万能 Java

PUBLISHED FEBRUARY 26TH 2015 BY QINGNIU

我常常问面试者，“你最喜欢的编程语言是什么？”答案几乎如出一辙，“工作中我只选择正确的编程语言。”废话，谁会故意选择错误的语言呢？这显然是为了逃避选择一种具体的编程语言，以免选择了一种我不喜欢的。

如果面试者这样回答“我最熟悉某一种编程语言”，这同样也没有回答我的问题。

当时要是我的话，我会这样回答，“我最喜欢 Python，因为使用它编程让我感到快乐，但我只在某某情况下使用它。其余时间，我使用 XYZ...”

然而，大约一年之前，我产生了一个奇怪的想法：Java 适合所有的编程工作。（在你吐槽之前，我暂停一下，）这个想法根植于你感觉正确但却与现实不符的一些观点，而且这个想法从来都没有流行起来，但不管怎样，请让我先来解释一下。

Python 的确是我喜爱的编程语言，用它编程真的让我感到快乐。它让我的大脑感到快乐，它和伪代码是如此契合，以至于用它来工作能让人真正感到愉悦。

多年以前，我曾经读过 Bruce Eckel 一本颇具影响力的书《强类型与强测试》（*Strong Type vs. Strong Testing*）。在书中，他声称静态类型（他称为强类型）是保证程序正确性的多种方式之一，如果你用单元测试去检查其它方面（例如算法和逻辑），那么类型也将得到检查，因此你不妨采用动态类型编程语言，并从中获得动态类型编程语言的优势。

Bruce 使用了 Python 来说明他的代码，其成效非常显著：我决定从今往后写什么都用 Python。不幸的是，工作中一个大型 Java 项目进展到中途时，我和同事一致认为这个程序应该用 Python 来写，也许有一天，我们会找到一个很好的借口来重写这个程序。

不到一年时间，几件事情让我的想法来了一个180度的大转弯：

- 在一家公司里，我写了一个模拟器，这样就可以让我的 Java 服务独立运行而无需一个全功能的网站。在这个模拟器中，我运行一些脚本测试包括失败在内不同的情景。由于 Java 6 已经内置了 JavaScript 的解析引擎以及很多人都了解这门脚本语言的缘故，我决定使用 JavaScript 来编写这些脚本。我认为使用脚本语言可以让我们和测试人员很容易地编写测试。一位名叫 Justin Lebar 的实习生认为我们应该只使用 Java。模拟器是用 Java 写的，测试脚本为什么不用 Java 呢？它已经在那里了，这是大家都知道的。我们仍旧坚持使用 JavaScript。在整个过程中，我不得不写各种各样的的代码让 Java 和 JavaScript 相互沟通。你知道，因为无法定位到具体脚本执行的所在行数，这意味着不同语言堆栈的足迹已经变得难以跟踪了。测试人员无法完成任何测试。到最后，我们从 JavaScript 中一无所获，Justin 的观点无疑是正确的。

- 还是在那家公司里，我们使用 JSON 格式（顺便说一下，这是个很好的想法）来存储我们的日志文件，一位同事写了一个名为 logcat 的 Python 程序，用来解析日志文件和输出标准的柱状图报告，这个程序有许多不错的功能特性（包括一个二分查询时间戳）。在我的一个私人项目中，也需要类似的东西，我再次建议使用 Python。但我的搭档 Dan Collens 则认为应该使用 Java，因为它已经在那里了，我们都了解它，而且它够快。他最终用 Java 实现了它，我的搭档是对的：它的速度极快。我对 Python logcat 和 logcat 的另一个 Java 实现作了一番对比，后者的速度大约快十倍。 使用 Python 去实现，无论程序员节省了多少时间都无法挽回失去用户的损失，因为很多用户

每次不得不等上十倍的时间才可以拿到分析日志报告。

- 最后一个例子，我编写了一个简单的程序用于搭建一个 Web 界面。我觉得应该使用 Python，但是这样做的话，我需要找出如何利用 Python 的类库来为 Web 页面提供服务的办法。我在此之前已经在 Java（采用 Jetty）中实现过这个功能了，所以使用 Java 的话，我可以在更短的时间内将其做好并且运行起来。这个时候，我开始意识到，随着我在第三方 Java 库上面的知识积累以及在实用工具方面的不断成长，使用其它语言的成本已经变得越来越高了。我需要把这些事情搞清楚再写一遍，而不是从已有的项目中复制和粘贴。注意，这不仅适用于 Java，它也适用于任何使用单一语言的场景。

对于 Java，最大的争论在于它的语法繁琐。可能吧，但那又怎样呢？我认为真正的争论是写 Java 代码需要更长的时间。我怀疑在最初的10分钟之后这是很真实的。当然你不得不写 public static void main，但那又能花多少时间呢？当然你也必须这样写：

```
Map<String,User> userIdMap = new HashMap<String,User>();
```

而不是：

```
userIdMap = {}
```

如果从一个更大的情景来看，这算长吗？一天之中能多花多少分钟，2天呢？而在 Python 更加接近实际的代码看起来是这样：

```
# Map from user ID to User object.
userIdMap = {}
```

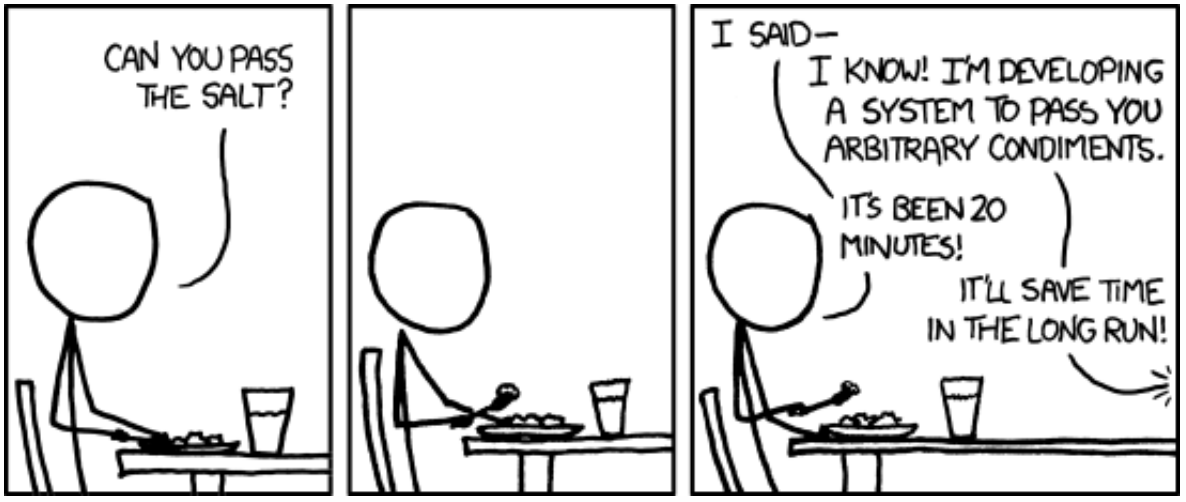
（如果没按照上述方式编写 Python 代码，那么你就可能面临更大的问题。缺乏文档的 Python 程序是很难维护的。）我们面临的问题是，程序员认为不用动脑筋的工作是痛苦且浪费时间。我在这里引用某个论坛中的一条帖子，很好地证明了这一点：

> 当你不得不为极其明显的对象增加类型声明时，没有比这更无聊的了，比如 Foo x = new Foo()。– @pazsxn

实际上不是这样，额外键入 Foo 并非“很无聊”。只是三个字符而已。由于这项工作无需动脑，致使其副作用被严重夸大了，但这只是小事一桩。程序员真正不愿意做的事，是为了处理列表事项而编写某种类型的命令：

```
if command = "up":
     up()
elif command = "status":
     status()
elif command = "revert":
     revert()
...
```

因此，他们将会编写一些自认为聪明的自动调度代码，但这样做需要花费更长的时间，而且肯定会让后来的程序员产生迷惑：revert() 方法是怎样调用的呢。即便如此，程序员却错误地觉得好像这样做会是在节约时间。这其实是一个动态语言的陷阱。它让你自我感觉更有效率，但除了编写一个新程序的前10分钟之外，其他时间并非如此。你只是通过手工编写了一些愚蠢的调度代码，到最后，你还得在那些真正的工作上花费精力。



（题外话：关于使用 vim 编辑器编写代码这一问题，我对不同选择的理解有误。我感觉使用 vim 非常有效率，似乎代码在终端上飞舞，而使用 Eclipse 让我感觉迟钝，这证明了我的选择更倾向于效率。但显然我的收益是建立在一定损失的基础之上的，

我不得不查看谁调用了一个特定函数，或者不得不手动查看一个对象的方法。动态语言的支持者们有他们自己的选择，我承认在这一问题上我同样是错的。）

那么到底为什么要选择动态语言呢？如果你和我比赛去写一个简单的博客系统，你用 Python 的话，使用 pickling 和 whatnot 你在30分钟内就会让事情变得有趣，而我用 MySQL 构建的话要花2天时间。许多语言方面的选择是基于类似这样的微不足道的竞赛。但在大约两周开发之后，当我们都需要增加一个功能时，我花的时间最多和你一样，而且我不需要在如何让我的系统应对大量用户上花费任何时间，或者追踪那些令人困惑的无效语句，其原因只是你的一个函数名拼写错误导致语句执行中断，或者想弄明白这个请求参数到底包含什么东西，悲催呀。

> 黑客级程序员蔑视“规范和要求严格的语言”是目光短浅的表现。大型、长期使用且由多名程序员参与的项目和自用的快速原型项目是完全不同的东西。– Source

而且，你觉得你可能不会很快碰到程序的可伸缩性问题？NaNoWriMo 网站在每年的10月31号都会当机，停止响应好多天。在这一期间的几个小时内，大约有60,000用户访问，每秒可能有4个请求。这个网站是用 PHP 写的。OurGroceries 的后端是用 Java 写的，它能处理（当前）大约每秒50个复杂请求，而对于 Java 线程来说 CPU 很少超过1%。

Twitter 最近通过把搜索引擎从 Ruby 切换到 Java，将查询速度提升了三倍。

一年之前，Joel Spolsky 发表推文：

> Digg: 200MM 页面浏览，500台服务器。Stack Overflow: 60MM 页面浏览，5台服务器。我漏掉什么了吗？
>
> — Joel Spolsky (@spolsky) October 13, 2010

@GregB 的反馈是：

> @spolsky: Digg: 200MM 页面浏览，500台服务器。Stack Overflow: 60MM 页面浏览，5台服务器。我漏掉什么了吗？<< 这就是 PHP 的原因。
>
> — Greg Brant (@GregB) October 13, 2010

StackOverflow 使用的是 ASP.NET。因此你会整天抱怨 public static void main，但需要搭建500台服务器这一事实也太可笑了。动态语言的缺点是真实存在的，代价也很昂贵，而且会一直延续下去。

那么单元测试这个观点又怎么样呢？如果你必须对你的代码进行单元测试，静态类型能为你带来什么呢？好吧，它能加快速度，而且是大幅度的。但是编写和维护单元测试也需要耗费时间。最重要的是，最常出现的 bug 并非单元测试都能完全覆盖。除了少数例外情况（例如解析器），单元测试是在浪费时间。引用我哥们的一段话，“单元测试是一种冗长且易于出错的方法，它试图挽回由于缺乏静态类型注解而失去的价值，但却以一种笨手笨脚的形式出现，因为它和实际业务代码本身是完全分离的。”

因此，我的新思路就是：做任何事都用 Java。不要试图使用 Python 写一些可以快速实现的黑客代码，因为：

- 你无法从使用主要编程语言开发的项目中复制和黏贴代码。//实现代码重用
- 开发起来可能感觉会快一些，但这是假象。实际节省的时间非常有限，尽管有些语法特征的确让人讨厌。
- 我和我的同事不得不学习和掌握另一门语言、平台以及一系列类库。
- 还有一个重要原因：很可能，这个快速实现的黑客代码将会成长为一个重要的工具，我没有时间去重写它，因而每次使用它我都要忍受由于性能不佳和难于维护而导致的惩罚。

使用 Python 开发是快乐的，我同意这个观点。我热爱 Python。当我写一个数独求解程序时，我会使用 Python。但是对于更大些的项目，它是个错误的工具，而且对于和支付有关的任何规模的项目来说，它也是个错误的工具，因为这可能会给你的老板带来一定损失。

我甚至想把这个思路发挥到极致 - 使用 Java 编写 shell 脚本。除了一个简单的包装器之外，我发现 shell 脚本最终都会发展到一种情景，即仅仅为了从 bash 中的一个数组中移除一些中间元素，需要我在晦涩难懂的语法中反复寻找方法。这是多么蹩脚的语言啊！错误的工具呀！还是使用 Java 吧。如果你觉得在 shell 上运行命令显得很愚蠢，编写一个工具函数就可以解决这个问题。

我已经编写了一个 Java 启动器脚本，这样我就可以将其写在 Java 程序的头部：

```
#!/usr/bin/env java_launcher
# vim:ft=java
# lib:/home/lk/lib/teamten.jar
```

我可以让这个 Java 程序执行，并且同时去掉 .java 扩展名。脚本提取头部内容，编译并缓存 class 文件，然后使用指定的 jar 包去运行。这原本是 Python 的特有优势：对于简单的一次性程序，就无需构建脚本啦。

专注于单一的语言将会产生一个有趣的影响：激励我完善我的个人工具函数库(上面的 teamten.jar)，因为我的努力不再需要分散到几个语言之中了。举个例子，我写了一个图片处理的类库。和你在 Java 和 Python 中能找到的任何类库相比，这个类库不仅速度快而且质量更高。这花费了我的一些时间，但我认为这是值得的，因为我发现，我无法使用 Python 脚本更好地更好地实现裁剪一张图片的功能。我现在可以充满自信地把对 Java 的投资作为我未来职业和个人技术的一个重要组成部分。

最后还有一个在众多编译型静态类型语言中，我为什么特别选择 Java 的问题。C 和 C++ 的优势（轻微的性能优势，可嵌入性，适合编写图形化库）不适用于我的工作。C# 挺不错，但不是跨平台的。Scala 太复杂了。其他语言像 D 和 Go 都太新了，因此我不能把工作赌在它们上面。

每当我告诉人们我现在写什么都用 Java 时，他们看起来都很恐惧的样子。甚至有一位朋友明显面带厌恶的表情。但是你知道吗，Java 是一门相当好的语言，当我进行代码编译时，往往在第一时间，它通常会正确地运行。任何其它语言都没有像 Java 那样给予我心灵上的宁静。Java 就像一匹兢兢业业的宝马良驹，对于各种类型的应用程序来说，它都适用。

作者：*Lawrence Kesteloot*，是一名自由程序员，住在旧金山，喜欢林迪舞、法语、旅行、以及结对编程。

原文： *Java for Everything*

感谢： *Jodoo* 帮助审阅并完成校对。