一、项目的基本背景和发展历程介绍

1.1项目背景与技术类型

在编写程序过程中，我们常常需要用到很多数据来进行测试。如果要是手动制造数据的话，肯定要花费大把精力，这不合理。此时我们可以使用Faker这个库，用它来生成各种各样的伪数据。Faker 是一个 PHP 库，可以自动为用户生成假数据。 无论您是需要引导您的数据库、创建美观的 XML 文档、填写您的持久性以对其进行压力测试，还是对从生产服务中获取的数据进行匿名化，Faker 都非常适合。 在Faker的仓库中，我们可以看到，Faker是一个完全由PHP语言编写的库。该项目在github上目前拥有26.4k stars和3.6k forks。项目最初的开发者**François Zaninotto(fzaninotto)**于2011 年 10 月开始开发 Faker，因为他需要为其的一个项目使用假数据填充数据库。也正是因为在这个项目的贡献，他在 PHP 社区中变得更加知名。然而，尽管目前Faker库仍然受欢迎，但从个人精力和项目更好的发展的角度来考虑，François Zaninotto却打算交出这个项目的维护权。一下是其在个人博客中的原话：

> A year ago, I realized that I needed someone to ~~take the curse from me~~ hand over the Faker maintenance.

1.2 版本发布历史

fzaninotto/Faker总共发行过12个Releases，原始版本v1.0.0发行于2013年10月22日，并于同一天更行了v1.1.0和v1.2.0。之后，fzaninotto几乎保持着每年一更的速度。目前的最新版本v1.9.2发行于2020年11月11日。相比于上一次更新，此次更新做了安全加固。

1.3 主要贡献者

在github上，该项目拥有452名贡献者，他们来自世界各地，使用70多种语言(这也导致了项目维护的困难)。事实上，根据fzaninotto自己的解释，该项目近些年来主要是靠着这些contributers才得以更新，因为与他本人而言，至少已经5年没有写过一行PHP代码了。

该项目的第二大贡献者**Andreas Möller**是一名来自柏林的软件工程师，他在 ergebnis 组织中维护开源项目，为广泛的开源项目做出贡献，并定期参加社区活动。

此外，我还在项目的贡献者列表里发现了一位来自上海的工程师***Song***，他目前在腾讯工作。

1.4 CI/CD 的使用

CI/CD是一种通过在应用开发阶段引入自动化来频繁向客户交付应用的方法。CI/CD的核心概念是持续集成、持续交付和持续部署。具体来说，CI/CD可让持续自动化和持续监控贯穿于应用的整个生命周期（从集成和测试阶段，到交付和部署）。这些关联的事务统称为“CI/CD管道”，由开发和运维团队协同支持。它的好处有以下两种：

- 快速发现错误。每完成一点更新，就集成到主干，可以快速发现错误，定位错误也比较容易；
- 防止分支大幅偏离主干。如果不是经常集成，主干又在不断更新，会导致以后集成的难度变大，甚至难以集成。

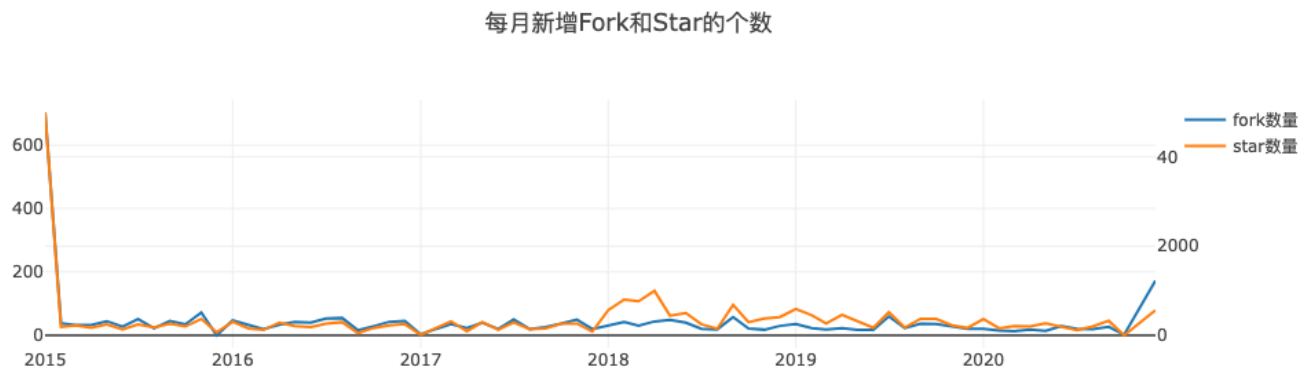
在该项目存放在代码仓库的`.github/workflows`目录下，有一个`continuous-integration.yml`文件。GitHub只要发现 `.github/workflows` 目录里面有.yml文件，就会自动运行该文件，实行CI。顺便提一下，这个文件就是上面提到的Andreas

1.5 其他有用的信息

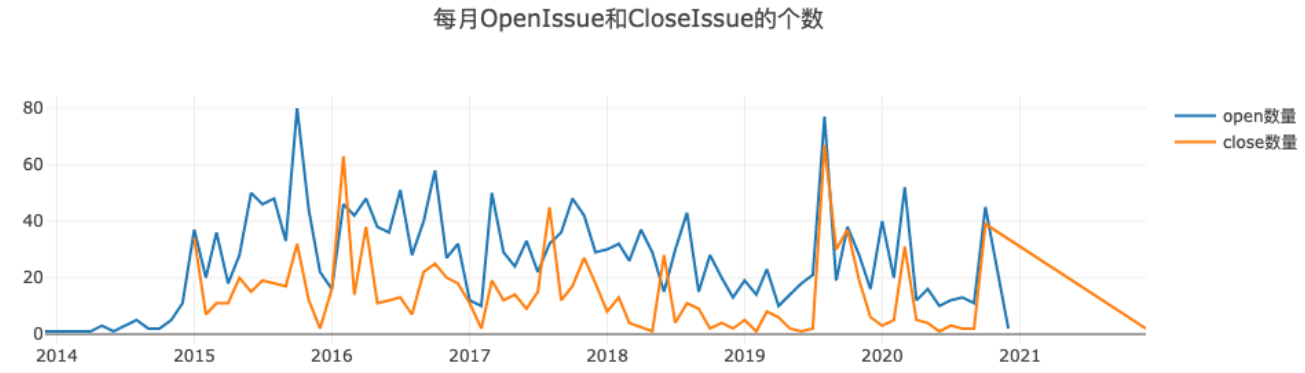
在这里https://marmelab.com/blog/2020/10/21/sunsetting-faker.html，François Zaninotto发表了一篇个人博客，在这篇博客里面，他详细阐明了自己对于Faker项目的看法以及以后的打算，并且文笔非常好。从一个项目开发者的角度来谈这个项目，十分有趣。

二、项目的历史轨迹分析

2.1 每月新增 Star 和 Frok 的个数



2.2 每月打开 Issue 和 关闭 Issue 的个数



2.3 每月打开 PR 和合入 PR 的个数

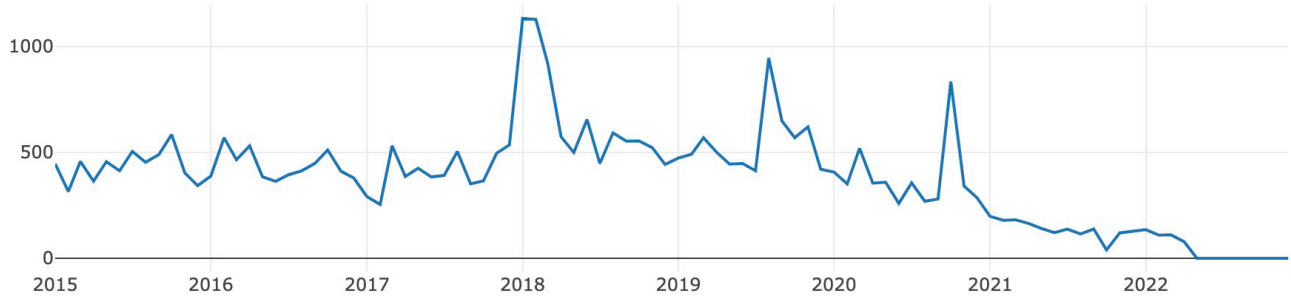
每月打开PR数量

2.4 每月在仓库中活跃（只要有日志产生就算）的不同开发者（也就是一个GitHub账号）总数

每月活跃开发者

Year	2015	2016	2017	2018	2019	2020	2021	2022
1	315	569	254	1127	491	351	179	109
2	457	465	531	918	569	519	182	111
3	364	531	386	574	500	355	165	78
4	456	385	425	499	445	359	141	0
5	413	363	384	655	447	259	121	0
6	504	394	391	447	413	356	138	0
7	453	413	504	592	945	269	115	0
8	489	448	351	553	648	280	138	0
9	584	511	365	554	569	834	39	0
10	403	411	496	522	620	341	120	0
11	343	379	535	443	420	286	130	0
12	388	291	1132	473	407	198	135	null

每月活跃开发者



2.5 Issue 从打开到关闭的平均时长和中位数（单位：天）

Issue 从打开到关闭的平均时长为 92.94天
Issue 从打开到关闭的中位数为 2天

2.6 PR 从打开到合入的平均时长和中位数（单位：天）

PR 从打开到合并的平均时长为 49.64天
PR 从打开到合并的中位数为 19天

2.7

ssue 从创建到第一次有人回复(非本人)的平均时长为 126.24天
Issue 从创建到第一次有人回复(非本人)的中位数为 2天

任务8 关键的时间节点

1. 项目起步阶段的2015年1月，新增issue和star数量非常多
2. 在2018年1月到4月，项目受到了井喷式的关注，新增了很多issue和star，此时的月活跃开发者数量也达到峰值
3. 在2019年8月，issue的打开、关闭量，以及PR打开、合并量达到峰值

三、洞察项目被归档的可能原因

3.1 项目可能的归档原因

1. 大多数人只使用1个语言环境，即只需要库大小的一小部分。然而Faker项目一开始就被设计为多语言库，即使只需要 10KB，他们也必须下载 3MB，别无选择。Faker应该是一个没有本地化内容的小型核心库。然后来自世界各地的开发人员将向packagegist贡献本地化包，而无需仓库拥有者监督合并。这样用户只需要下载他们需要的核心和语言环境。但是这个问题在项目归档前，唯一的解决方法就是完全重写，所以这可能是归档的最大的一个原因。
2. 从Faker作者的博客中了解到，在Faker项目归档前，他已经5年没有写过一行 PHP代码。他有尝试过交出Faker仓库的维护权，但是后续开发者的重写都很难令人满意，进展也非常缓慢，所以基于第1点原因，以及这里对开源社区所有人利益的考量，作者决定将Faker归档。

3.2 项目归档后可能产生的影响

1. 归档一个仓库会让它对所有人只读（包括仓库拥有者）。这包括对仓库的编辑、问题issue、合并PR、标记、设置milestone、发布、设置标签、分支处理、反馈和评论。所有开发者和用户都不可以在一个归档的仓库上创建新的问题、合并请求或者评论，但是仍可以 fork 仓库——以允许归档的仓库在其它地方继续开发。
2. 对用户而言，程序仍然可以正常运行，只是版本不会再更新，随着时间的推移，当前程序会越来越过时，直至也被用户淘汰。
3. 对开发者而言，仓库归档意味着丧失了一个共同交流、开发的平台，虽然可以fork后进行个人开发，但是不能再统一地进行优化和升级，不过普遍来讲，一个归档的项目，必然是存在着已很难或者完全无法解决的问题，也很少有开发者fork后建立第二平台。

3.3 对开源项目如何可持续发展的理解

- 开源项目要可持续发展，需要人为地引入开源治理模型，并进行良好的治理，才能实现项目的可持续性。无论是作为贡献者加入开源项目，还是创立开源项目，都要遵从开源治理模型，通常会由一群由社区选举出来的有积极贡献的开发人员作为决策者，他们能为项目的未来做出技术决策。
- 不同的项目的开源治理模型也有区别，这里引入三类模型
    1. 选举：这种模式允许项目通过选举进行治理。贡献者会投票给候选人以填补各种项目角色。通常，遵循此模型的开源社区会建立并记录选举程序。

2. 企业支持：个别公司可以选择根据开源许可条款分发他们的软件，作为接触潜在开发人员和用户的一种方式。在这种模式下，管理组织可能拒绝接受外部的贡献，或者需要签署贡献者协议（CLA）为前提来接受贡献。
3. 基金会支持：一些项目可能会选择由非营利组织或行业协会管理。该模型确保单个项目参与者没有对项目资源的独占控制权。在某些情况下，基金会领导和项目领导可以形成一个单一的治理结构，全面地管理开源项目。在其他情况下，基金会管理一些事务，例如商标和社区活动，但其他治理结构（例如代码批准）则由项目负责人处理。

- 开源治理模型能给开源项目带来许多优势。首先，它降低了项目放弃或不被维护的风险。其次，项目治理消除了单一供应商控制的问题。当项目开始取得一定的成功时，如果没有开源治理模型，供应商将有可能做出有利于公司战略但不利于整个生态系统的选择。最后，开源治理模型为创新提供一个安全的摇篮。也就是说，当项目有一个开放的开源治理模型时，开发人员能够相信他们的贡献将基于相应的优势和对项目最有利的方式被评估。这种方式能够吸引具有不同经验和背景的人来为开源项目赋能。
- 综上所述，对于开源项目而言，制定合理的开源治理模型，并严格地执行、完善，是开源项目可持续发展的关键。