如何在三年内快速成长为一名技术专家

工作前三年是职业生涯中成长最快的几年，在这段时间里你会充满激情，做事专注，也容易养成良好的习惯。在我们公司有些同学在前三年中就快速成为某一个领域的技术专家，有些同学也可能止步不前。本文和大家一起探讨下如何在三年内快速成长为一名技术专家。

学习方法

- 掌握良好的学习心态
- 掌握系统化的学习方法
- 知识如何内化成能力
- 广度和深度的选择

实战技巧

- 你需要学会的编码习惯
- 在业务团队做开发如何成长

掌握良好的学习心态

空杯心态

首先要有空杯的学习心态，而不是傲娇自满，故步自封，空杯子才可以装下更多的东西。首先要学会取百家之长，带着欣赏的眼光看团队的同事或学校的同学，欣赏每位同事或同学的优点，然后吸取他们的优点，每个同事都有其擅长的能力，比如有的同事技术能力强，那么可以观察下他如何学习的（或者找他请教学习方法），有的同学擅长解决线上问题，那么观察他是如何解决线上问题的，解决思路是什么？如果他解决不了时，他是如何寻求帮助。有的同学擅长使用IDE或MAC的快捷键，那么可以向他学习提高工作效率。有的同学能快速理解业务知识，观察他是如何做到的，自己如何达到他的程度。沟通能力，解决问题能力以及规划能力都可以向同事学习。

挑战权威

从书上看到一个知识点，或者从别人那里听到一个知识点，一定要去挑战和质疑这个知识点的正确性，否则学到的知识点可能是错误的。先用逻辑思维推测下，再实战检测

下，一定要记住实践是检验真理的唯一标准。比如同事说这个SQL加这个索引是最快的，首先要思考同事的结论是如何得出的，是靠历史经验还是测试过，如果我们没有经验，就加上这个索引跑下SQL，看看执行计划和执行时间，再换下其他索引试试会不会更快。依次类推，并发一定比串行快吗？无锁一定比加锁快吗？很多结论都是在特定的场景下才会产生的，一定要自己亲手实践验证下。

坚持学习

有的同学可能工作了五年，但是学习的时间可能一年都不到。学技术不能急于求成，只要学习方法正确，量变一定会引起质变。我在大学刚学JAVA时，怎么都学不会，但是坚持学习了几个月，每天看张老师的JAVA视频教学，买书按照书里的代码一行一行的敲代码，白天睡觉，晚上学习和写代码，写到宿舍关灯时就去避风塘呆一晚上，早上6点钟回宿舍睡觉，学到一定时间后，突然恍然大悟，才入了门。

在工作中，我曾经花了一个月的时间学习AOP的实现原理，学习了各种方式来实现AOP的原理，并写了几种实现方式的代码，虽然花的时间很多，但是到现在仍记忆犹新，对于排查问题和学习其他知识都非常有帮助。

要做到坚持学习，学习的环境非常重要。如果你想学，但是又不在学习状态，可以考虑换个学习环境，我经常会去星巴克看书和学习。我听说有的同事会周末抽一天去大学教师上自习。

把事做精

对自己要求越高，进步越快。要有强烈的把事情做完美的心态，我刚开始工作的时候，总是快而不精，做事做的不够细致，总希望快速拿出结果证明自己，但是反而证明不了什么，技术能力也得不到提升，缺少技术亮点，在团队中也没什么影响力，后面就开始锻炼一次就把事情做对的心态和方法。我观察过，很多人都擅长快速做事情，但是把事情做好做精致的人会比较少，但是结果却是**在精益求精的路上才会快速提高自己的能力。**比如用100行代码实现的功能，思考下是否可以用10行来实现，以便于降低运维成本，提高下次的编码效率。引用GUAVA等类库，提取公共方法，和使用JDK8新特性等。系统的方法压测过后，单机只能承受1700QPS，可以思考和实践能否优化下程序提高QPS，减少服务器数量。

把事情做精，一定是要强迫自己多花心思多花时间在这件事情上。有位技术牛人给我分享了一个心得，我觉得说的非常好，老板给你布置了一个任务，**你要花百分之150的精力做到100分，这样在老板那里你就能拿到80分或者60分。**

掌握系统化的学习方法

如果学习到的知识不成体系，那么遇到问题时就会非常难解决。有些同学会出现这些情况，比如编码时遇到问题百度搜索，如果百度上找不到答案，这个问题就解决不了。再

比如，在开发中要用到某个技术点，就学习下API，程序调通后就不再深入研究，浅尝辄止，如果程序遇到其他问题也不知道如何解决。

以上情况我认为叫**点状学习**。遇到一个问题，解决一个问题，需要一项技术，学习一项技术。那么如何由点到面，由面到体，形成系统化学习呢。

首先要确定学习的知识领域，需要达成的学习目标，针对目标制定学习计划，就像你要写一本书一样，先把目录写出来，然后根据目录上的知识点逐步去学习，最后把这些知识点关联起来，形成一个系统化的知识体系。学习的时候，可以制定一个计划，以周为单位，比如第一周学什么，第二周学什么。

比如我最近在学习人工智能，学习步骤是：

- 高数基础知识：线性代数，微积分和统计学。最近在打德州扑克时，我也会用统计学里的知识计算下输赢的概率。
- 人工智能基础：买几本书人工智能的基础书籍，如《机器学习基础教程》《Python机器学习》。
- 框架：TensorFlow等。
- 实战：在工作中找到一个应用场景，把学到的知识运用进去。

知识如何内化成能力

作家格拉德威尔在《异类》一书中指出，1万小时的锤炼是任何人从平凡变成世界级大师的必要条件。1万小时有多久？每天学习10小时，需要大约三年。但是很多人都工作了五年甚至更长，但是为什么成为世界级大师的却非常少。读者可以先自己思考下这个问题。接下来谈谈我的看法。

成长必须经历一个步骤，就是把知识内化成能力。**知识是用脑记住的，能力是用手练习出来的**。在工作的几年里，我们可能看过很多书，听过很多技术讲座和视频，但是通过听和看只是让你能记住这些知识，这些知识还不能转换成你的能力。

听和看只是第一步，更重要的是实践，通过刻意练习把听到和看到的知识内化成你的能力。

刻意练习，就是有目的的练习，先规划好，再去练习。首先给自己定一个目标，目标可以有效的引导你学习，然后使用3F练习法：

- 专注（Focus），专注在眼前的任务上，在学习过程中保持专注，可以尝试使用番茄工作法。
- 反馈（Feedback），意识到自己的不足，学习完之后进行反思，思考下自己哪些方面不足，为什么不足。

- 修正（Fix），改进自己的不足。

不停的练习和思考可以改变大脑结构，大脑像肌肉一样，挑战越大，影响越大，学习更高效，并且也会产生突破性。

广度和深度的选择

技术人员的学习路径有两个维度，深度和广度。很多程序员都有这个疑问，是先深后广，还是先广后深呢？

通过这么多年的学习和思考，我的建议先深后广，因为当技术学到一定深度后，就会有触类旁通的能力，自己掌握的广度也自然有了深度。但是在实际学习过程中，深度和广度相互穿插着学习，比如学习并发编程时，首先学习JDK源码，然后学进去之后，开始看JVM源码，最后看CPU架构，在技术点逐渐深度研究的过程中，广度也得到了完善。

所以无论哪种学习方式，学习态度才是最重要的，在广度学习的时候有深入研究的态度就能达到一定的深度，在深度学习的时候，主动学习相关的技术点，广度也得到拓宽。

你需要学会的编码习惯

程序员应该学会通过技术的手段来提高效率。几个常用的手段是使用工具，快捷键和编写脚本。

1. 使用各种工具

   技术人员电脑尽量用MAC，使用命令行效率一定比在 `1024*1024` 像素中找一个 `10*10` 像素的按钮更快。IDE用IDEA，比Eclipse更智能。命令行工具用iTerm和IDEA里的Terminal。写文章用MAC的客户端工具MacDown，左边编写，右边展示，比Word等工具方便快速很多。有时候我还会用按键精灵里配置脚本需要解决工作问题，比如通过点击我们的系统，来执行任务。这样的工具很多，只要能提高工作效率的工具，大家都可以尝试使用。

2. 使用快捷键

   MAC，IDEA和Eclipse有很多快捷键都要学会使用，比如在MAC命令行中通过idea .快速打开工程，通过open . 快速的打开文件夹，把IDEA里通过快捷键把一段代码抽成一个单独的方法，快速生成getter setter方法。

3. 用脚本写工具

   当我们用人工的方式做一件重复性很强的事情，首先要考虑使用工具来帮我们自动完成，如果没有类似工具，可以自己写个脚本来实现，这样除了能快速解决问题，还能提高自己的技术能力。

比如，我经常要在两个maven仓库发布jar包，我就写了个脚本来实现jar包的发布，`deploy.sh`代码如下：

```
cp pom.xml pom.xml.bak
rm pom.xml
ln -s pom-2-deploy.xml pom.xml
mvn deploy
rm -rf pom.xml
cp pom.xml.bak pom.xml
rm pom.xml.bak
```

在业务团队做开发如何成长

我一直在业务团队中做开发，在业务团队最主要的提高的能力是业务抽象和架构能力，通过业务场景，不断思考如何通过合理的架构和业务抽象能快速支持业务，降低运维成本。同时在这个过程中锻炼技术能力，比如写一些技术框架来快速支持业务，做到技术驱动业务。

可配置化的方式支持业务

设计业务的领域模型，把不随着业务逻辑变化的领域模型做成系统能力，把随着业务逻辑变化功能，做成可配置化，上一个新业务，通过配置的方式或少量开发就能支持。

在做客户后台功能时，由于需要展示的数据种类非常多，每种数据展示可能需要花费几天的时间，所以设计了一个通用的技术框架，实现了通过配置化的方式展示各种数据。

写框架解决业务问题

我在上家公司经常做一些CRUD的业务功能，我就自己开发了一个快速做CRUD的框架jdbcutil,通过配置实体生成SQL语句，实现了子类只要继承父类，就自动拥有CRUD的能力。后面还写过生成CRUD页面代码的程序。

目前我们团队在做的TITAN框架通过模块化开发的方式，解决易变的业务系统在多人开发时遇到的问题。

技术驱动业务

在业务团队，一定要不断的思考如何利用技术来支持快速支持业务，配置化是一种思路，但是有些功能配置复杂度比较高，配置加验证的工作量，可能需要一个星期的时间，那么能不能减少人工配置，实现系统自动化配置，于是可以研究下人工智能，通过人工智能的方式实现，系统告诉人需要配置哪些东西，然后交给人来进行确认，这样可以大大减少人工成本，更快的支持业务。