拓展：如何做好一个开源项目（下篇）

持续运营

项目有了一定的规模和进展后，需要持续运营，让更多的人知道和使用。运营并不是个技术活，对于程序员来说，还是或缺的技能。最简单的运营手段，就是在一些技术社区分享“软文”，iView 在早期就是这样做的，还总结出了一个 “500 star 定律”，也就是说，每一次分享文章，差不多能在 GitHub 带来 500 个 star。star 对于一个开源项目来说，还是蛮重要的，它直接决定了用户是否会选择你的项目，但用户都是程序员，又不傻，如果项目质量低，star 就变的一文不值了，还会坏了口碑。切记，不要刷 star，前端圈堪比娱乐圈，会被针对的很惨。

当然，不是什么内容都能发，比如更新日志，最好就不要发了，除非像 2.0、3.0 这种大版本。即使是发大版本的更新内容，也不是说把更新日志一贴就完事，如果你足够重视你的开源项目，就应该重视每一篇文章，把更新的核心思路说清楚。典型的案例可以参考 iView：

- iView 发布 3.0 版本，以及开发者社区等 5 款新产品 (https://juejin.im/post/5b5e6f19e51d4515b01bf36e)
- iView 近期的更新，以及那些“不为人知”的故事 (https://juejin.im/post/5aa5d56e5188255563 5df2b2)
- iView 发布后台管理系统 iview-admin，没错，它就是你想要的 (https://juejin.im/post/59e6a995518825619a01c433)

标题的重要性就不必说了，在信息爆炸的时代，你的标题不够吸引人，根本没人看。

目前，有几个社区是值得关注和积累粉丝的：

- 掘金：比较活跃的程序员社区，前端属性较浓，社区运营做的很好，对开源项目有扶持，相关的文章首次亮相，官方都会给予一定的资源支持。
- 知乎：流量最大的社区，大 V 属性，如果你是初入知乎，可以把文章投稿到热门的专栏，比如前端评论外刊、前端之巅等。因为自己起初是没有粉丝订阅的，发表了也不会有人看到，投稿就不一样了。而且，被某个大 V 赞一下，那效果就像中奖。
- v2ex：不用解释，就是很火的社区。

开源项目，一般都会在 GitHub 托管，不过也可以在开源中国（Gitee）同步一份，每个版本的更新日志，可以以新闻的形式，向开源中国投稿。开源中国在国内还是有一定的影响力的。

除了发表文章，一些技术分享大会也可以关注，可以以公司的名义申请成为嘉宾做分享。如果有机会，还可以到其它公司做技术分享，尤其是大厂商，这些都是难得推广开源项目的好机会。你的开源项目，如果有几个 BAT 这类的大厂使用，那会成为维护者、社区用户和观望者的信心来源。

还有发布会。在国内，开源项目搞发布会的，据我所知只有 iView。没错，18 年 7 月，iView 搞了一场新品发布会，线下进行，线上同步直播，当时有超过 2 万的在线用户观看，推广效果还是不错的。一场“合格”的活动，要分**活动前**、**活动中**、**活动后**。活动开始前一个月，就要散布消息，让用户有个初步印象，之间还可以爆料一些活动热点；活动进行中，要有专人负责现场，还要与观众互动；活动结束后，要加个班，把核心内容整理为文字，在第一时间通过官方渠道发表出来。这种大规模的活动，没有公司支持，个人很难完成的，因为这不是一两个人的事，需要很多工作人员一起完成，幕前幕后、直播的网络、现场 wifi，还要应对各种突破状况，不过，最重要的还是活动内容的策划准备了，否则一切都是纸上谈兵。

讲到这，你可能会说，老老实实做技术不好吗，非要弄这些花里胡哨的东西。的确，推广这件事，并不是做开源必须的，老实做技术没有错，推广只是让你的开源项目更快传达给目标用户。做这些事的目的就一个，让更多的人使用你的开源项目，让更多的开发者参与贡献代码。

最后一点，如果你的公司或团队有经费，适当投放一点广告也是不错的。

国际化

是时候与世界接轨了。一般来说，国际化（Internationalization，简称 i18n）分 3 个部分，首先是你的开源项目支持多国语言，对于 UI 组件库来说，这个还是很好支持的，只需要提供一个多语言的配置文件就行，每种语言一个文件，然后由社区贡献更多的语言。以 Vue.js 为例，社区也提供了 vue-i18n 插件，那你的组件库还要兼容 vue-i18n，可能还要考虑兼容多个主流的版本。

另一部分是文档的国际化，除了中文，至少应该提供一个英文版本，毕竟英文算是通用的语言。如果文档内容不多，可以让社区来提供更多的翻译版本。维护多语言的文档是一件很辛苦的事，这意味着每一个版本更新都是中英双语的，并不是说文档翻译一遍就不管了。好在翻译文档是个一次性的技术活，前期多找一些英文好的热心用户一起翻译，后面只要确保每次更新都保持中英双语就好了。

做国际化，意味着要服务国际友人，那就不能强求他们用微信或 QQ。在开源界，比较通用的是 Gitter (https://gitter.im)，只需要关联一次 GitHub 的 repo 就行。除此之外，官方可以在 Twitter 开通一个账户，来更新一些动态，与其它 Twitter 互动。Discord (https://discordapp.com/) 也是技术圈比较热门的一个 App，以 Vue.js 来说，你可以加入一个名为 **Vue Land** 的服务器，在里面找到 **#ui-libraries** 的频道，就可以和全世界的开发者讨论组件库的话题了。

支持国际化，短期来看，是一件付出回报比很低的事，但从长远利益出发，对国际化的支持，有助于更多的国外开发者成为核心 maintainers，让全世界能够参与进来，才是开源的意义所在。

让更多的人参与

开源项目从来就不是一个人的事，一个健壮的开源项目，需要不断有人贡献代码。在项目有了一定知名度和使用人群后，自然会有不少 PR 进来，知名的开源项目 contributors 都有几百人，哪怕修改一行代码，只要被 merge，就算一个 contributors。最核心的维护者一般不会超过 5 人，而且除了作者本人，很多都是阶段性的，毕竟是开源，大多数人还是兼职做的，能贡献一点是一点，业务忙了就没顾不上了。

为开源项目贡献代码，主要以 PR 的形式进行，作为一个开源项目的 owner，即使 organization 的其他成员有直接 commit 的权限，也应该建议他们提交 PR，而不是直接 commit，owner 需要认真 review 每一个 PR 以确保代码质量。修复一个问题最怕的，是引起新的问题，或导致以前已修复的问题又复现，有时候，contributor 可能只为了 fix 某一个 issue，但它对整个项目是不了解的，而且对以前“发生的事情”都不了解，会导致一些他看不到的问题，这种情况作为 owner 就要认真审查了。

参与一个开源项目的方式有很多，除了最直接的 PR，还可以 review issues。项目活跃时，每天都会有不少 issues 进来，owner 可能没时间及时处理，但可以**打标签**（labels）。一个 issue 被标记了 label，说明已经审核过此 issue，常见的 label 有以下几种：

- **bug**：已确定为 bug；
- **feature request**：已确定为请求新功能；
- **invalid**：无效的 issue，一般可以直接关闭；
- **contribution welcome**：owner 可能暂时没有精力处理，期望社区来贡献代码；

- **provide example**：issue 需要提供复现示例；
- **discussion**：暂时无法断定，需要进一步讨论；
- **may be supported in the future**：先标记一下，也许未来会支持。

管理 issues 的另一个方式是用好**里程碑**（milestones）功能。milestones 可以按照版本号创建，把期望在这个版本解决的 issues 添加进去，发版前对当前 milestone 的所有 issues 集中查看，是否都处理完成了。

有一些**有价值**的 issues 可能会耗费不少精力处理，而且社区很多用户都希望能够解决，owner 当然也希望处理，只是没有时间。这种情况不妨**有偿悬赏**，推荐一个新起的国外社区 IssueHunt (https://issuehunt.io/)，用户可以为某一个 repo 的 issue 众筹，谁处理了，就可以得到全部赏金。

每一个版本发布后，记得创建一条 release，这样做一是有一个版本更新日志记录的地方，二是 watch 你项目的人都可以及时收到邮件通知提醒升级，三是 release 会打一个 tag，其它贡献者可以切换到此 tag。release 最好不要在发版前再创建，不然整理起来很费劲，建议每个 release 发布后，就新建下一个版本的 release 作为草稿（draft），处理一个 issue，就记录一条，避免遗漏。

版本号也是有讲究的，比如 3.2.1，这里的最后一位，代表只有 bug fixed，中间一位代表有 new features，第一位代表有 break changes。一般来说，除了第一位，剩下的版本都是兼容式的，就是说用户升级后不会影响当前项目，如果有 API 的变更，应该发布第一位版本号。

代码贡献越活跃，贡献者越多，开源项目也越健壮，作为 owner，应该及时联络有价值的贡献者，一个人的能力毕竟是有限的。当你与世界各地讲着不同语言的的人，一起完成一个开源项目，会觉得开源真是一件了不起的事情。

让 Robot 来做“坏人”

开源项目有一定的规模后，社区就会很活跃，每天都会有大量的 issues，这些 issues 越积越多，不及时处理掉，对 owner 来说就是精神压力。在项目初期，由于使用者不多，是鼓励提 issues 的，建议、新功能请求、bug 反馈、问题咨询等各种内容都可以提交，而且作者有足够的时间和精力来认真回答。到了一定规模后，可能什么 issues 都会出现，不乏一些带有恶意的、言语攻击的，如果直接关闭 issue，可能还会继续“纠缠”，说 owner 态度不好之类的，这些都是笔者亲身经历过的。

除了恶意的 issues，还有很多 issues 不符合格式要求，连代码格式化都没有，甚至连问题都说不清楚，也没有描述，就一个标题，这些无效的 issues 一个个回复都会占据大量的精力，直接 close 还会被说没处理怎么就关闭了，实属无奈。

这时你需要一个 GitHub 机器人来充当“坏人”的角色，也就是注册另一个 GitHub 账户，用它来处理一些不符合要求的 issues，这是一个很聪明的做法，关闭 issues 这些活都让 robot 来操作。比如 iView 的“坏人”就是 iview-bot (https://github.com/iview-bot)，不过它是一个智能的 robot，不需要 owner 控制，会自动关闭不合格的 issues 并回复提问者。GitHub 提供了 API 来接收每一个 issues 并通过 API 来操作 issues，包括关闭、打标签、回复等，只要给 robot 设置足够的权限就行。比如 iView 的 issues 机器人代码是 https://github.com/iview/iview-bot (https://github.com/iview/iview-bot)。用户如果直接通过 GitHub 提交 issues，会被 robot 立即关闭，并回复：

> Hello, this issue has been closed because it does not conform to our issue requirements. Please use the Issue Helper (https://www.iviewui.com/new-issue) to create an issue - thank you!

就是说，用户必须通过 Issue Helper (https://www.iviewui.com/new-issue) 这个页面提交 issues 才可以，不是通过它提交的，会被检测出来立即关闭。在这个页面中，用户需要提供详细的描述才能通过表单验证。issues 只接受 bug 报告或是新功能请求 (feature requests)，对于使用咨询等其它问题，都不能提交，而是鼓励到 Stackoverflow 之类的社区讨论。如果是 Bug，还必须提供能够最小化复现问题的在线链接，以及详细的复现步骤。

robot 还有一个作用：翻译。它会把中文的 issues 自动翻译为英文并把翻译内容自动创建一条回复，同时标题也会修改为英文。开源项目到这个规模，使用者和贡献者不仅仅是中国人了，世界各地的开发者都有，使用英文会让所有人都看懂 issues。

虽然 robot 能自动过滤 80% 不合格的 issues，但仍有浑水摸鱼的用户跳过这些验证，这时可以给 robot 设置一些快捷回复，人为来 comment & close：

- Hello, this issue has been closed because similar problems exist or have been explained in the documentation, please check carefully. *已有相同 issues, 或文档有说明*
- Hello, this issue has been closed because it is not required to submit or describe is not clear. *描述不清楚*

- Hello, this issue has been closed because it has nothing to do with the **bug report** or **feature request**. Maybe you can ask normal question through SegmentFault (https://segmentfault.com/t/iview) or stackoverflow (https://stackoverflow.com/). *不是 bug 反馈或 新功能请求，请到社区讨论*

- Hello, this issue has been closed because it is not a bug, but a usage problem, please consult other communities. *用法不对*

- Please provide online code. You can quickly create an example using the following online link：https://run.iviewui.com/ (https://run.iviewui.com/). *没有提供在线示例*

其实呢，这个“坏人”也没那么坏，还是挺可爱的。

赞助与商业化

开源项目的发展离不开资金的支持，向社区寻求赞助并不是一件“羞耻”的事情，而是理所当然的。

最简单的赞助方式就是通过二维码打赏，不过这种方式在国内几乎没有什么用，中国的开发者大多比较“囊中羞涩”，而且由于打赏的匿名性（微信），时不时收到个 1 分钱，也就呵呵了。

这里推荐几种比较好的“募资”方式：patreon (https://www.patreon.com/) 和 opencollective (https://opencollective.com/) 是开源项目最常用的，可以一次性支持，或周期性，以美元结算，可转至 PayPal。不过这两种都是美元，而且转到 owner 这里，扣除手续费可能少很多，不过对赞助者（往往是企业）来说，好处就是有发票。另一种方式是通过开源中国来赞助，开源中国的用户还是比较慷慨的。

另一种是投放广告，这里推荐 Carbon (https://www.carbonads.net/)，不同于 Google Ads 的是，它的广告都是与互联网相关的，而且样式可以完全自定义，很美观，不会让用户产生反感，广告根据展示和点击转化付费。Carbon 的中国市场负责人中文很溜哦，作为中国开发者，不用担心谈不来。

不过呢，最值得推荐的还是接入**品牌广告**，但前提是你的文档要有一定的流量。开源项目的文档有着最大的特点：访问者几乎都是程序员，所以你要是挂个某多多的广告，几乎会被喷死。在线教育、云主机服务商都是不错的选择。一般不会有人主动联系 owner 投放的，除非像 Vue.js 这种级别的，但你可以尝试发一封友好的邮件来询问。不知道发给谁？告诉你个好办法，去其它社区（比如 v2ex）看看都有哪些金主投放就知道了，既然已经投放，说明有投放广告的需求，都是潜在的目标“客户”。

再来说说商业化。

开源并不是意味着免费，根据开源协议的不同，有的开源软件在用于商业时，可能要购买授权，源码是开放的，但不一定可以免费使用。不过能够收取授权费，也说明你的软件确实无可替代。企业为了避免不必要的纠纷，肯定是愿意购买你的软件的。但是对于大多数 MIT 的开源项目，可以商业化吗？答案是肯定的。

首先要知道，能够付费的，都是企业，而非个人，个人也没有付费的必要。一种比较常见的模式就是软件免费，然后可以向企业提供额外的付费咨询服务或顾问。最懂开源项目的人，绝对是这个项目的 owner，如果企业是深度用户，还是很愿意支付一些费用来咨询问题的。我是做 to B 业务的，我们公司也是做 to B 的，公司高管大多也来自 Oracle（算是比较大的 to B 企业了），所以我对企业服务也有一定的理解，一款好的产品，绝对是技术加咨询服务。

商业化还是有很多方式的，具体要看开源项目的类型。以组件库为例，它本身是免费的，也可以无限制免费使用，但可能提供付费的高级组件或模板系统，以及其它生态产品，比如基于组件库的 IM 系

统。

当然了，并不是所有的开源项目都要商业化，大部分还是完全免费的，商业化也有利弊，如果没有一定的实力，很有可能搞砸哦！

以上，就是我从事开源工作两年多的一些浅薄经验，希望能给聪明的你带来帮助。

结语

每个开发者，都应该尝试维护一个开源项目。

每个开发者，都应该抱着一颗敬畏之心使用他人的开源项目，而不是“用你的是看得起你”。

每个开发者，都应该适当地赞助一个帮助过你的开源项目。