练习，才能让自己一小步一小步地去做。但一个有追求的软件工匠不就应该这样锤炼自己的基本功吗？

如果今天的内容你只记住一件事，那请记住：**锤炼你的重构技能。**

最后，我想请你分享一下，你对重构的理解。欢迎在留言区写下你的想法。

感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给你的朋友。





上一篇　划重点 | 一次关于“沟通反馈”主题内容的复盘

下一篇　29 | “懒惰”应该是所有程序员的骄傲

## 精选留言 (18)

写留言

**非鱼**
2019-03-15

4

函数式编程有什么推荐书籍吗?

展开

作者回复: 有几本可以了解一下，《计算机程序的解释与构造》（Structure and Interpretation of Computer Programs，SICP），还有《Haskell 趣学指南》（Learn You a Haskell for Great Good!），《函数式编程思维》（Functional Thinking）。

**Zapup**

2019-03-24

1

维护别人的“旧代码”，可以从重构开始。

展开

作者回复: 重构，从理解重构开始。

**毅**

2019-03-17

1

重构不仅是愿景（名词），也不仅是行为（动词），还应该成为程序员必备的习惯和工作方式。但要成为习惯，甚至是深入骨髓的那种，是需要有积极意识和大量联系的。
有程序员会说，先把功能实现了，后面我再去重构，但后来的情况往往是不重构，或是债务过多重构代价太大，原因也大多是之前课程中提到的诸如任务分解不到位，微操作缺失，缺乏合理有效的单元测试等等，所以程序员的自我修养也是要体系化的，所谓功到...

展开

作者回复: 观察和解释都很到位。

**西西弗与卡...**

2019-03-15

1

不少人把重构望文生义或者扩大到成重新构造、重新构建甚至推倒重来了吧

展开

作者回复: 是的，典型的标题党式理解问题。:)

**helloworld**

2019-05-02

知道什么是代码坏味道，分步骤消灭坏味道

展开

**刘晓林**
2019-04-18

时隔三天，总感觉自己没有理解这一章的内容，尤其是文中举的把一个通用方法挪到一个通用类的例子到底是在说明什么。今天走在路上的时候突然好像“顿悟”了，于是又回来把文章看了一遍，总结如下：
1.重构是专业的。重构不是简单的把一堆代码挪到另一个位置，然后再去调整它，使得它能够编译通过和顺利运行。重构应该是一个具体的可以描述和追踪的任务，是由一个...
展开

**旭东**
2019-04-03

重构是“改革”，重写是“改朝”
展开

作者回复: 这个比喻...有点道理

**enjoylear...**
2019-03-27

我所理解的重构是在原有代码的基础上，无论是让代码clean,还是重构成模式，在有代码测试覆盖率保证的前提下，让其变得humanread和易维护。

**旭东**
2019-03-26

重构到重写也是个量变到质变的过程。个人理解。但重写不一定是从重构做到的

作者回复: 重写和重构是两个套路，重写基本上需要重新设计，重构一方面可以在日常工作中应用，一方面可以用以发现代码中的结构。

**Matrix**
2019-03-20

最近就在重写一个系统，因为我觉得重构已经救不了它了。。

展开

作者回复: 重写不是问题，别说重构就好。

**Haile**
2019-03-17

计算机程序构造与解释是神书哇……

展开

**我行我素**
2019-03-16

在之前的理解中一直认为重构就是将看着别扭或者影响性能和可能出现问题的代码进行调优，

作者回复: 如果你看到了差异，我的目的就达到了。

**hua168**
2019-03-16

我用idea建立了一个spring写了一个简单的MVC，按您的方法搞了半天终于搞定了。
但有一个问题，代码如下
package com.hualinux;

import org.springframework.stereotype.Repository;...

展开

作者回复: 你没有在这个类里面添加新类的实例。

```
public class T1Dao {
    private NewCommon target;
    ...

    public void printName(String name) {
        System.out.println("Dao层的name");
```

```
    }
}
```

**hua168**
2019-03-15

老师，我有点疑惑：
文中 重构的手法就会把它做一个分解：
●添加一个新的通用类，用以放置这个方法；
●在业务类中，添加一个字段，其类型是新添加的通用类；
●搬移实例方法，将这个方法移动到新的类里面。...
展开

作者回复: 最后一步叫“搬移”，就是把服务类的方法删除，在通用类里增加。用 IDE 的“重构”动作完成，它会替你搞定所有问题。

**金文**
2019-03-15

重构真的挺重要的，毕业时候去了一家公司代码写的很乱，清一色的复制复制，但是要改还不被允许，那时候真的有苦难言

**UnivTime**
2019-03-15

题外话：一个容易产生歧义的词本身可能就不是一个好词？或者说没翻译好？
展开

作者回复: 标题党之所以有市场，就是太多人不看内容了。

**Charles**
2019-03-15

最近正好在看这本书 如果光看名字 还真是很容易理解错
展开

作者回复: 赶紧去看第二版。

**Time**

2019-03-15

上次去面试，我说我们公司用的项目是ssm的，技术比较老，然后面试官问我没有想过重构嘛？使用新的框架。请问他这个'重构'的意思 其实是 '重写' 了吧？

展开

作者回复: 确实，他的理解是错的。