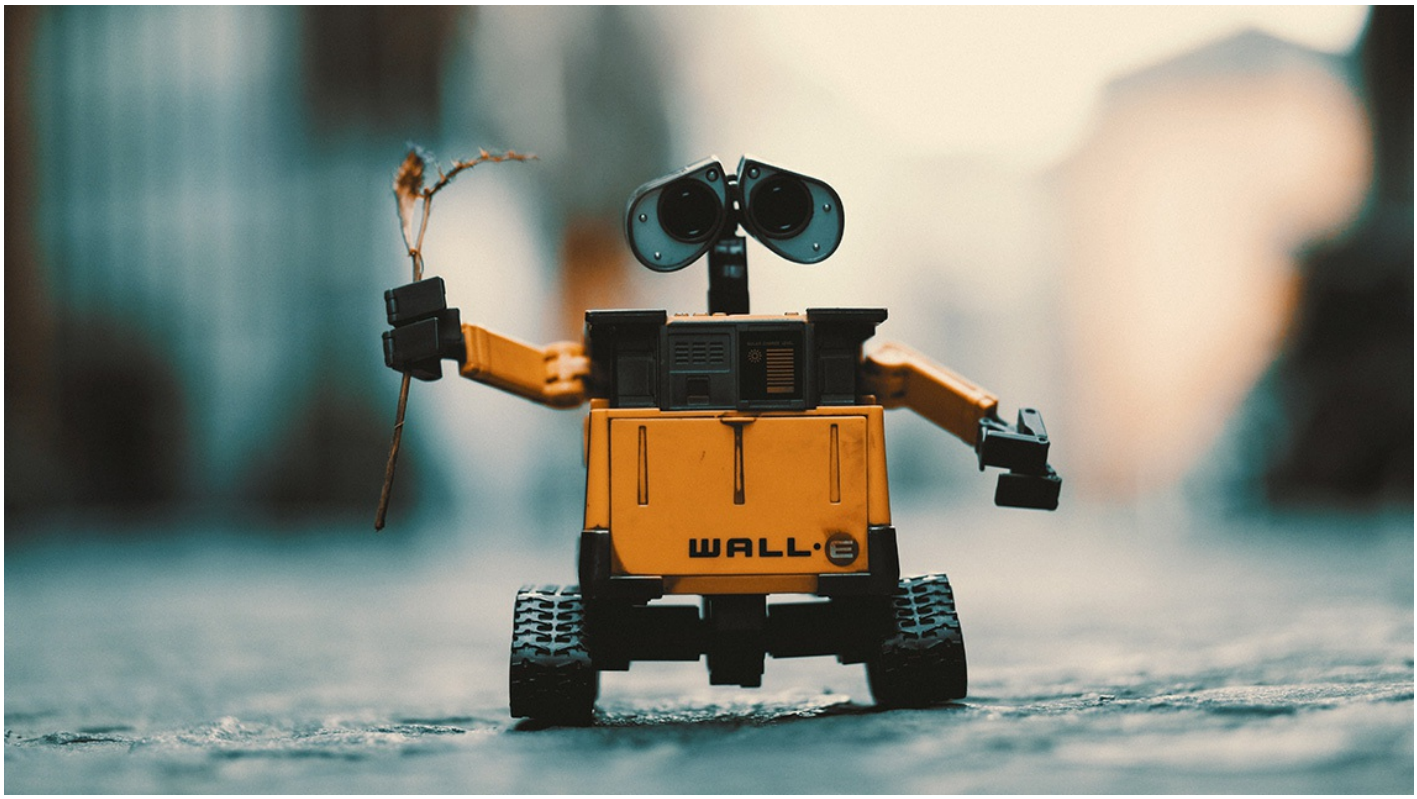
29讲“懒惰”应该是所有程序员的骄傲



经过前面几个模块的学习，我们的专栏终于进入到程序员看上去最熟悉的一个主题：自动化。

每每提及自动化，我就会想起 Perl 语言的发明人 Larry Wall 一个经典叙述：优秀程序员应该有三大美德：懒惰、急躁和傲慢（Laziness, Impatience and hubris）。

有人甚至为此专门打造了一个三大美德的网站，阐释这个初看起来匪夷所思的说法。

> 懒惰，是一种品质，它会使你花很大力气去规避过度的精力消耗，敦促你写出节省体力的程序，别人也能很好地利用，你还会为此写出完善的文档，以免别人来问问题。

> 急躁，是计算机偷懒时，你会感到的一种愤怒。它会促使你写出超越预期的程序，而不只是响应需求。

> 傲慢，极度自信，写出（或维护）别人挑不出毛病的程序。

不知道你是否感受到，程序员独有的幽默和透露出的那种骄傲：我做的东西就应该是最好的。

之所以要从 Larry Wall 的这段话开启“自动化”这个模块，因为只要一说到自动化，我就会情不自禁地联想到“偷懒”这个词。是的，我们程序员的工作，本质上就是打造各种自动化的工具，让人们从各种繁复的工作中解脱出来，让人有机会“偷懒”。

不过，我也知道，从机器那里偷来的“懒”很快就被更多的工作填满了。但 Larry Wall 的这段话却可以鼓励我们不断地打造出更好的工具。

作为程序员，你当然知道“自动化”这件事的价值，在日常工作中，也实实在在地践行着打造自动化工具的任务，但很多人对自动化的理解可能有些单薄。今天，我就从一个你可能会忽略的主题开始讨论：不要自动化。

不要自动化

我先给你讲一个让我印象深刻的“不自动化”的例子。

之前在 ThoughtWorks 工作时，我们有一项工作是，帮助其他公司启动一些新产品。有一次，我的两个同事被一个公司请去启动一个视频网站的项目。那时候还不像如今的市场，已经由几大视频网站瓜分完毕，当时不少公司看到了视频网站的苗头，觉得自己有机会。这个来请我们的公司也不例外，觉得自己也能分一杯羹。

两个星期之后，我的两个同事回来了。我们饶有兴趣地去问项目的进展，因为项目启动之后，通常会有后续的开发合作，但结果令我们很意外，这个项目停止了。

“出了什么状况吗？”我们问。

“是我们建议用户停掉这个项目的。”他们回答到。

我们“恨恨地”问他们为什么丢掉了一个这么重要的机会。这两个同事的回答也很直白，他们结合着客户的想法算了一笔账：这个项目需要大量的资金投入，投入规模之大，是超出客户想象的，按照现有的规划投入，这个项目肯定会亏本。要么重新规划，要么取消这个项目。客户认真研究了一番，最终决定取消项目。

这件事大约发生在10年前，今天我们都看到各大视频网站在烧钱上的投入，以那个公司的实力，想要参加这场比拼，确实还差太多。

这件事之所以给我留下深刻印象，因为它是我职业生涯中见到的第一个通过“主动取消项目”获取项目成功的案例。

或许你不能理解我这里所说的“项目成功”。在我看来，**做有价值的事是重要的，这里面的有价值，不仅仅是“做”了什么，通过“不做”节省时间和成本也是有价值的**。我的两个同事阻止了客户的浪费，所以，我将这个项目视为成功。

对于开发来说，也遵循同样的道理。程序员这个群体技术能力实在太强，做一个技术方案简直是太符合直觉的做法，我们就是忠实地把一个个需求做出来，把“全世界”都自动化了。

但事实上，这个世界太多的浪费就是做了不该做的东西。在我们的专栏里，我反复地说，我们要多问问题，目的就是为了不做那些不该做的事。

小心 NIH 综合症

你可以从需求的角度判断哪些工作是可以不做的，但我们也要防止程序员自己“加戏”，我再给你讲一个技术人员普遍存在的问题：NIH 综合症（Not Invented Here Syndrome）。

NIH 是什么意思？就是有人特别看不上别人做的东西，非要自己做出一套来，原因只是因为那个东西不是我做的，可能存在各种问题。

这种现象在开源之前尤为流行，很多公司都要做自己的中间件，做自己的数据库封装。虽然很多公司因此有了自己特色的框架，但是因为水平有限，做出来的东西通常极为难用，很多人一边骂，一边还要继续在上面开发。

开源运动兴起之后，我以为这种现象会好一些，但事实证明，我想多了。

比如，这种乱象在前端领域也出现了，各种各样的框架，让很多前端程序员哭诉，实在学不动了。再比如，我曾经面试过一个接触 Go 比较早的程序员，他就是恨不得把所有框架都自己写。

因为他学 Go 的时候，确实框架比较少，但问题是，如今的 Go 已经不是他学习时的那个 Go 了，现在各种框架已经很丰富了，不需要什么都自己做。当时我问他，如果有一天你离开了，公司怎么办呢？实际上，他从来没考虑过这个问题。

说了这么多，无非就是想说明一件事，写代码之前，先问问自己真的要做吗？能不做就不做，直到你有了足够的理由去做。对应到 Larry Wall 的说法，你要懒惰，花大力气去规避精力消耗。

做好自动化

说完了不要自动化的部分，再来说说要自动化的部分。

我还是先从你可能会忽略的问题入手，**你的日常工作是给别人打造自动化，但你自己的工作够自动化吗？**还是问一个更具体的问题吧！如果你写的代码要上线，会经过怎样的过程？

我先给你看一个极其糟糕的例子。刚开始工作不久，我有一次出差到客户现场。临近下班时，我发现了程序的一个Bug。在那个年代，我们的程序是按照官方推荐做法编写的 EJB（Enterprise JavaBean），今天很多年轻的程序员可能不了解了，它只有部署到应用服务器才能运行。

我的解决方案就是加上一些打印语句，然后部署到应用服务器上，看输出的结果，再加上另外一些语句，再部署，如此往复。那时我们完全是手工打包上传，每次至少要十几分钟。最终，定位到了问题，只修改了一行代码。但几个小时的时间就这样被无谓的消耗了。

那之后，我花了很长时间研究怎么做自动化的增量部署，最终让这个过程简化了下来。但这件事对我的影响很大，这是我第一次认识到一个部署过程可能对开发造成的影响，也让我对自动化在开发过程内的应用有了属于自己的认识。

相比于我刚开始工作那会。现在在工具层面做类似的事已经容易很多了，在后面的内容中，我会结合着具体的场景介绍一下现在的最佳实践。

你要懂得软件设计

最后，我们再来说说我们的本职工作，给别人打造自动化工具中需要的能力：软件设计。

软件设计，是很多人既熟悉又陌生的一个词，说熟悉，很多人都知道，做软件要设计，还能顺嘴说出几个设计模式的名字；说陌生，是因为在我的职业生涯中，遇到真正懂软件设计的程序员少之又少。**大多数人都是混淆了设计和实现。**

举个例子。有一次，我要在两个系统之间做一个连接器，让上游系统向下游系统发消息，或许你一听就知道了，这里需要的是一个消息队列。但实际上，我们需要的能力要比消息队列更丰富一些，比如，要将重复的消息去除。一个同事给我推荐了 Kafka 当作这个连接器的基础，我欣然地接受了。

不过，在后续设计的讨论中，我们就经常出现话语体系的分歧。我说，这个连接器要有怎样的能力，他会说 Kafka 能够如何如何。究其根因，我在讨论的是设计，而他说的是实现，所以，我们两个很难把问题讨论到一起。

为什么我会如此看重设计呢？**在软件开发中，其它的东西都是易变的，唯有设计的可变性是你可以控制的。**

同样以前面的讨论为例，尽管 Kafka 在当下比较火热，但是我不敢保证 Kafka 在未来不会被我换掉。因为就在几年前，消息队列还是传统中间件的强项，现在也渐渐被人淡忘了。

我不想让我的设计随着某一个技术选型而不断摇摆。如果工作许多年，知识体系只能靠各种新框架新工具支撑，我们做程序员就只剩下疲于奔命了。不懂软件设计，只专注各种工具，其结果一定是被新技术遗弃，这也是很多人经常抱怨 IT 行业变化快的重要原因。

回到 Larray Wall 的说法上，你要想写出一个别人挑不出毛病的程序，你先要懂得软件设计。幸运的是，软件设计这些年的变化真不大，掌握了软件设计再来看很多框架和工具，学习起来就会容易很多。在这个模块的后半部分，我会与你探讨软件设计的话题，降低自己给自己挖坑的概率。

总结时刻

Perl 语言的发明人 Larry Wall 曾经说过，优秀程序员应该有三大美德：懒惰、急躁和傲慢（Laziness, Impatience and hubris）。想要成为一个优秀的程序员，就要让机器为自己很好地工作，而这需要对自动化有着很好地理解。

我们学习自动化，先要知道哪些东西不要自动化，尽最大的努力不做浪费时间的事。一方面，我们要从需求上规避那些没必要做的事；另一方面，我们也从自身防止 NIH 综合症（Not Invented Here Syndrome），争取做一个懒惰的程序员。

对于要自动化的事，我们需要反思一下，在为别人打造自动化工具的同时，我们自己的工作过程有没有很好地自动化。而如果我们想拥有打造良好的自动化工具，我们需要对软件设计有着充分地理解。

如果今天的内容你只能记住一件事，那请记住：**请谨慎地将工作自动化。**

最后，我想请你分享一下，学习了本讲之后，你现在是怎样理解自动化的呢？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。

精选留言

Jxin

从老师这学到一个很喜欢的思想。有价值的事并不局限性于事情本身。做自动化很重要，写代码很重要。但根据根据现有情况判断是否需要自动化，是否需要写代码也很重要。有的放矢，任务分解。权衡跟设计是件很艺术的事情，令人着迷。

2019-03-18 12:37

作者回复

以现在大家的努力程度，少做点事是需要锻炼的技能。

2019-03-21 22:50

西西弗与卡夫卡

面试的时候，常常听到应聘者提起换工作的原因之一是手头任务重复性高，都是增删查改，代码粘帖复制。我就会问，你有没有想过把工作变得不那么重复，不要粘帖复制代码。有不少人就没什么话说了。

其实还是有很多可做的。比如自动生成增删查改的管理功能和页面，集成好缓存、搜索等服务。

懒惰真的是程序员的优秀品质，只是有些人理解成思想上的懒惰了

2019-03-18 00:33

作者回复

动手的人多，动脑的人少。

2019-03-21 22:52

like_jun

自动化。从重复的老动中解脱出来。

2019-03-22 07:50

Sudouble

最近确实感觉追随快速迭代技术追得自己挺迷茫的，也一直在思考到我究竟缺了什么。之前偶然看了软件工程相关的，才意识到算法、分析、设计等基础重要性，看到郑老师这一篇，让我更加坚信了这一点！

2019-03-20 22:43

作者回复

追变的东西，永远追不完，追不变的东西，就那么点东西。

2019-03-21 22:57

非鱼

软件设计除了设计模式以外，还有其他方面吗？

2019-03-19 15:58

作者回复

本模块后半部分我们来谈。

2019-03-19 20:13

毅

“懒惰”就要有付出，同时并不是想付出就能付出的，不到那个火候往往很难出一个好的自动化工具。前面也提到过，手里有锤子满眼是钉子，先权衡一下是否有必要再造一个轮子。如果有，该怎么做？是设计先行还是实现优先？怎么做好自动化，让团队效率和个人价值得到最大化，我想后续课程会一一道来！

2019-03-19 14:02

作者回复

期待值这么高，我努力啊！

2019-03-21 22:49

Y024

郑老师提到的那个问题，其实还有另外种解决方案：远程调试。当然这也无法避免部署的活，但是可以大幅减少部署的次数。有次出现了开发环境无法复现，只有测试环境才能复现的问题，就是通过远程调试解决的，当初使用的中间件是 WebSphere(估计很多童鞋都不知道了)。

2019-03-18 11:54

作者回复

远程调试，是一种重量级的工具，能不用就不用。

2019-03-21 22:46

hua168

老师，有什么好的软件设计的书推荐吗？目前没学过软件设计，后面有没有相关章节介绍的？

2019-03-18 10:08

作者回复

在这个模块的后半部分，我会讲到一些设计的话题，到时候如果还有问题，欢迎提出！

2019-03-21 22:46

西西弗与卡夫卡

另外，文中提到的TW顾问值得赞扬，能为客户着想

2019-03-18 00:31