我是如何把 github 开源项目做到 5300+ star 的

作者：star7th
原文链接：https://ld246.com/article/1578879911030
来源网站：链滴

前言

大家好，我是开源文档工具showdoc的作者。目前showdoc在github上有5300+的star，受到了部开发者的欢迎。我写这篇文章，是想与大家分享一下我在整个创作过程中所用到的技术以及相关技能如果你看完整篇文章觉得毫无营养，比某些摸鱼吹水的帖子要更没价值，那可以不再关注甚至举报。果你觉得还不错，可以让你有一点点的收获，欢迎捧场，加入讨论，甚至转发分享。

任务管理

无论是做自己规划的产品功能，还是做用户反馈过来的bug，最终都需要把这些点细化成一个个单独小任务，串联执行（毕竟自己是单个人力）。在这点上，我主要是利用团队看板来实现我的任务管理团队看板工具有好多，搜索一下就好，我个人目前在用tower.im

看板上我新建五个清单，分别是“在做”、“要做-高优先级”、“要做-低优先级”、“待定”、“留问题”，我这样分类是有技巧的。首先，它有优先级划分，保证先做重要的事。其次，它有状态管，方便我随时中断某个任务，过一段时间再回来继续任务。同时，它也能应对新任务——比如说新需进来，我先放在“待定”，等有空了再分配到其他清单去。最后，它还有“遗留问题”清单，可以让不在某个“疑难杂症”上卡住太多时间，快速推进下一个任务。

本地开发

用Linux或者MAC会非常方便开发、搭建测试环境等，对开发者的方便是毋庸置疑的。但是电脑有时需要玩游戏，需要装一些专业大型软件。从兼容性和广泛性的角度来考虑，我个人还是使用windows统。在win上，我使用的是Vagrant + virtualbox这个组合。Vagrant是一套工具，帮助你快速在win统上，利用virtualbox搭建起虚拟机，从而方便使用linux或者MAC(比如上架苹果app的时候我需要MC虚拟机)。关于这点，可以参考我几年前写的一篇教程：blog.star7th.com/2015/06/1538.html

代码管理

我用git作为代码版本管理工具，同时使用tortoisegit进行可视化操作。

说一下我是怎么维护官网在线版showdoc和开源版showdoc的。我在自己的服务器安装一个私有版的ogs作为私有git管理平台。功能开发好了，我再推送到github上去。官网版和开源版一开始是同一个库里的不同分支。后面它们的差异（尤其是后端代码的差异）越来越大，所以我干脆把它们分成两个同的仓库了。做开发的时候，我一般是现在官网版上做开发以及测试验证，然后再用tortoisegit的代修改差异比较功能，把功能迁移到开源版去。

说一下分支管理策略。我至少建立两个分支——主分支和开发分支。新功能会在开发分支做，验证好再合并到主分支来。用开发分支的时候也有技巧。比如说，一个大的功能可以基于开发分支再新开一分支去做。例如我今天想做A功能，那就在A分支上操作。但是A功能遇到瓶颈了，这会儿我暂时没精去找bug，所以此时我可以再基于开发分支开启B分支，继续做B功能。这很重要，因为在人力有限的况下，我做什么事情都是串联的，同一时刻只能做一件事。这样的策略有利于保证自己随时中断、随上手任务，灵活地安排不同的时间去处理容易的或者棘手的问题。这个过程也需要配合上面说到的团看板使用。中断前要记录好进度，方便自己随时恢复工作。

shell自动化脚本

我为showdo写了三个shell自动化脚本。其作用分别是自动化部署showdoc，自动生成api文档到shodoc，自动生成数据字典到showdoc。之所以选择shell而不是其他脚本语言（如python），是因为shll基本可以运行在绝大部分linux上，部分运行到win上（需要安装工具），兼容性超好，依赖非常少根据反馈，受到的好评比负面评论多得多。自动化脚本大大节省了使用者安装环境的麻烦，降低了sh

wdoc的使用门槛。showdoc大部分的用户不是php开发者，要他们安装php环境还是挺折腾的。有一键脚本，他们用得方便，我也省心，不需要在解决新手反馈上花太多功夫。这个方法是具有普适性，并且语言无关（因为一键脚本使用docker跑服务）。是可以大量应用到开源项目中，非常有利于开项目的代码分发。

顺便强调一下，做web应用开发的人，尤其需要克服一下对shell脚本的疏远感。我以前以web开发为的时候就会潜意识地疏远shell。在腾讯的时候向另一个技术栈的同事请教了下，发现其实还是蛮简单。只是因为自己过去心理上的疏远感导致自己没想过要去写shell。跨过这个心理槛，就会扩展自身的力边界，写起来跟其他语言没有太多区别，仅仅是需要多搜索查询下语法而已。

后端开发

可能是因为showdoc仅是一个小产品，涉及到的后台逻辑并不复杂，所以我在开发后端花的时间不多基本上都是CRUD，对数据库进行增删改查操作。需要多动点脑力的地方在于团队管理功能，新建多张数据表联合实现精细化权限控制。

showdoc后端主要采用的是国产框架thinkPHP。这个框架有好也有不好。不好的地方在于它的框架定、生态共享比较一般，好的地方在于它可以快速撸出一个东西来，而且兼容性还可以。这是四年前刚开发showdoc时的决定，后来也懒得换框架重写了，所以沿用至今。技术是相对落后了一些，但是胜在兼容性好，可以兼容老的php环境和一些老的服务器。这个还是挺重要的，因为showdoc是开发助性质的工具，协助写文档。兼容性越好就越容易被各种各样的群体接受。个人觉得这一点比用各种进技术要更实用一些。当然了，如果是现在新开发其他项目，我应该会使用laravel，或者NodeJS写的ggjs。

前端开发和UI

我在前端开发上花费的时间比后端开发时间多很多。可能是因为这是个体验型产品，需要把前端体验好。起步一个产品的前端页面时候，我建议一定要选好一个UI框架。比如我选择的是ElementUI做showdoc，而runapi则使用museUI（runapi是辅助showdoc用的一个在线api测试工具）。

showdoc用的编辑器是editor.md，是几年前的产品。虽然说它代码组织方式可能有点落后，且原作似乎有放弃维护的趋势。但我觉得它功能强大且简洁美观，所以我花了时间将它封装成了vue组件，且修改其源码增加了安全性。

项目拖拽和页面排序我用的是vue组件vuedraggable，还蛮好用的。以后有拖拽的需求我估计还是用。

图标设计方面，可以使用UI框架内置的字体图标，也可以用阿里妈妈的矢量图标库。或者自己使用Iconion进行图标设计。这里我强调一下Iconion。这个工具可以非常简单快捷地使用一些预定的图案和背，组合成一个快速可用的产品图标。showdc的产品图标就是这样制作的，快速省时地做到媲美专业效果。如果以后把产品做大了，可以考虑请设计师设计。但在项目前期，用Iconion就够了。

在这里要提一下前端技能的重要性。虽然说UI框架以及相应的工具能够帮助你节省很多时间。但如果做好一个产品的体验，那么其涉及到的UI和交互可能超出框架自带的范围。因此自己必须学会使用原css，会js交互，会封装组件。而且，前端技能跟下面要说的app多端开发也有帮助。

app多端开发

关于移动app产品原型设计，我很建议使用“墨刀”来做简单的原型规划。有了一个可视化的原型，的能节省很多大脑时间，让你在开发阶段的时候可以专注代码，而不需要写一下代码，又回头思考下步做什么功能，这样的来回切换相当耽误效率。

app开发我用的是uniapp，使用html5等前端技术把代码封装成手机本地app，包括安卓app、苹果a

p，甚至小程序。这种技术几年前就有了，但是性能一直不温不火。2019年的时候我看到了这块发展还是可以的。所以就产生了做showdoc app端的想法。不过由于showdoc重在pc web端，手机只是到辅助作用，所以我没有很精心去做。大概把web版简化一下，复用一些组件，重写下前端，然后就线了。顺便说一下，我做得比较精细化的app是时光树洞（blog.star7th.com/2019/09/2380.html）这款app的UI就花了点心思。

如果要让我评价这种混合型app和原生app，我会说，肯定原生app最好。只是出于成本和人力的考，对一些展示型的、交互体验要求不那么高的产品，可以采用这种h5技术处理。大家如果在验证市场段，可以采用类似技术快速做一个可用产品出来。

此外说一下苹果app上架的问题。我是利用虚拟机安装了一个黑苹果系统，然后装相应工具，把app传到苹果商店去。关于如何开通苹果开发者账号，如何在虚拟机安装苹果系统，这个你可以自行搜索。

用户反馈和社区运营

一开始，用户反馈消耗了我不少精力。敢情自己成为一个客服了。后来我开始做了一些改变，基本把己从这些反馈中抽身出来了。

首先我分开了官网在线版和开源版的反馈，开源版的反馈统一到github（gitbug的使用门槛能过滤掉些非常新的新手，避免新手问题），在线版的反馈统一发邮箱。有功能或者bug要改，我先记在团队板。而对于一些常见问题，我整理好了文档，和一些固定的回复术语。另外我也开了三个qq群，供showdoc使用者自身交流。不过我要提的一点是，千万别试图回答qq群里提的每一个问题，会把人逼疯。所以我更改了群公告，改写了群定位，将qq群定位为用户自行交流的地方。让热心用户去回答新手使用showdoc时遇到的问题。而我则只需要很偶尔看一下。需要官方的支持还是让用户走github或邮件通道。

不过值得一提的是，我这种运营思路是不适合所有团队的。我是人力有限，效率优先，所以过滤了很不太必要的反馈。假如说有很足够的人力，我倒建议可以多花时间帮助用户解决问题，既扩大用户量又提高产品口碑。

后记

先讲这么多。其实还有很多东西可写，上面很多点也可以细化成一篇篇独立的文章。先看看用户反响。如果没人看，那就这样吧。如果有不少人觉得有帮助，我再写点什么，或者扩充上面的小点成独立章，发出来或者写到自己的博客上。