44 | 完成作品：理想与现实

2018-11-12 胡峰

有时工作久了，会陷入这样一种状态中，整天不停地写代码，开发业务需求，周而复始，日子长了，自然觉着厌倦，感到似乎真的有点像“码农”了，日出而作，月落而息。在过去的某个时期，我应该也陷入过这样的循环之中，后来又是如何脱离的呢？

困境：代码与罗马

陷入这样一种写代码的“困境”，还是要回归到写代码这件事本身上。

写代码是因为有需求，需求来自业务的发展需要，经过产品经理再传递到程序员。刚开始，作为一个新手程序员，不停地为各种需求写代码。开发完一个，接着又是下一个，生生不息，循环不止。

一开始也许会感觉有些累，但并没有产生太多的厌倦。这是一个从不熟悉到熟悉再到熟练的过程，这里有太多的新东西可以去探索和尝试，让你在疲惫中依然能获得了好奇心的满足和成长的快感，因此不会感觉厌倦。

那技能从不熟悉到熟练需要多久呢？现在成为专家的要求已经有了共识：一万小时的刻意练习。但达成熟练要不了那么久，也许两三年足矣。有句俗语叫：“条条大道通罗马”。罗马，一座城市，包罗万象，类比到程序员这里就像一个个需要完成的业务需求。几年过去，每一条通往“罗马”的大道都被你走过了，再去往“罗马”时，自然找不到新鲜感了，困倦油然而生。

继续停留在通往“罗马”的循环往复中，已无法让你继续成长为专家。而要想跳出这循环往复的

路，无非就是不再去走那熟悉的条条通往“罗马”的大道，而是选择一条离开“罗马”的路，走出去，走向未知与未来。

在一万小时的刻意练习中，“罗马”已逐渐成为过去的熟悉、熟练区，而离开“罗马”便是要进入下一个陌生的学习区。但也许还会有一种 “现实” 的困境让你不得不继续走向当前的“罗马”，那么这时就不妨换一个视角：既已对通往当前“罗马”的所有路都了然于胸，闭眼可达，那就仔细观察了解现在“罗马”的构成与运作机制，也许将来有机会去创造属于自己的“罗马”。

从走向“罗马”到创造属于你的“罗马”，这里你的 “罗马”，就是你的作品。

理想：作品与创作

也许条条通往罗马的大道，堆砌罗马的砖石，有些已经消失在历史的尘埃中，但罗马作为一个时代和历史的作品，留了下来。

今天我们再看什么是作品？维基百科上对“作品”的定义是：

> 作品，亦称创作、创意、著作，是具有创作性，并且可以通过某种形式复制的成品。

从这个定义来看，作品最重要的特质是：**创作与创意**。所以，只有包含了创意和创作性质的事物才能叫作品。那对于程序而言，怎样才算作品？你从网上复制来一段代码，解决一个问题，这不是创作，也不会成为你的作品。

代码作品，可以小到一段函数、一个类，大到一个库或框架、一个服务，甚至一个系统。但打磨代码作品的方式，应该是定期对自己写完的代码进行沉淀、梳理和规整，提取可复用的功能，同样的功能只写一次，形成自己专属的编码脚手架和代码库。在以后的学习实践中定期反思，不断优化其效率和品质。

当你再碰到类似功能的实现时，能直接复用库就复用库，不能直接复用的就在脚手架代码上进行扩展，后续的重心就放在了优化实现思路上。这样日积月累下来，你的程序思维和能力才会变得科学、高效，而且产生积累效应。最终，这些留下的代码库或脚手架都成为了你的作品。

不过，同是作品，却有高下之分。吴军老师曾在文章里写过：“完成一件事，做到 50 分靠直觉和经验，做到 90 分要靠科学和技艺，而要做到 90 分以上则要靠艺术。”我是认同这个观点的，而且我们完成作品的目标应是 90 分以上，这是作品的特性决定的，因为创作就是艺术的核心所在。

到了 90 分或以上的作品，也许分数相差无几，但市场价值却可能差异巨大。iPhone 就是一个很好的例子，它当是一件 90 分以上的作品，90 分的工程技术加上几分的艺术，相比差几分的同类，在市场上的价值和价格却是遥遥领先。

作品，是创作的，创作是需要设计的，而设计是需要品味的，正如《黑客与画家》一书里所说：

> 优秀作品的秘诀就是：非常严格的品味，再加上实现这种品味的能力。
>
> 大多数做出优美成果的人好像只是为了修正他们眼中丑陋的东西。

也许，我们可以先从感知和修正代码中丑陋的东西来训练这样的品味和能力。

而完成作品的收益是什么？理想的情况下，也许我们期待直接从作品中获得经济收益，但这并不容易。十九世纪，有一名画家，一生作画数千幅，但只卖出过一幅，换回了四百法郎，这名画家就是梵·高。

梵·高的例子比较极端，他的作品都是 90 分以上的，但在直接换取收益方面依然困难。而对于你来说，今天的作品虽不一定能立刻给你带来经济收益，但在你打磨作品的过程中，把“条条通往罗马的大道”都走完了，甚至还反复走试图找到更优化的路线，这会让你掌握系统化的知识和体系化的能力，同时还会让你的作品变得更值钱。你可以想象这样一个场景：**当你给别人介绍自己时，只需要介绍自己的作品，而不再需要介绍自己的技能**。

成长的路上，写过的代码最终也许会烟消云散，但完成的作品会成为你点亮的勋章。

现实：产品与特性

作品要实现直接的经济收益，必须还要走完从作品到产品之路。

产品，是指能够供给市场，被人们使用和消费，并能满足人们某种需求的任何东西，包括有形的物品，无形的服务、组织、观念或它们的组合。现实情况是，大部分时候我们的代码作品，都是存在于产品之中的，所以你需要关注包含了你的代码作品的更大范围的产品及其特性。

如果产品无法获得市场的价值认同，技术作品自然也就埋没其中了。

有个说法是：要做好技术需要懂业务和产品。这大体没什么问题，但需要提到的细节是懂的方向。技术不需要了解业务产品的每一个显性特征，一个足够大的业务产品，有无数的显性特征细节，这些全部的细节可能分散在一群各自分工的产品经理们中。所以应该说，**技术需要懂的是产品提供的核心服务和流程，并清晰地将其映射到技术的支撑能力与成本上**。

另外，技术不仅仅需要支撑满足产品的全部显性和隐性服务特性，这些对于技术都相当于显性服务特性。而技术还有自己的隐性服务特性，这一点恰恰也正是高级和资深程序员需要重点关注的。所谓技术的隐性特性，通俗点就是程序员常说的非功能性需求，它的产生与来源都是源自程序和程序员本身。

用一段新算法实现的函数取代了旧函数，那么多余的旧函数就变成了负债而非资产，是需要去清理的。重构代码变得更简洁和优雅，可读性增强，节省了未来的维护成本。一个能同时服务一万人的程序实例，你知道你没法加十个实例就能简单变成能同时服务十万人的系统。这些都是技术冰山下的隐性特征，**显性的错误会有测试、产品甚至最终用户来帮你纠正，但隐性的错误却**

很难有人能及时帮你发现并纠正。

产品的显性特性就如泰坦尼克号，而技术的隐性特性则是泰坦尼克号撞上冰山后的反应。一旦隐性的错误爆发，就像泰坦尼克号撞上了冰山，一切外显的繁华最终都将沉入海底。

技术从特性到作品，去支撑产品的体验与服务，这是一条更现实的技术作品实现价值的路。

从反复写代码实现需求的重复困境中，到打磨作品实现价值的理想，再回归产品化的现实之路。

代码，有些人写着写着，就成了“码农”；有些人写着写着，就成了作者；还有些人写着写着，就改变了你、我的生活。那你想成为哪一类人呢？