一直以来，写日常问题、前沿技术和架构思考类的文章比较多，今天为什么突然来说说Swing这个陈年老技术呢？

因为在CSDN上看到了这样的一篇文章：





可以看到作者对于学Swing还是挺愤怒的，不过确实Swing现在很少用了。但我的观点还不是那么绝对，因为从个人成长来说，Swing的学习确实给到了不少收获。开始也没打算专门来写篇文章说说看法，然后上了一下掘金，居然就又看到了这篇文章（强大的推荐机制）：



开始看标题格式还以为是同一篇，仔细看才知道原来是驳前面那篇的，对于后面那篇的不少观点，个人还是比较认同的。

同时也因为作者的观点，联想到经常有读者跟我聊遇到职业瓶颈的问题，与我给出的建议有很多不谋而合的认识，所以还是想写一篇文章来说说。

首先，我尊重每个人对技术学习的观点，这里不评论到底谁对谁错，我就结合自己的学习和成长路径，谈谈这个问题的看法吧。**如果屏幕前的你是正在读书的学生或是遇到瓶颈的在职者，强烈推荐看完本文！**

Swing对我的最大收获

Swing，这个算是Java领域里非常古老的一个技术点了，主要用来做客户端程序的界面。

记得上大学那会儿，Java课的大作业、暑期实践和后来研究生时期的实践项目都有所涉及。还清晰的记得，当时跟宿舍小伙伴，一起做的是模仿当时MSN的扫雷对战游戏实现的一个支持联网的客户端小游戏，界面部分就是用Swing来实现的。

在这个时期，我觉得最大的收获就是在学完了Java之后，通过Swing可以实现一个看得见的，让自己很有成就感的作品了，这样的学习反馈其实对学习者来说，是非常棒的，也是感受编程乐趣的重要开始。这个时候其实是会有一些骄傲的，因为在实现的过程中，碰到了很多问题，也不断的尝试着去解决了很多问题。对于编程的思维模式是一次很棒的锻炼。

在骄傲之后，随着后续课程的跟进，马上就会迎来一次打脸和进步的课，就是设计模式！这个时候就会发现之前的实现有多糟糕，记得开始最多碰到的问题就是各个Panel之间的操作联动，由于开始没有学过设计模式，不知道观察者模式等高级设计，一股脑的搞了很多静态变量，以实现各种联动，最后导致的就是程序耦合性非常高。

再之后，也是用Swing做了不过不少东西，比如：矢量图的画板，无服务的聊天室。进一步学到了状态模式的应用，网络相关的编程等。

也许有读者会说，那你不会直接学Spring Boot + Vue来搞不是很多也都能实现么？我觉得这两者之间就学习上来说最大的冲突并不是将来会不会用，而是Swing应用的技术是非常纯粹的，只要你学了Java，你就能马上上手。而对于Spring Boot这样已经封装上几层的框架，加上Vue这样的前端框架，已经建立在了太多基础内容之上，可能跟着教程去做，可以完成一些固定的东西，但要去深入理解这些技术的本质和更为重要的编程思维，帮助就没那么大了，而且从编程启蒙、进阶、深入的认知过程来说，初期我并不太建议把技术栈搞的太杂，反而容易觉得乱，不利于学习和培养兴趣。

大学时期我们学的Swing，虽然最后在工作中并没有给到我任何直接的帮助。但在学习的过程中，对于锻炼基础的编程思维作用是非常大的，尤其对于设计模式的应用，作为来学习Spring MVC也好，Vue也好，很多设计思想上都是共同的。通过一个足够单纯的技术内容，对思维模式的理解更容易上手，掌握之后再去类比着学习其他的就会非常快。

大学专业课程对我的最大收获

其实不光Swing，很多大学里的专业课程，在我们未来的职业生涯中往往都不会直接用到。

比如：

* 我们学了操作系统原理，就要去做操作系统吗？
* 我们学了计算机网络，就要去做网络设备吗？
* 我们学了数据库系统原理，就要去做数据库产品吗？

既然我们未来都没有去做这些基础设施的工作，那我们**还学了干嘛？干脆别学了？书扔了算了？**大学期间，我虽然顺利的完成了这些课程的学习，但以前我也有过这些极端的想法，我们将来都去做业务应用了，这些东西有个鬼用！

然而，在不断的实践和挑战中，我发现对于框架知识的掌握程度能帮到我们的范围非常有限，而真正让我能够突破上升瓶颈的恰恰是这些当初觉得不那么重要的课程所带来的认识。因为这些内容里所蕴含的设计思想和思维模式是未来指导我思考问题寻找解决方案的重要源泉！比如：很多就操作系统这一门课来说，其中设计到的各种**并发场景解决方案**、**缓存加速设计**等重要思想，都能对应到如今我们经常遇到的高并发、分布式场景下的各种问题。因为有这些基础认识的加持（方法论），你在思考难题和设计解决方案的时候，会更有想法，更容易找到好的办法！

在实际的工作过程中，对于开发童鞋在解决难题时候的一个特点。一些在大学里对这些基础课程掌握扎实的，往往只需要给他们直接讲一遍设计思路，基本上马上就能get到要点，因为他有了这些设计思想的基础，通过类比的方式去理解就会非常容易。而对于这些经典内容或者基础知识掌握欠缺的童鞋，往往你就得讲的更加细致一些，给他一步步介绍为什么要这样，他才能理解这么做的原因。

所以，**如果你是专业科班的学生，那我强烈建议你这些基础课程一定要好好上，不要觉得将来用不到，就不用心学，甚至不学！如果你是培训班出生，一样强烈建议把我上面说的三个基础课程好好自己补一下，因为培训班基本不会教你这些，他们只管教会你一些企业在用的东西，让你能快速的上手成为一个CRUD开发者**。

后记

我认为在一定程度上，对于**基础知识的掌握决定了你在技术生涯这条路上的上限**。

“知其然，更要知其所以然”。只有了解了本质原理，这样不论问题如何变，我们都能灵活的去解决它。就像我对很多想做架构的读者建议的，架构的本质一定要理解，一些基本原则要烂熟于胸，这些基础方法论是未来指导你实践最核心的内功心法。

切勿沉迷你过去的某一个成功经验来指导你未来的所有工作。架构工作最怕生搬硬套，学了微服务，到哪里都推微服务，学了中台，管他多大团队就干中台。最后，不但自己失败，还给团队和企业带来巨大的损失！这样的例子，这几年太多了，相信大家也都碰到过，之前我也写过一些不好的案例。究其原因，就是设计者对基础、对本质的理解不到位，一味求表面，赶新潮框架，追热点架构所导致的。

那么最后，你觉得大学里的这些过时的、未来不会用的基础知识重要吗？有必要花大力气学吗？留言说说你的看法吧！