



- [首页](#)
- [最新文章](#)
- [IT 职场](#)
- [前端](#)
- [后端](#)
- [移动端](#)
- [数据库](#)
- [运维](#)
- [其他技术](#)

- 导航条 -

[伯乐在线](#) > [首页](#) > [所有文章](#) > [业界](#) > Steve Yegge对Amazon和Google平台的长篇大论

Steve Yegge对Amazon和Google平台的长篇大论

2011/11/03 · [业界](#) · [1 评论](#) · [Amazon](#), [Google](#), [Steve Yegge](#), [亚马逊](#)

分享到：

来源：[陈皓](#)

Steve Yegge， Amazon的前员工，现任Google员工，其本来想在Google+上和Google的员工讨论一些关于平台的东西，结果不小心把圈子设成了 Public，结果这篇文章就公开给了全世界，引起了剧烈的反应。发布后很快他就马上把这篇文章删了，不过，互联网上早备份了下来——[SteveY’s Google Platforms Rant](#)。后来，Steve在[其Google+上作了一些解释](#)，大体是说他喝多了，而且又是在凌晨，所以大脑不清，文章中的观点很主观，极端且不完整，还有Google的PR对他很好，等等，等等 。

几个星期前看到时就一直都想翻译一下这篇文章，不过因为最近事情太多，文章又很长，所以现在才翻译完成，翻译的不好，还请大家指正。

导读

在你阅读正文以前，我想说明几点，希望你注意一下：

- Steve这个人非常喜欢写长篇大论的东西。而且比较喜欢辛辣调侃和恶搞的文风，这点大家要注意！
- 文中先“骂”Amazon公司，再通过“骂”Amazon的创始人贝索斯Bezos并烘托出他的的悟性和雄心，最后教育了一下Google。
- 我把文章分成了三个部分，这样方便大家阅读和讨论。第一部分只是个人情绪化的抱怨，第二部分是说Amazon的成长，第三部分是教育Google，我觉得第二部和第三部分是重点。
- 对于我们来说，我们应该获取Steve那些关于平台（Platform）相关的那些有价值的观点。尤其是他说的Amazon如何进化成一个平台性的公司，以及阐述Google应该怎么做的那些观点。
- 关于对Amazon的那些指责，我想说，6年，对于一个世界级的互联网公司，已经很不一样了。

正文

第一部分

我曾在Amazon工作了六年半，现在，我在Google的日子也这么长了。对于这两家公司，有一件事总是萦绕关我——这种感觉一天比一天强烈—— 那就是，Amazon每件事都做错了，而Google每件事都做对了。当然啦，这是很笼统的话，但却是惊人的准确，相当的疯狂吧。大概有一百甚至两百种不 同的地方可以让我们去比较这两个公司，而Google可能在每一项都能胜出，如果我记的没错，除了其中3项以外。因为，我曾用电子表格把这些项都列出来 了，只是法务部门不会让我给任何人看，即使人事招募部门很喜欢这个报表。

这里，让我先给你个例子让你稍微体会一下：Amazon的人事雇用流程有根本上的缺陷，因为各个团队各招各的人，以至于，各团队之间的招聘标准相当 的不一致性，即使他们通过各种努力来统一标准，但是实际操作上却是一团糟；他们没有真正的SRE（陈皓注：Site Reliability Engineer ），工程师们什么事都要做（陈皓注：SDE – Someone Do Everything）、几乎没时间编码。当然，不同的部门有不同的情形，不过，这取决于你的运气。他们不搞慈善，也不帮扶贫困人群，也不搞社区贡献，或 是其它相似的活动。在那里，他们从来不谈这些，或许只有在说笑话的时候才会提到。他们的办公环境是个灰尘及污迹四处的像农场一样的隔间，他们在公共区域连 一分钱装修的都不会花，而且，他们的薪水和福利相当差，只是近来与Google和Facebook竞争人才，这个差距才变得非常地小。不过，他们没有我们 有的津贴或额外奖金——他们只是给你录用信上的那个数字，就这么多。他们的程序代码完全就是灾难，无论什么都没有任何的工程标准，除了各别团队有一些。

公平起见，他们的确有套非常非常不错的版本控管系统，而这是我们（Google）需要尽力赶上他们的地方，他们还有一个漂亮的发布/订阅系统，我们 也没有相对应的东西。不过，就大体而言，他们有的不过是一堆蹩脚的工具，用关系数据库来读取或写入状态机里的信息中罢了。我们不应该这么搞就算这样做是可 以。

这就是我所所说的那3件事中的两件事Amazon比Google强的，那就是的他们的发布/订阅系统以及版本控制系统。

事都里，包括正在纪 律或是其他一堆可能云让其他时间的事务。所以，即使这么做让他们在市场中有了某种程度的竞争优势，但也造成其他足够多的问题，总之，这样的做法算不上是个 漂亮的扣篮。

但是，他们有一件事做的非常非常好，其好到可以把其他政治，理念，技术上的消耗和混乱和**完全**弥补回来。

第二部分

Jeff Bezos是个臭名昭彰的微管理经理人，他的微管理都管理到了Amazon零售网站上的每一个显示像素。他雇佣了Larry Tesler——Apple的首席科学家，他可能是全世界最有名也最受尊敬的人机接口专家，然而，Bezos忽略了Larry三年来提出的每一个建议，直 到 Larry最后——明智地——终于离开了公司。Larry本应做一些大型可用性（Usability）研究，并可以系统地了解那个根本就没有人能够搞懂、使用那该死的网站，可是，Bezos对于那些像素不放手，这些页面上的那几百万个显示像素就像是他的孩子一样。所以，他的这些孩子还留着，而 Larry没有。

当然，微管理不是第3项Amazon做的比我们好的事。我的意思是，没错，他们微控管理做地非常地好，但我不会把这项列在他们的强项清单上。我这样 说只不过是为我下文做铺垫，帮助你了解我后面要说的事儿。我们现在要说的这个人，是在多个严肃的公开场合说要来Amazon工作就应该付他钱才对的人。当有人跟他意见不同时，他会递出写有他名字的黄色即时贴以提醒那个人“谁是公司的老大”。这家伙是.....，Steve Jobs，我猜。除了没有品味和设计能力。千万别误解我，Bezos是个绝顶聪明的人，只不过他把那些正常的管控搞得像嗑了药的嬉皮士一样罢了。

所以，有一天，Jeff Bezos下了一份命令。当然，他总是这么干，这些命令对人们来说就像用橡皮槌敲击蚂蚁一样。这个命令大概是2002年，我想误差应该是在正负1年内 —— 这个命令发布的范围非常地广，设想很大，让人眼珠子鼓出来的那种，这种惊讶程度和其他的命令相比，就好像突然收到奖金一样。

这份大命令大概有如下几个要点：（陈皓注：这里是本文章的要点！如果这真是Bezos发出来的，那么太赞了，Bezos完全就是一个系统架构大师啊，那可是2002年左右啊。作者调侃Bezos完全是正话反说啊）

- 1) 所有团队的程序模块都要以透过Service Interface 方式将其数据与功能开放出来。（陈皓注：Service Interface也就是Web Service）
- 2) 团队间的程序模块的信息通信，都要透过这些接口。
- 3) 除此之外没有其它的通信方式。其他形式一概不允许：不能使用直接链结程序、不能直接读取其他团队的数据库、不能使用共享内存模式、不能使用别人模块的后门、等等，等等，唯一允许的通信方式只能是能过call Service Interface。
- 4) 任何技术都可以使用。比如：HTTP、Corba、Pubsub、自定义的网络协议、等等，都可以，Bezos不管这些。（陈皓注：Bezos不是微控经理吗？呵呵。）
- 5) 所有的Service Interface，毫无例外，都必须从骨子里到表面都要设计成能对外界开放的。也就是说，团队必须做好规划与设计，以便把接口开放给全世界的程序员，没有例外。
- 6) 不这样的做的人会被炒鱿鱼。
- 7) 谢谢，祝你有个愉快的一天！

哈哈！你们这群150位前Amazon员工，当然能马上看出第7点是我开玩笑加上的，因为Bezos绝不会关心你的每一天。

不过第6点是很真实的，于是，所以人们都去工作。Bezos并派出了几位首席牛头犬来监督并确保进度，领头的是和熊一样大的牛头犬：Rick Dalzell，Rick是以前是陆军突击队队员，西点军校毕业生，拳击手，和沃尔玛的首席虐刑官 / CIO，而且他也是个高大、和蔼、令人敬畏的人，还是经常使用”hardened interface”词的人，Rick 本来的走路和说话都比较hardened interface，所以不用多说，每个人都得干 出有**重大的**进展，这样Rick才能看得见。

在接下来的几年，Amazon内部转变成面向服务架构SOA(Service–Oriented Architecture)，在这华丽转身的过程中，他们学到了相当巨大的东西。我在那个时候，世界上就有很多很多的关于SOA的学术文档，但在 Amazon的那种超大规模的面前，这些东西就好像告诉印第安纳琼斯（陈皓注：电影奇宝奇兵男主角）过马路前要先看看两旁有无来车一样没用，Amazon 的研发工程师们在这个过程中有了很多很多的发现。下面只是他们发现中的苍海一粟：

- pager escalation（陈皓注：生产线上问题的寻呼系统）变得比较困难，因为ticket可能会转过来转过去（陈皓注：ticket就是处理问题的工 单），只到转了20次，都找到真正能解决问题的团队和人。如果每一个呼叫都花去团队的15分钟的响应时间，那在找到真正的团队之前几小时就过去了，除非， 你建造出很多很多的脚手架，测量标准和报告。
- 每一个和你的相关团队突然间都可能成为一个潜在性的DOS攻击者。没人可以让事情有进展，直到在每一个Service里放上配额（quota）与节流阀（throttling）的机制。
- 监控与QA是被统一了。如果你不进行一个大规模的SOA，你就不会这么去想。但是，等到你的Service说，“是的，我还好！”，情况可能是， 服务器里唯一能正常运作的功能就就是一个快乐的机器声音在呼叫你：“我很好，收到，收到”。为了要确认整个服务能正常运作，你需要对每一个部分都去 Call一下。这个问题会以递归的形式地出现，直到你的监控系统能够全面性地系统地检查所有的Services和数据，此时，监控系统就跟自动化测试QA 没什么两样了，所以两者完美的统一了。
- 如果你有上百个Services，而且你的程序只能通过由这些Services来跟其他团队的程序做沟通，那么，没有一套Service发现机制 的话，你就不能找到这些Service。所以，你得先有一套Service的注册机制，这也是一个Service。所以，Amazon有一套全体适用的Service注册机制，以例可以通过反射机制来找到Service，并知道Service的API，以及是否可用，在哪儿。
- 调试其他人的代码以调查问题变得非常的难，几乎都不可能，除非有一套全面性的标准的方式，他可以在可被调试的沙盒里运行所有的Services。

了。对于把Service外部化甚至还有很多人几乎没有入云云的非常生僻的知识，当然，也不会有你怨家的那么多。把业务组织成Service让团队学会了不 能相信对方，就如同他们不能信任公司以外的程序员一样。

当我在2005年中期离开Amazon加入Google时，这个努力进化的过程还在进行时中，但那时已经相当的先进了。从Bezos颁布法令的时间到我离开的时候，Amazon已经把文化转变成了“一切Service第一”为系统架构的公司，今天，这已经成为他们进行所有设计时的基础，包括那些绝不 会被外界所知的仅在内部使用的功能。

那时，如果没有被解雇的的恐惧他们一定不会去做。我是说，他们仍然怕被解雇，这基本上是那儿每天的生活，为那恐怖的海盗头子Bezos工作。不过， 他们这么做的确是因为他们已经相信Service这就是正确的方向。他们对于SOA的优点和缺点没有疑问，某些缺点还很大。但总的来说，这是正确的，因 为，SOA驱动出来的设计会产生出平台（Platform）。

是的，这就是Bezos的法令要达成的目标。他以前（现在也是）一点不关心各团队是否好，也不关心他们使用什么样的技术，实际也不去管他们如果动作 他们的业务，除非团队开始把事搞砸。但是，Bezos比绝大多数的亚马逊人都很早很早就领悟到，Amazon必须成为一个平台。

如果你，你会想到要把一个在线卖书的网站设计成为一个有扩展性，可程序化的平台？你真的会这样想吗？

嗯，第一件Bezos领悟到的大事是，为了销售书籍和各种商品需要的基础架构，这个基础架构可以被转变成成为绝佳计算平台（Computing Platform）。所以，现在他们有了Amazon Elastic Compute Cloud（亚马逊弹性运算云平台EC2），Amazon Elastic MapReduce，Amazon Relational Database Service（亚马逊关系数据库服务），以及其他可到AWS aws.amazon.com查得到的一堆Service。这些服务是某些相当成功的公司的后台架构，比如 我个人喜欢的 reddit 是这一堆成功公司的其中一个。

另一大领悟是，他知道他们不可能永远都创造出对的东西。我认为，当Larry Tesler说他妈妈完全搞不懂怎么使用那个该死的网站时，Bezos的某根筋被触动了，当然，我也也不清楚到底是谁家母亲，这无关紧要，因为没有人的母 亲能够会用那个该死的网站。事实上，连我这个在那那工作超过5年的人都觉得Amazon网站的接口令人胆战惊心。

我并不是很确定Bezos是如何领悟到的——领悟到他不能创造 出一个产品能适用于所有的人。不过，怎么来的这不重要，重要的是他的确领悟了。这种事有一个正式的术语，叫Accessibility，这是计算机世界中最最重要的事情了。

最！重！要！的！事！

如果你在心里面在想“哼？你是说，像盲人和聋人那种Accessibility吗？”，那么，你不是唯一这样想的人，因为我已经知道有很多很多像你这样的人：这种东西对你们这种人来说是不可能 有正确的Accessibility，所以这事你还不能理解。当然，不能理解也不是你的错，就像眼盲，耳 聋，或是其他行动不便的残疾人，这些也不是他们的错。当Software——或ideal-ware——如果因为某些原因不能被存取或使用，那么，这就是 软件或是那想法的错了。这就是Accessibility failure。

就如同生命中那些重大的事一样， 有一个邪恶的双胞胎姊妹，它在幼年都受到父母的溺爱，现在它已经成长为同等强大的复仇女神（是的，Accessibility有不只一个复仇女神），这个复仇女神叫安全性（Security），他们在一些总是争执不休，冤家一对。

不过，我会和你争论Accessibility要比安全性来的重要多了，因为零Accessibility就意为着你根本没有做出产品来，而如果安全性为零，你仍然还是可以有一个某个程度上成功的产品，譬如说Playstation Network。

对了，也许你还没注意到，我其实可以为这篇文章写出一整本书，很厚的一本，其中填满了那家我曾工作过的公司里关于蚂蚁与橡皮槌的事。但是，我可能就永远无法发表这短篇的夸夸其谈了，而你也就无法读到除非我现在开始结尾。

第三部分

那三件Amazon比Google强的中的最后一件事是，Google很不会做平台（Platform）。我们就不懂什么是平台。我们就根本不知道 平台的内涵。你们其中一些人明白，但是你们是少数派。在Google过去这六年来，越清楚这一点就越让我痛苦。我曾有一线希望，来自Microsoft和 Amazon，以及近来Facebook的竞争压力，会让我们全体人都清醒过来，并开始打造我们公司的Service。不是那种特制的或半生不熟的，而是 多少和Amazon的类似的那种：一次到位，真正的，没有作弊或是欺骗，并且把它放在最高优先级的位置。

但实际上却不是，这个事被放在了好像是第10还是第11位，或是第15位，我不知道，反正是相当低。只有少数几个团队严肃地看待这个事，但大多数的团队不是从没有思考过这个事，就是只有一很少的人很鼠目寸光地在看待这个事。

对大多数的团队来说，只要是让他们以提供给别人那种可程序化的方式存取他们的数据与运算的方式来开发软件，就算几个小小的粗糙的Service，对 他们来说也是翻天覆地。他们大部分人都以为他们在做产品，但他们只是在提供那些凄惨粗糙的Service。回去看看前面我所列的那些部分的Amazon学 到的东西，然后告诉我，哪一个粗糙的Service能让你有超凡脱俗。迄今为止，就我所知，一个也没有。就算是这些粗糙的东西很不错，不过这就好像要汽车 的时候，你却只有汽车的零件。

没有平台的产品是没用的，再精确一点，去平台化的产品总是被平台化的产品所取代。

Google+是我们完全失败的不懂Platform最明显的例子，从最高层的管理层（嗨，Larry、Sergey、Eric、Vic，你们好）一直到最最底层的员工（嘿，你）都没不懂。我们全部统统都不懂。平台Platform的黄金守则是Eat Your Own Dogfood（吃你自己的狗食——自己都要用自己的平台）。Google+这个平台是个怀具的马后炮。我们在在发布它的时候完全没有任何API。我查了 一下，目前也只有少得可怜的API。Google+的一个团队成员在发布API时告诉我这个事，我问：“这是Stalker API（用来偷窥的API）吗？”，她郁闷地说“是啊”。我的意思是，我那是是个玩笑话，但是，不，我们提供的唯一的API就是取得某人的信息流，所以， 我想我把玩笑开到自己头上了。

Microsoft知道“狗食守则”至少有20年了。这已经成为他们世世代代文化的一部分了。不能是你吃人类的食物而给你的开发人员们号狗食。那样做只会是为了短期的成功而掠夺了平台长期价值。平台就是要你考虑得长远。

Google+就像膝跳反射，一种短视的的东西，是基于以为Facebook其伟大产品的成功作出的错误判断。但那不是为什么他们能成功的东 西。Facebook的成功是因为他们建立了一个可以让外界在其上上面开发的产品群。所以对Facebook对每个人来都不一样。有些人把全部时间花在“Mafia Wars”上，有些人则是花在“Farmville”（开心农场）。那里还有成百上千个不同的高质量的时间消耗游戏，所以，人们总是可以在那里找到他们想 要的。

有多少个靠谱了呀？问题在于我们试图预测人们想要什么，然后推出产品给他们。

你不能这么做。真的不能。也不可靠。在这个世上，甚至在整个计算机的历史上，只有极少数几个人能够这么干，Steve Jobs是其中一个。但是我们没有Steve Jobs。对不起，我们真的没有。

Larry Tesler有可能说服了Bezos相信他并不是Steve Jobs，但Bezos意识到他不需要成为Steve Jobs也能提供给所有人好的产品：大家容易使用的接口与工作流。Bezos明白他只要有让第三方开发人员来做的平台，这些东西自然就会有的。

我要向一些人道歉，这些人会觉得我所说的是再明显不过的了。是的，的确是巨明显的。只是我们没有去做。我们没有领会平台，我们也无法领会到 Accessibility。这两者本来就是同一件事，因为平台会解决Accessibility。而平台就是Accessibility。

- 是的，Microsoft领会到了。而且你们也像我一样知道Microsoft他们对这些东西一知半解。那是因为他们能够了解平台完全是他们商业上意外性的副产品，是他们一开始的业务就是提供平台。所以他们在这个领域有着三十多年的经验。如果你去看看 [msdn.com](#)，并多花点时间浏览一下，假设你以前从没去看过，你等着被吓到吧，因为那里面的东西可是多得不能再多。他们拥有成千成千成千个API。他们拥有一个超巨大的平台。说实话，太巨大了，因为他们要霸占一切，但至少他们做了。
- Amazon也领会了到了。Amazon的AWS([aws.amazon.com](#))相当的惊人。去看看吧，四处点一下。令人羞耻吧。我们今天什么都还没有。
- 很明显Apple也领会到了。他们做些在基础上不开放的选择，具体来说是移动平台。但是他们明白什么是Accessibility，并且他们知道 如何燃起第三方开发团体的力量，而且他们吃自己的狗食。你知道吗？他们的狗食做得很好吃啊。他们的APIs比Microsoft的要干净不知道多少倍，而 且是远古的时候就这样了。
- Facebook也领会到了。这正是让我所担心的。这使得我不得我抬起懒惰屁股写下这些东西。我恨写Blog。我恨.....Plus（指Google Plus）不管怎么称呼它，反正在Google+上发表长篇大论，就算这是个糟糕的地方，但是你还是希望Google能成功.我真希望！我的意思 是，Facebook想挖我，而且很容易就去了。但Google是我的家，所以我坚持我这个小小的家庭干涉，就算你不舒服。

等到你为Microsoft与Amazon提供的平台感到神奇后，当然，我想你也可能会被Facebook吓到（我不敢去看，因为我不想让我太沮丧），让我们回头看看 [developers.google.com](#) 。是不是有很大的差别？我们的这个平台看起来像是你家小学五年级的侄子搞出来的东西一样——让一个小学五年级的他，试着描述一个强大的的平台公司，如果这家公司什么都有，会整出个什么东西来？

这里请不要误解我——我知道一个事实，dev-rel 团队为了发布这些API曾经不得不去“搏斗”。据我所知，这个团队很不错，因为他们知道什么是平台，并且他们如英雄般努力挣扎地要做出来，然而遇到的却是“平台冷漠”的环境，难听点是那种有敌意的环境。

我只是在直白地描述出一下 [developers.google.com](#) 在外人眼里是什么样子。它看起来很幼稚。Maps APIs在哪呢，老天爷啊？其中有有些东西还是实验性的项目，我点进去看的APIs.....他们都毫无价值。他们很明显都是些狗食。甚至都称不上是好的有机食品。跟我们内部APIs比起来，他们全部简直就是猪屎马粪。

当然，也不要错误地理解我对Google+的看法。他们还不算是最差的。这是文化氛围的事。我们现在做做的简单来说就是要进行一场战争，是一场失败很多的少数的平台派和那些强大的信心坚持的产品派的战争。

那些从头到尾明白理解供外部可程序化的平台概念的团队都是受压迫的人——Maps跟Docs团队浮现在我脑海中，而且我也知道GMail是这个方向 的先头部队，但是他们很难得到资金注入，因为这不是我们文化的一部分。Maestro的资金完全没发和Microsoft Office开发平台的资金相比：就像小毛兔和暴龙相比一样。Docs团队知道自己永远无法和Office竞争，除非他们能赶上Office的脚本能力，而且他们得不到他们相要的资源。我的意思是我假定他们没有，现在应用的脚只在电子表格中有，而且没有为API设置键盘快捷键。在我看来，这个团队完全没有 被重视。

具有讽刺意的是，Wave是个伟大的平台，愿他能安静地长眠。我们需要知道，做一个平台并不会马上给带来成功。平台需要杀手级应用。Facebook——他们供应了的涂鸦墙和朋友等其他东西——则是Facebook平台的杀手级应用。但是，如果你说没有Facebook平台，仅有 Facebook应用也能像今天这样成功，那么，这会是个一个非常严重的错误。

你知道吗？人们总是在说Google的傲慢自大。我是个Google人，所以我和你一样当听到那些话都会觉得很愤怒。但总体而言，我们并不傲慢。我们大约99%不自大。我在文章开头时就写到——如果你回去看看——我是这样描述Google的“所有的事都做对了”。我们知道人们为什么要这么说我们自大，因为我们没有雇用他们，或是因为他们对我们的政策不爽，或是那一类的事情。他们推断出我们自大是因为这样会让他们心理平衡一些。（陈皓注：作者在这里的反话正说）

但是，当我们摆出那种我们知道怎么给用户设计出完美的产品的姿态时，你最好相信我，我们就是笨蛋。你可以说是自大，天真，或是别的什么，无所谓，但最终的结果就是我们干的很愚蠢。因为，这世界不可能有一个产品对所有人都是完美的。

你看，我们的浏览器居然不能让人设定默认的字号。这就是我们对Accessibility的公然冒犯。我的意思是，我总有一天会老的，我也会得老花眼，并会变瞎的。我的意思是我不会变瞎，但是如果你到了40岁，你的老花眼让你看不清近的东西。那么，字号的选择会成为生和死的问题：某用户就会被完全排 除在产品之外。但是Chrome团队就是这么NB傲慢：他们想要开发出无需配置的产品，他们对此相当自豪，去你TMD是瞎子还聋子，管你是谁，在你剩下的 日子每访问一个页面都按一下Ctrl-+吧。

并不仅是他们。是第一个。问题是，我们是一家“产品”公司，一直一直都是。我们开发的最成功最有吸引力的产品——搜索引擎，那样巨大的成功让我们产生了很多定式和偏见。

Amazon过去也是家产品公司，一道神秘的力量使得Bezos领悟到他们需要平台。那道神秘力量来源于，他们被 逐渐蒸发的市值逼到墙角了，不得不想方设法突围出来。但他当时所拥有的只有一群工程师和他们的一堆计算机.....除非他们能变成印钞机.....你可以看到他们是怎 么搞出来AWS的，而不是像我们Google+一样事后诸葛亮。

Microsoft从一开始就是个平台，所以他们有很多很多的实践。

Facebook：我有些没看透。我不是专家，不过我很肯定他们一开始也是一个产品，并且成功了很长时间。所以我不知道他们什么时候开始转变成为平 台的。应该是很久以前的事了，因为他们要成为平台后，Mafia Wars这玩意才会出现（而Mafia Wars也很老了）。

我们面对的问题非常的庞大，因为我们需要经过剧烈的文化转变后，我们才能迎头赶上。我们没有内部的SOA平台，所以我们外部也没有。这就是说，我们 整个公司都“没有领会到”：产品经理没有，工程师没有，产品团队没有，没人领会到。就算是个别人有，比如你你有，那也相当于没有，除非我们在生死存亡的时 候。我们不能这样不断推出产品，并装作我们以后会把这些产品转变成迷人美丽的可扩展式的平台。我们试过了，不行。

平台的黄金守则，“Eat Your Own Dogfood 吃自己的狗食”，换句话说，“先打造出自己使用平台，然后把它用在所有的地方”。你不能事后再做，那样做就太困难了——你去问问那些把 MS Office平台化、把Amazon平台化的人。如果你放在后面做，那么你比一开始要花十倍的精力才能做对。你不能作弊，你不能让内部软件走秘道去取得特 定的优先权限，不为什么，你必需从一开始就要解决这个问题。

我不是说现在做已经太迟了，但我们等的越长，我们就会越接近——“太迟了”。

老实说，我不知道这篇文章怎么收尾。我今天在这里说得太多了。因为这篇文章花了我6年时间。请包涵我言语冒犯之处，包涵我可能误解了一些产品，团队，或某个人。也许我们真的在开始做了很多平台方面的东西，只是我没看到。我只想说声对不起。

但是，我们必始在开始时把事做对！

 赞

 收藏

 [1 评论](#)



相关文章

- [最终一轮面试被 Google 刷掉，这是一种什么样的体验？](#)
- [开源巨献：2017 年 Google 开源了这些超赞的项目 · !\[\]\(e27c4336460e9e6729a19580c0456728_img.jpg\) 1](#)
- [谷歌用两年时间研究了 180 个团队，发现高效团队有这五个特征 · !\[\]\(1a140e8db538fd46d58af9f9540232fd_img.jpg\) 7](#)
- [谷歌大牛说：为什么 Kotlin 比你们用的那些垃圾语言都好 · !\[\]\(5a658b86f2c8900a276c586c1f8f9f2f_img.jpg\) 7](#)
- [谷歌人工智能背后的大脑](#)

可能感兴趣的话题


- [各位觉得大公司的规定适合小公司吗？ · !\[\]\(b6d55d0b173caf9b2505126db01e6158_img.jpg\) 2](#)
- [Vuescroll – 一个基于Vue的虚拟滚动条](#)
- [现在报个班可以帮助自己吗？ · !\[\]\(12811766810e4126d2bed4d8c0808e60_img.jpg\) 15](#)
- [国外的程序猿可以工作到退休而国内的为什么这么短命（思维认知） · !\[\]\(ef4c06c861a77cbd8cff5c2a4ca34233_img.jpg\) 4](#)
- [Java非要用繁杂的框架吗？JSP+serverlet有什么优缺点 · !\[\]\(80b05c8a80151a7cedd31bb12aa6add6_img.jpg\) 9](#)
- [自然语言处理（NLP）如何入门？](#)


登录后评论

新用户注册

直接登录     




最新评论



人见人爱的土豆 （ 7）

2016/05/10

五年后再看这篇，感觉每句话都应验了。

 赞  回复 



- [本周热门文章](#)
- [本月热门文章](#)
- [热门标签](#)

0 [为什么码农要了解业务？](#)

- 2 [九年程序人生](#)
- 3 [Linux 权限控制的基本原理](#)
- 4 [RabbitMQ 发布订阅实战：实现延...](#)
- 5 [2018 年 Java 程序员必读的十本书](#)
- 6 [Vim-plug：极简 Vim 插件管理器](#)



业界热点资讯

更多 »



[地址 1.1.1.1, Cloudflare 推新公共 DNS 服务](#)

04/02 · 29 · 5



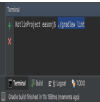
[安卓用 Java 侵犯甲骨文版权，谷歌或赔 88 亿美元](#)

03/28 · 34 · 1



[李文星家属诉 BOSS直聘：哪怕赔一分 能给个交代也值](#)

03/27 · 45 · 3



[Android Studio 3.1 正式发布，默认使用 D8 Dex...](#)

03/27 · 22



[GitLab 发布全球开发者报告：开源仍是主流](#)

03/25 · 18



精选工具资源

更多资源 »



[mlpack: 一个C++机器学习库](#)

[C++, 机器学习](#)



[Whitewidow: SQL 漏洞自动扫描工具](#)

[数据库](#) · 4

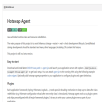


机器学习 · 工具



静态代码分析工具清单：公司篇

静态代码分析



HotswapAgent：支持无限次重定义运行时类与资源

开发流程增强工具

关于伯乐在线博客

在这个信息爆炸的时代，人们已然被大量、快速并且简短的信息所包围。然而，我们相信：过多“快餐”式的阅读只会令人“虚胖”，缺乏实质的内涵。伯乐在线内容团队正试图以我们微薄的力量，把优秀的原创文章和译文分享给读者，为“快餐”添加一些“营养”元素。

快速链接

[网站使用指南](#) >>

[问题反馈与求助](#) >>

[加入我们](#) >>

[网站积分规则](#) >>

[网站声望规则](#) >>

关注我们

新浪微博：[@伯乐在线官方微博](#)

RSS：[订阅地址](#)

推荐微信号



合作联系

Email：[bd@Jobbole.com](mailto:bd@jobbole.com)

QQ： 2302462408 （加好友请注明来意）

更多频道

[小组](#) — 好的话题、有启发的回复、值得信赖的圈子

[头条](#) — 分享和发现有价值的内容与观点

[相亲](#) — 为IT单身男女服务的征婚传播平台

[资源](#) — 优秀的工具资源导航

[翻译](#) — 翻译传播优秀的外文文章

[文章](#) — 国内外的精选文章

[设计](#) — UI,网页，交互和用户体验

[iOS](#) — 专注iOS技术分享

[安卓](#) — 专注Android技术分享

[前端](#) — JavaScript, HTML5, CSS

[Java](#) — 专注Java技术分享

[Python](#) — 专注Python技术分享

