别学框架，学架构 - 文章 - 伯乐在线

前段时间，我有过一次非常有趣的谈话。有个同事站出来支持Angular，他说Angular加快了Web开发的速度。我已经开发复杂的web服务超过10年了，曾经在Microsoft工作，也曾在Cyprus为Spotware工作。目前，我为硅谷的一个初创公司编写应用程序。总的来说，我会顺应潮流。但我感觉自己像只恐龙，因为在我看来使用前端框架没有什么意义，但它被证明是主流。在2014年，我投入到Angular、Knockout和Backbone的世界中。如果你想弄清楚我从中得到了什么，为什么停止使用它们并且推荐你们也这样做，那么欢迎继续往下看。

我们都知道Angular有很多问题，调试是其中一个主要的问题。当发生一个没有文档化的错误时，只有stackoverflow能够救我们。但是我们也应该找出发生了什么，以及最重要的找到哪里发生的。Backbone和Knockout也有缺点。然而，很多人在使用它们，因为它们的优点更加重要。说实话，他们没有看到别的选择。但是，我们是有选择的，只是我们忘记它了。

每个模块都应该只执行一个功能，还记得这条古老的原则吗？如果一个模块执行了两个或者更多的功能，我们应该将其拆分为不同的部分。我们可以找到大量的理由来说明为什么这么做以及为什么我们应该坚持这样做。无论如何，所有已有的框架都违背了这条原则。而且，“框架”这种方法本身就违背了这条原则。框架给我们带来一定限制，它让我们遵循最佳实践。最佳实践也在不断发展，对于一个小规模的开发人员团队来说，他们可能不会知道哪些实践更适合一个小页面，一个包含了数据管理的复杂逻辑的管理面板，或者具有高性能需求的多媒体网站。只有当你是一个新手程序员的时候，使用框架可以约束和规范你的代码。我的建议是使用最佳实践，但是不要使用框架。让我来解释为什么是这样。

框架看起来像是一些非常大并且很难重现的东西。但是它只是一些标准模式的集合。例如，Backbone模型中使用了观察者模式，同样在Angular和Knockout数据绑定中也使用了它，这带来了非常好的效果。但观察者模式只是一个很著名的模式，我们可以用30行JavaScript代码来实现它，或者从成千上万的现成的实现中下载一个（顺便说一下，它们除了方法名字，其它都是一样的，因为模式的原则是相同的）。框架的其它组建也是通过类似的方式实现的。在理解这些原则之后，我们在很多时候可以不用写任何代码。例如，当我们在一个小组件中实现MVP时，我们可以在头脑里将这些方法分为控制器，将属性分为模型等等。

从实践中得到的一个示例：我参加一个西班牙公司的工作面试，在那里我必须通过在线编码的形式，在一个小时以内完成一个测试。这个任务是针对文档创建一个单页面应用程序。我使用JavaScript来完成这个任务，其中只使用了libraries模块。我甚至有时间去写一些测试。他们不能理解我是怎么在不使用框架的情况下实现路由的，以及复杂的交互元素和其它很多事情。他们都是在这个行业中混迹了10年以上的资深人士，但是他们只是学习了特定的解决方案，而非原则。

学习框架，你不得不重新学习，学习那些不断出现的新的解决方案，而且你的部分经验会最终变得毫无价值。但是当你学习原则——原则是不会变的。为了创建类，我使用一个五年前编写的类库，观察者模

式的实现一直没有变化。每一个类库都只执行一个功能，并且执行的很好。我从来没有想过像框架那样，将一个组件替换成另外一个。因为观察者就是观察者，它是一个模式，而不是代码。我们可能会根据任务来组合不同的模式，但模式不会改变。另外一个原则是我们可以扩展代码，但不能修改代码。我们可以在网上，或者“四人帮”的书中找到对应的根据。在这个原则下，如果你哥框架或者类库有第二个，第三个或者第十个版本，一些功能会被删除，其它功能会被修改，这会造成一些产品bug。对于修改代码来说，唯一的好原因就是适配新的浏览器，但是公开方法无论在任何情况下都不应该被修改。

编程成为了市场的受害者。它们为我们提供一个“魔法”按钮，承诺可以解决我们所有的问题。但是结果呢，我们逐渐习惯去使用它，但并不能分解复杂的问题，将小麦从麦壳中分离出来。我会去使用框架吗？只有当编写一个将来无需维护的产品时，我才会用框架。但如果要在一个会持续至少一两年的服务中使用框架的话，则完全是一种自杀行为。在此期间，你将会编写更多的代码，比整个框架的代码还要多，并且不止一次面对框架带来的各种限制。你花费在编写各种变通方法的时间，可能会比不使用框架而去实现大量必要组件的时间更多。

你不是在发明轮子。实际上，你确实是在使用类库，但是你是根据真实的场景而非预定义的方式来组合它们。有人可能会说，框架也可以进行扩展。但是如果我想要根据我的API来获取Backbone模型，或者根本不获取它们，会怎样呢？或者我想从本地存储中获取它们？如果我们有复杂的更新逻辑，它依赖时间和标记，而我们应该在获取之后向另外一个服务器发送同一个模型池，又会怎样呢？你永远不会知道。我们应该在这种情况下使用Backbone吗？我们只会使用它的5%的功能，其余的都是各种变通方法和自定义逻辑。与此同时，在理解架构原则后，我们去创建一个解决方案来适应每一个任务，并让它可以应对需求变更，并不是一件很难的事情。

众所周知，程序员在大部分时间并没有在打字，而是在思考，在设计模式方面进行思考有助于提高效率。当我寻找一些有趣的架构解决方案时，我经常会阅读一些新的类库的公开方法。如果它的实现不是很清楚，我会查看代码，但是这里有一条原则，想法是最重要的。例如，我们可以在10分钟之内完成promises，但是它的效果如何，那就是我告诉你不要学习框架而要学习架构的确切的原因。

P.S.:这篇文章就是希望能引起争论。当然，框架有一定的优势，但是它会让人变成“傻瓜”。如果不使用框架，你不能解决问题，并且因此耽搁数天甚至数周，这就是一件很丢人的事情。事实上，如果我们集中注意力在正确的事情上，在架构解决方案上，就会变得非常简单。这就是我试着告诉你的事情。希望这篇文章能给新手带来帮助，也希望这里描述的方法能够让他们将来变成非常酷的程序员。

你曾经在大型的B2C项目中使用过框架吗？

- 是的，我使用过。这些框架的限制要比它的能力更多。
- 是的，我使用过。这些框架的能力要高于它的限制。
- 取决于项目。
- 没有，我没有使用过。

> 打赏支持译者翻译更多好文章，谢谢！

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