国内首套2016架构师速成教程

阿里架构师带你阶级架构师

**作者简介:**

****

* 本人资深苦逼it架构师，在it架构苦海中苦修10余年。进过华为，干过阿里。多次创业，屡战屡败，至今尚未成功。
* 爱coding，爱总结，爱说教，爱老婆，爱儿女，爱dota。
* 总结10余年苦逼经验，倾囊而授，助你架构速成，创业速成，顺便扒开伤口搏你一笑。
* 坚信每个人都是自己世界的神，欢迎来我的世界，你的一生即将改变！

[架构师速成-目录](http://blog.csdn.net/for5million/article/details/46584049)

天地会总舵，陈近南给了韦小宝一本武功秘笈，韦小宝说：“嗯？这么大一本我看要练个把月啊！”  
 陈近南说：“这本只不过是绝世武功的目录，那边才是绝世武功的秘笈！”

这就是架构速成的秘笈目录

* 架构师速成1-前言
* 架构师速成2-概述
* 架构师速成2.1-论成功
* 架构师速成2.2-论成功
* 架构师速成3-开发者境界
* 架构师速成4-幼儿园
* 架构师速成4.1-幼儿园要学会如何学习
* 架构师速成4.2-幼儿园要学会如何学习
* 架构师速成4.3-幼儿园要学会查找资料
* 架构师速成5-小学
* 架构师速成5.1-小学gtd进阶
* 架构师速成5.2-如何掌握综合性技能
* 架构师速成6-初中
* 架构师速成6.1-思维导图
* 架构师速成6.2-知识什么时候才是你的？
* 架构师速成6.3-设计开发思路
* 架构师速成6.4-开发框架
* 架构师速成7-高中
* 架构师速成7.1-速读、速记
* 架构师速成7.1-为什么要学习协议、规范
* 架构师速成7.2-devops为什么很重要
* 架构师速成7.3-架构师为什么要带团队
* 架构师速成7.4-性能优化为什么写的这么晚？
* 架构师速成8-本科
* 架构师速成8-研究生
* 架构师速成9-博士

[**架构师速成1-前言**](http://blog.csdn.net/for5million/article/details/46499823)

**版权声明：本文为博主原创文章，未经博主允许不得转载。**

从事it工作10余年，痛并快乐着。

忠告以下人员远离it：

1. 不能吃苦
2. 耐不住寂寞
3. 想赚大钱

如果你不是上面的人，而且非常想成为架构师话，请继续看下去。

1. 需要3年时间
2. 需要超强自制力
3. 需要极强计划能力
4. 需要吃苦

如果你能满足以上4条，那肯定就可以速成。可能有人会说“3年也算速成，这也太龟速了”，我回答你，如果没人系统指导，想成为一个合格的架构师需要10几年甚至更长，我只是帮你少走一些弯路而已。本人成为一个合格的架构师花了10年，而我的2个兄弟只用了3年。如果你更聪明更勤奋，甚至可以缩短到2年，那你将是同龄人中的top 100。

列举一下我的成功案例：

* 马同学 山东top3的软件公司技术总监
* 侯同学 新加坡国立大学计算机学院产学研中心技术总监
* 庄同学 某上市公司子创业公司CTO
* 李同学 某即将上市的营收10亿美金创业公司前端技术架构师
* 赵同学 我买网技术经理
* 王同学 先去搜狐，再去京东，现在是某银行子公司CTO
* 于同学 某top3数据公司技术经理

庄同学和李同学是我提到的2位用了3年就成为架构师的案例，后面希望他们也能给我写一些段子。

希望在后面的日子，我的文章能帮你迅速的成为你所在行业的top100架构师。

[**架构师速成2-概述**](http://blog.csdn.net/for5million/article/details/46508503)

成为一名合格的架构师，需要经历菜鸟、码农、资深码农、项目经理、技术经理、架构师等一系列的过程。为了让大家通俗易懂，我把整个过程按照大家熟知的上学的顺序排了一下，从幼儿园-小学-中学-…一直到博士，至于博士后需要大家自己去实践和想象了。每个过程我都会进行统一的描述：

* 阶段：例如 幼儿园
* 需要做的事情：例如 学会一门编程语言
* 完成任务耗时：例如 2-5个月
* 升级标准：例如：能写出简单的计算器，接受用户输入的+-x/运算
* 风险：例如 有人打断

当然在正式开始之前，我还是要提示一下相关的风险：

1. 任何行业想走到顶端都要付出超出常人的努力，it尤为明显，因为人多，而且新技术层出不穷。解决方案：
   1. 尽量不要搞it
   2. 做好准备，乐观面对，以苦为乐
   3. 大道至简，把握基础理论，新技术自然就懂
2. 经常加班，导致个人提升和工作关系很难协调，解决方案：
   1. 早起的2小时时间永远留给自己
   2. 尽量将个人提升和工作融为一体
   3. 开发自己的工具，节省时间
3. 经常加班，导致个人提升和家庭关系很难协调，解决方案：
   1. 单身的时候早努力，就不需要担心
   2. 学会时间管理，和另一半一起提升
4. 没人带，一个人学起来很苦，很难，解决方案：
   1. 三人行必有我师
   2. 学会时间管理、知识管理等软技能非常重要

ok，肯定还有更多的风险，我会再继续补充，其实风险根本不可怕，可怕的是不知道如何控制风险。

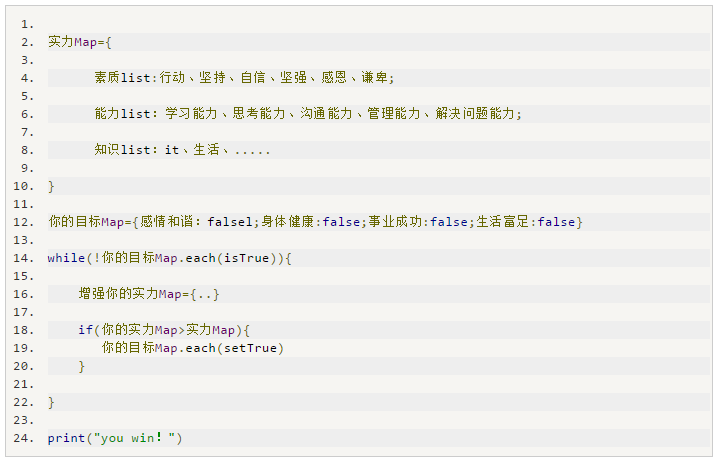
[**架构师速成2.1-论成功**](http://blog.csdn.net/for5million/article/details/46543767)

为什么在这里插入这么一个看起来无比宏大的问题？

是想告诉你成功是什么。知识在整个人生中的作用。

成功是什么？这是一个很难定义的问题。有人说知足，有人说有钱，有人说有权。我自己对成功的定义是自我情感的满足和自我状态的认同，其实就是知足，而成功是通过自身的提升的达到的。

自身实力包含什么呢？



所以你需要不断磨练自己的素质，提升自己的能力，积累自己的知识，不断投资自己。自己强大后外界的帮助也会随之而来，你离成功就不远了。

**架构师速成2.2-论成功(转载）**

查看文章索引请通过http://www.cnblogs.com/seesea125/archive/2012/04/17/2453256.html

成功是大部分人都梦想的，如果想成功，我们都需要些什么？下面是我的总结，正确不正确都请欢迎拍砖！

一、 成功必须有正确的目标。

目标就像是航海的灯塔，如果没有，我们很容易迷失方向。目标也是动力的源泉，试想一下，比如我们跑步，如果我们目标都没有，可能跑几步觉得累，停下来休息了，如果我们目标是1500米，那我们可能就会呼呼的跑了1500米，停了下来休息，如果我们目标是10里地，我们可能真的超乎想象，跑完了10里地。

所以目标是成功第一因素，目标越大，人就会走的越远！

确定目标的几个注意因素：

1 目标当然也要切合实际，不切合实际的目标只会打击自己的积极性，或者虎头蛇尾，坚持一段时间看不到希望就放弃了。

2 目标要有大的目标，然后有小目标，只有把大目标分成一个阶段一个阶段小目标，实在每个小目标，才能最终实现大目标。

3 目标最好是选择自己最感兴趣的东西，兴趣是最好的老师，兴趣也是成功的关键！一个人如果不做自己感兴趣的事，是不可能做到卓越的！当然很多人不知道兴趣在哪，当问起你的兴趣爱好是什么时，基本都是玩、旅游、睡觉之类的，这不行，目标必须是正事，哈哈。如果实在不知道兴趣在哪，那就找自己最擅长的事作为目标，因为兴趣也是一点一点的小成功，获得了满足感，才产生的兴趣！如果不擅长的事，一次一次打击自信，那绝对不会爱好它了。如果擅长的事也不知道，那就让别人分析下或者自己多分析下，自己的性格特征，看什么行业比较适合你，或者多换换行业也可以，最次的是换行业也艰难了，比如已经工作几年了，那就认真把现在的行业给做好做专把。

二、正确的目标就相当于战略，我们还需要战术来配合去实现它

很多人有了目标，却不知道怎么实现它，最后也不了了之了，战术就是达到战略的手段了，所以确定下目标后，战术就是第二重要的东西。要想有正确的战术，那就必须要有大量的了解、学习和调查。比如说你现在是一个初级程序员，你想成为高级程序员，成为架构师，最后成为技术总监，这个是目标，战术上就是你具体怎么做？那就需要认真分析，高级程序员都了解什么技术，具备什么品质，架构师都要学什么，总监都要懂什么，需要什么经历，什么学历等等，我们可能不在那个层次，无法了解那个层次，这时候就特别的需要做认真的了解和调查，比如多交往架构师或者总监，了解他们比你优秀的地方，了解他们要必须懂的知识，必须有的经历等，当把这些都搞明白后，我相信，你就明白了，你都哪里不足需要认真学习的，然后做个计划，开始弥补差距把。

三、我们还需要良好的人际关系

在中国，是人情的社会，做事的成功，首先是做人的成功，否则就算能力再出众，也会因为没有机会，得不到重用，在公司里也同样，如果和上司、同级、下属关系处理不好，也不会有太多发展，人际关系的重要性甚至超出了能力。

关于怎么拥有好的人际关系，一句话也说不完，基本上有几点需要做到的

1 真心，热诚，乐于帮助他人

2 善于沟通交流

3 学会做人做事，做人要直，做事要曲

4 要学会从对方立场去考虑，己所不欲勿施于人

5 不要有攻击性，要有亲和力，让别人和你交往很舒服，很自然

人际关系是我们必须要长期修炼的，不能懈怠的东西，我们要经常反思自己的缺点和不足，以及经常总结如何处理各种人际关系中的问题。

四、我们还需要一些优秀的个人品质

像目标、战略、处理人际关系都是外练的内容，我们还需要内练基本功，这是内功，内功深厚，才能练出好功夫，否则都是花拳绣腿了。因为内练品质也非常重要，需要长期修炼。基本上有几点需要做到的。

1 诚实

这个就不用说了，一个不诚实不阳光的人，是不可能拥有好的人际关系的，有也顶多是一些狗朋狐友了。要想做到诚实，就必须行的正，不做缺德的事，俗话说玩物丧志、玩人丧德，在现在骗子横行的社会，保持诚实尤显得珍贵。

2 谦虚

谦虚太重要了，这个一定要做到，俗话说谦虚使人进步！只有谦虚了，我们才能看到别人的优点，向别人学习，也才能知道自己不足的地方很多，需要努力去学习完善自己。而且这样的人也具有亲和力，谁喜欢和一个喜欢自夸，或者刚愎自用，听不进去建议的人交往呢？邓爷爷、毛主席等老一辈经常教导的，就是做人要低头，也是这个意思。

3 自信

周总理曾说过，一个人的自信心和能力成正比，可见自信的重要性，如果一个人都不自信，那他干什么都觉得自己干不好时，最终结果也肯定干不好，或者甚至连做的勇气都没有！一个人的自信心也和最后的成功有多大，有很大的关系，因为我们要时刻保持自信，相信自己，别人能做到的事，自己也一定能做到，不要对自己的能力产生怀疑！

4 自律

要学会自我控制。前途是美好的，道路是曲折的，每个人都希望成功，但是成功的只属于少数，我们必须付出和别人更多的汗水，付出比别人更多的艰辛，没有人能随随便便成功，如果不懂得控制自己，强迫自己去努力，那结果就是一直看着目标在前方，但就是走不到！甚至是看着目标睡大觉，一醒来摇摇头，感叹一下人生如梦啊。。。。

5 爱

这个太牛了，一定要做到的，人活的一切都是因为爱，我们要发自的内心的去爱这个世界，去爱身边的亲戚、朋友、同事。只有爱，才懂得感恩，只有感恩，才懂得自己的使命和责任，一个人如果太自私，那他的成功也会变得没有意义，因为我们的奋斗的目的，就是为了让这个世界变得更好点，或者让亲戚，朋友更开心点，只有让别人幸福了，自己的价值才得到体现，自己也才能幸福。一个人的价值，往往不是他得到了多少，而是他付出了多少！世界上名垂千古的人，都是为了这个世界做了什么贡献，否则就算出名，也是遗臭万年的！

这些也需要长期坚持不懈的去修炼。

6 坚持

为了目标，就要学会坚持，成功路上必然会遇到很多意想不到的困难，如果轻言放弃，那只能半途而废了，而根据科学调查，成功的人，他们往往有一个普遍的特征，就是他们知道坚持下去，不达目标决不罢休。

五、唯有不断修炼，才能走向成功。

世界上没有天才，天才也是不断的修炼练成的。只有不断的修炼，才能完善自己，才能走向成功。

修炼的内容，当然就是上面说的这些了。具体怎么修炼，就是要多学习，然后多实践，就是学以致用了。

当然要成功，需要学的东西太多太多了，因此还有2个两个事必须要认真研究和锻炼的。

1 学习方法

要学的东西很多，我们只有找到适合自己的学习方法，才能事半功倍，学习方法也要靠经常总结，如果学习方法不对，那学起来速度太慢了，成功也太慢了点，等老了才成功，这个估计谁都受不了。

2 注意力

天下武功，无坚不破，唯快不破！唯有以快制慢，方能笑傲江湖！我一直在想，为什么有的人学了几个小时，却可能比其他人学了几天都学的东西要多？后来我想明白了，原来他们一直在修炼他们的一样东西—注意力！

如果一个人注意力集中，则可能用2个小时干完一天的工作！一个注意力集中的人，更能培养出思维能力，理解能力和记忆能力。像我们平时工作，动不动开个QQ，或者看个新闻之类的，效率其实很低，老觉得自己没时间，忙的很，实际仔细想想，大部分时间都浪费了！

基本就这么多了，这个相当于总纲，细节有很多东西，网上，书上也有很多，比如卡耐基成功学，希尔成功学，还有毛主席选集，都是很经典的东西，以后有时间我也会继续写点总结什么的。当然说着容易做着难，实际上我们可能什么道理都懂，但是就是不想下这个苦功夫，哈哈，因此也需要建立正确的人生观，端正人生态度，不要受那些及时享乐，人生苦短等消极的理念所影响，保持积极的心态很重要哦。多学，多思考，多实践，多总结，就算最后没成功，也要死在走在成功的路上，很多时候真的努力了，最后的结果你会发现，你得到的结果比你开始想要的要好的多，可能好到我们自己都不敢相信，因为具备了优秀的品质，那你就会从一个成功走向另一个成功，而失败也这样，从一个失败走向另一个失败，很多事都是有因果关系的！人生就是这样的，成功属于有准备的人，有勇气的人。

[**架构师速成3-开发者境界**](http://blog.csdn.net/for5million/article/details/46529817)

修炼的境界自下而上分为：

**筑基、开光、融合、心动、金丹、元婴、出窍、分神、合体、洞虚、大乘、渡劫**

其实开发者也可以按照修炼的境界进行划分：

**入门、对象、模式、框架、架构、成神**

1.入门 初学者就是为了把功能实现，不考虑其他，此时根本不考虑可读或者可修改性。  
2.对象 以面向对象方式进行编码，把代码分开写到不同的对象中，能够进行跨对象的交互。  
3.模式 关键点可以使用设计模式进行设计  
4.框架 某一语言内部进行高度封装，使常用的功能开发步骤极度简化，提升开发效率，并极大降低对开发人员的要求。例如使用spring mvc进行封装，对错误、事务、日志等进行统一处理，或者更进一步对前后端交互进行封装。  
5.架构 针对不同场景，进行跨语言、跨系统、跨容器进行设计及高度封装，使系统高可用，并能支持高流量，高并发。

[架构师速成4-幼儿园](http://blog.csdn.net/for5million/article/details/46552419)

* 阶段： 幼儿园
* 完成任务耗时：1-3个月
* 升级标准
  + 熟练使用word、excel、ppt软件
  + 能写出简单的计算器，接受用户输入的+-x/运算
  + 掌握基本的沟通技巧

下面详细讲一下怎么做，其实很多人都比较讨厌那种讲空洞理论的书，me too。既然我们讲技能型的知识，就讲一下如何去实践。

word、excel、ppt就不需要详细讲了，但是这3个工具会对你未来的发展有很的影响，尤其是ppt能力，写一个打动人心的ppt，是你赢得赏识，快速晋升的法宝。下载一个金山wps，练习一下word的排版，excel的多行计算，制作图表，做几个简单的ppt。幼儿园期间仅要求熟练的使用，不要求精通各种高级技能，后面会逐步要求提升。

下面几个是给你预习的材料，有余力的同学可以预习一下：

* [《别告诉我你懂PPT》思维导图读书笔记](http://www.write.org.cn/do-not-tell-me-you-understand-ppt-mind-map-reading-notes.html)
* [《说服力：工作型PPT该这样做》思维导图读书笔记](http://www.write.org.cn/persuasive-ppt-the-type-of-work-to-do-mind-map-reading-notes.html)
* [《说服力：工作型PPT应该这么做》PPT制作七步法](http://www.write.org.cn/persuasive-ppt-type-of-work-should-do-so-ppt-produced-seven-step-method.html)
* [《写给大家看的PPT设计书》思维导图读书笔记](http://www.write.org.cn/we-wrote-the-book-to-see-the-ppt-design-mind-map-reading-notes.html)
* [《PPT，要你好看》读书笔记思维导图](http://www.write.org.cn/ppt-want-you-to-look-good-reading-notes-mind-map.html)
* [《乔布斯的魔力演讲》思维导图读书笔记](http://www.write.org.cn/jobs-of-the-magic-of-speech-mind-map-reading-notes.html)
* [Robin Williams《写给大家看的设计书》思维导图读书笔记](http://www.write.org.cn/robin-williams-xie-gei-da-jia-kan-di-she-ji-shu-si-wei-dao-tu-du-shu-bi-ji.html)
* [《PPT演示之道》思维导图](http://www.write.org.cn/ppt-yan-shi-zhi-dao-si-wei-dao-tu.html)

掌握一门编程语言，后端语言更好，例如：java，scala，php，python等等，这个是有诀窍的，一般人我不告诉他。以java为例，

1. 先上网搜索相关的入门书籍推荐，会找到比较多的人推荐，整理成列表<thinking in java>,<head first java>,<java 核心技术>，<java从入门到精通>等等。
2. 根据推荐度进行筛选，如果这时候咨询一下有经验的前辈，会得到比较中肯的答复。我推荐<head first java>。
3. 下载编辑器，例如eclipse，这个一定要下载，别听某人说某牛人用记事本编写了niubility的程序。那不是牛人，那是傻x。我的感觉就是，砍柴的时候有砍刀不用，非得用手，智商着急。
4. 用2天时间，先把书看一遍，建立一个宏观概念。java到底可以干什么，有什么东西要学，此时不要纠结很多看不懂的东西。
5. 开始每章突破，每章看完之后，自己敲所有的代码。切忌去网上下载了执行，而且尽量每个代码敲5遍（这个不信就算了）。此时如果有特别不明白的就问一下前辈，如果没有可以去看视频。当然你看见有些后面才会讲的内容就不要纠结了，能执行就行。尽量高速的完成每一章，纠结太久是错误的，每章1-2天。
6. 做一个计算器。

掌握一项技能，沟通。“谁不会沟通啊，这还用学”有人暗骂。我只想说“呵呵”。说话大家都会，但是沟通不只是说话，而是把话说的让别入明白。好好学吧，你会发现这太有用了。怎么学呢？

1. 同java学习，我就是喜欢你这样能够举一反三的学生。
2. 但是这个要找人练习，直至做到无意识的使用沟通技巧，即变为被动技能。
3. 顺便推荐一些相关的资料
   1. http://www.write.org.cn/category/psychology/communication-skills

补充：前面学到的东西，后面阶段有可能会提到，有可能不会提到，但是默认你会自己安排时间不断的去练习。

# 架构师速成4.1-幼儿园要学会如何学习（转载自36氪）

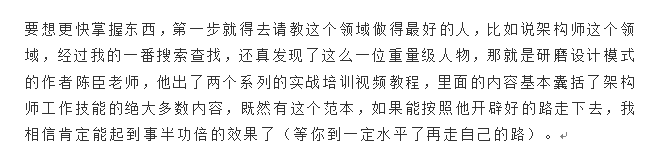
*在这个资讯爆炸、新东西层出不穷的时代，学习是件终生的事业。书到用时方狠少，很多时候我们总是后悔当初没学，然后抱怨时间不够多，最后开始反思—时间对每个人来说都是公平，所以我们的问题应该是如何才能更快地学会东西？也许Sean Kim的这篇*[*学习终极指南*](http://rypeapp.com/blog/learning-curve/)*可以参考。*

如何缩短学习曲线这个课题已经被研究了几十年。其中讨论到的一些加速学习进程的核心原则这份指南都会讨论到。利用好这些原则，无论你学的是语言、乐器等等，都可以学得更快。

在当今我们生活的这个时代，如果你想过上一段非凡的人生，就得掌握快速学习这项本领。        —Anthony Robbins

**不做重复工**

我们学东西有个倾向是试图自己掌握，但是有个学会的人帮助可以节省你很多的时间和精力。想想自己以前是怎么学语言或技能的。是不是一开始走了很多弯路？有了你的帮助，后面的人再学是不是就可以避免很多你犯过的错误？



正如Tony Robbins所言：

许多伟大的领袖已经证明，掌握任何技能、策略或目标的最快方式就是照着前面的人开辟的路走下去。如果你能找到已经拿到你想要的结果的人并且采取跟他们一样的行动，你也可以获得同样的结果。

你的年龄、性别、背景如何都没有关系，照着做让你可以快速跟踪并以短得多的时间实现你的梦想。

在今天这个时代，你几乎从通过书本、博客、培训视频、咨询或网上获得任何解决方案。

毕加索曾说过：

好的艺术家抄袭 伟大的艺术家剽窃。

（这话乔布斯也剽窃过。:)）

**技能解构**

破解学习曲线的下一步是将要掌握的技能解构为基本要素。把这些要素分解，然后找出最重要的首先进行练习。

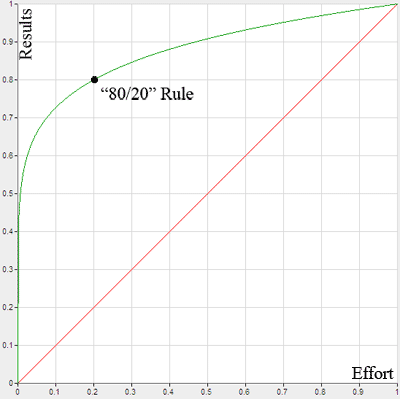
这个做法源自帕累托的2/8原则：即用20%的努力获得80%的结果。



这个原则几乎到处都可以体现：

* 商业（80%的销售来自20%的客户）
* 员工效能（80%的成果出自20%的员工）
* 快乐（80%的快乐来自20%的关系）
* 旅游体验（旅游80%可用20%的精彩时刻概括）

下面这张图是2/8原则的另一种表现形式（横轴为努力，纵轴为效果）：



2/8原则的要点是只有很少的东西会对我们的生活（包括学习）产生很大影响。

那么我们的目标就是把这能产生80%结果的20%的学习材料独立出来。

实际上一些快速学习专家早已采取这种意识形态。

Josh Kaufman在TED演讲中提出，掌握一项技能并不需要1万小时的练习。关键是前面的20小时要学会最重要的子技能获得最大的效应：

动作与认知技能获取领域的若干研究表明，新技能前面几个小时的练习对表现产生了最大的影响。

一般模式是这样的：一开始时你很害怕，但你学会技能最重要的部分之后改进就会非常快。

相关学习：[帕金森定律](http://baike.baidu.com/view/40865.htm)。

举例：

* 学习一门乐器必须知道少数最常见的和弦，这样80%的曲子都能弹。
* 学习新外语应该聚焦在最常见的1500到2000个字，这样80%的文字都能懂。

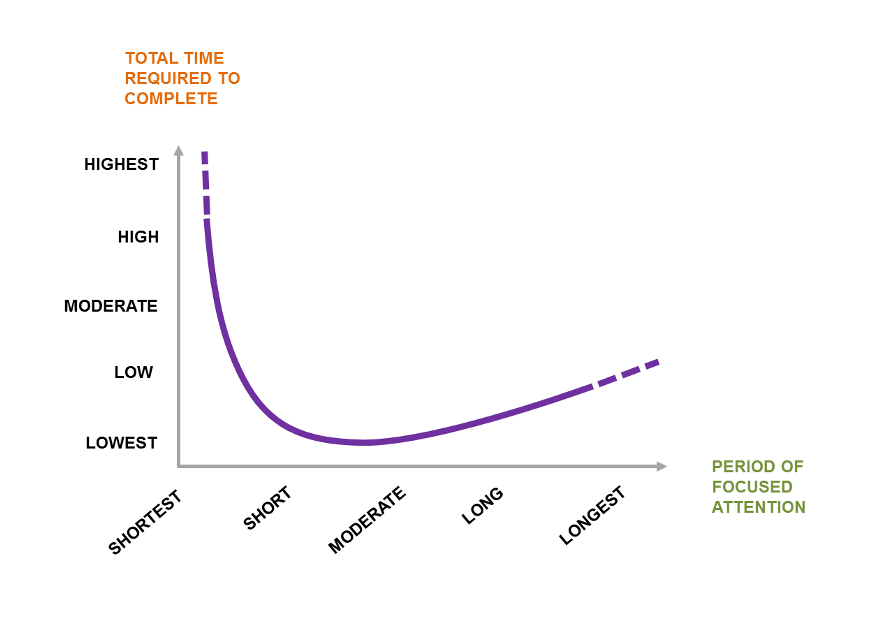
**一心不能二用**

多任务是我们在连续通告和移动应用时代培养出来的一种罪恶快感。每10分钟就要查查邮件、刷刷微博、看看微信或者跟经过的同事聊聊天等都是表现。

但是多任务处理会成为加快学习的最大障碍。

可以想想自己的电脑。

浏览器打开了20个以上标签页时，计算机就会开始变慢，然后执行后续动作就会放缓。



*横轴：集中注意力的时间，纵轴：完成需要的总时长*

研究表明，人如果分心后平均需要**25**分钟才能让心思回到手头的工作上。

更值得注意的是，加州大学的Irvine发现，自己的同事往往只干了**11**分钟就开始分心。

长期专注也一样。我们很多人都无法腾出6到12个月以上的时间去学习一项技能，因为总会有新项目、新想法、新爱好冒出来。

当我们决定把注意力转移到新的事情上面时，往往很难再对之前技能保持同样的激情和专注。

在解构出能产生最大结果的子技能之后，就得把精力集中在改进这些子技能上，在掌握这些子技能之前不要学其他任何东西。

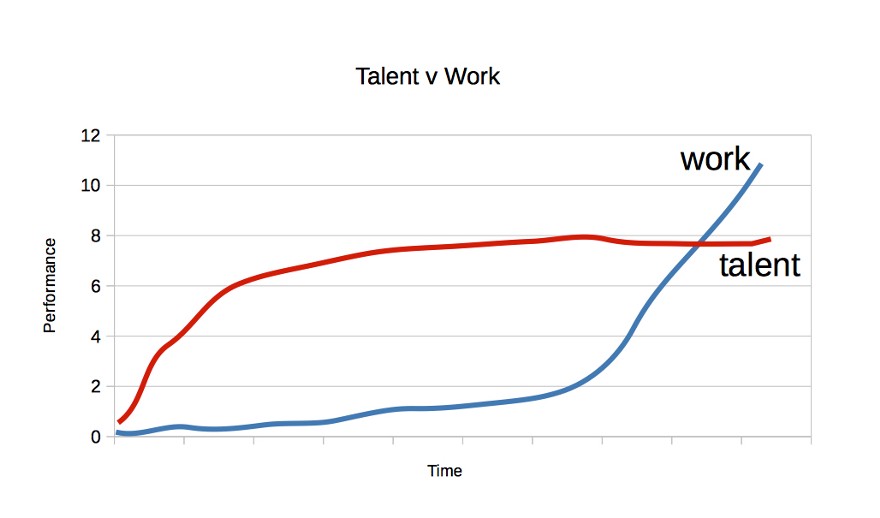
**重复练习**

这是我们大多数人最纠结的部分。

**是的！**更快掌握任何东西的关键是练习。

这需要频繁持续地反复练习同一种技能，直到不假思索就能下意识地做到。

全世界表现最好的人都理解这个学得更快成为最佳的“秘密”，但是却很少会讨论它的重要性，因为这听起来并不性感。



*专家级表现主要是专家级练习的结果，而不是因为与生俱来的的天才。*

佛罗里达州立大学的K. Anders. Ericsson指出：

大家认为，专家表现跟一般表现之所以有质的不同，一定是因为专家有与生俱来的天分。这种看法让科学家忽视了按照普通心理学定律原则去检视专家的表现。

**寻求即时反馈**

1960年当时还还默默无名的甲壳虫乐队去德国汉堡的场子演出。

报酬低、音效差、听众不欣赏就是当时他们的感受。如何避免呢？

不停地练习然后马上听取反馈迫使他们变得越来越好。

Macolm Gladwell在《异类》中总结认为，这是甲壳虫登上音乐殿堂巅峰的关键不同。

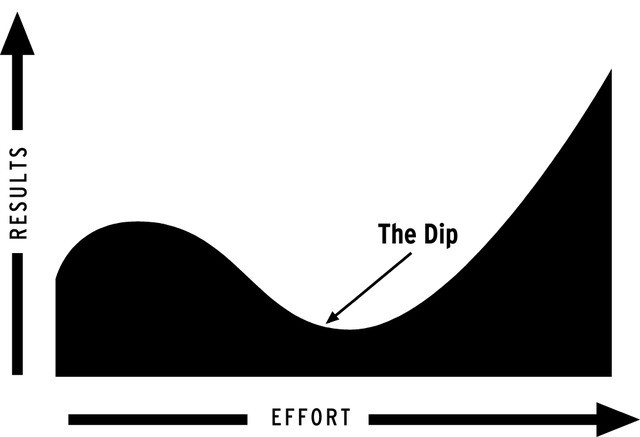
哥几个并不只是呆在车库里面埋头练习，他们同时还努力站在现场观众面前，无论是喷口水还是建设性意见，均能获得第一手的反馈。

随着他们水平的提高，观众要求他们演出的时间也加长了。到了1962年，他们每晚的演出时间提高到了8小时，而且1周7天不间断。到了1964年他们在国际舞台崭露头角时，甲壳虫乐队已经完成了1200场音乐会。

相比较而言，今天的大部分乐队整个职业生涯的演出都达不到这个数字。

**坚持**

不幸的是，大多数人在到达赛斯·高汀所谓的“低谷（The Dip）”期间或之前就已经放弃努力。



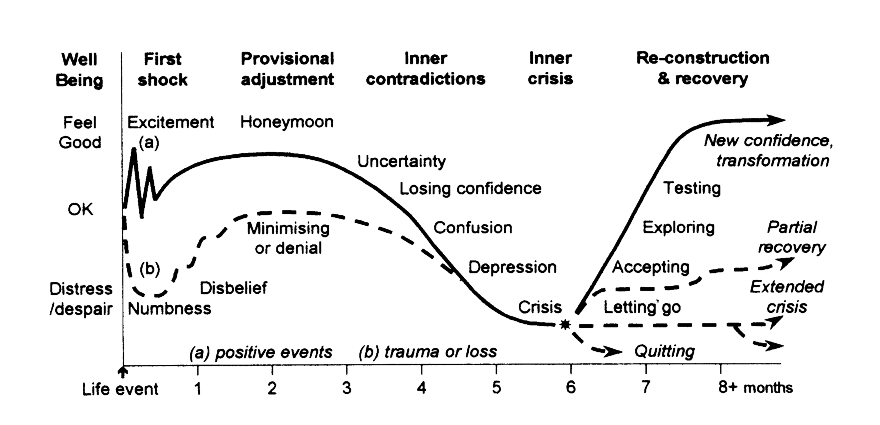
高汀说，尽管知道适时学会放弃很重要，但是很多有可能取得成功的人正是没有选好放弃的时机。

你没能成为全世界最好有5个原因：

* 你时间用完（然后放弃）了。
* 你钱用光（然后放弃）了。
* 你害怕（然后放弃）了。
* 你不认真对待（然后放弃）了。
* 你没兴趣（然后放弃）了。

心理学家也研究过所谓的转变周期。

这是指人经历变故（如悲伤事件）或新奇事件（如学新东西）时的心理变化周期过程。



如图所示，我们一开始学新东西时都会经历过精神愉快。这也是我们看社交媒体的通知会上瘾的原因，因为每次都会释放多巴胺。

不过，随着蜜月期渐逝，我们就会进入“低落”期，进展开始放缓或者减少。这段时间也是最多人放弃的。

把这个周期进行可视化很重要，因为这会让你明白这是黎明前的黑暗。如图所示，熬得过这段时间的人很快就经历了新的突飞猛进。

总结：

* **以专家为榜样**，不必自己重新探索，最好找资深专家的权威教程
* **解构技能**，找出实现80%效果的那20%
* **不要一心二用**
* **练习练习再练习！**然后获得即时反馈
* **坚持，**不要在低谷期放弃

# 架构师速成4.2-幼儿园要学会如何学习

《如何高效学习》，这本书的作者是scotthyoung，最知名是的1年内自学完成4年麻省理工学院计算机科学的33门课程，同时也写了一个学习方法的Blog,他使用费曼技巧来加强理解和学习。  
费曼技巧很简单：

1. 拿张白纸;
2. 在白纸顶部写上你想理解的某想法或某过程；
3. 用你自己的话解释它，就像你在教给别人这个想法。

最要紧的是，对一个想法分而化之，虽然可能重复解释某些已经弄懂的知识点。但你最终会到达一个临界点，无法再解释清楚。那里正是你需要填补的知识缺口。为了填补这个缺口，你可以查课本、问老师、或到互联网搜寻答案。通常来说，一旦你精准地定义了你的不解或误解，找到确切的答案则相对而言更轻松。

另一种说法:

第一步 – 选择一个你想要理解的概念

选择一个你想要理解的概念, 然后拿出一张白纸, 把这个概念写在白纸的最上边.  
第二步 – 设想一种场景，你正要向别人传授这个概念

在白纸上写下你对这个概念的解释, 就好像你正在教导一位新接触这个概念的学生一样. 当你这样做的时候, 你会更清楚地意识到关于这个概念你理解了多少, 以及是否还存在理解不清的地方.

第三步 – 如果你感觉卡壳了, 就回顾一下学习资料

无论何时你感觉卡壳了, 都要回到原始的学习资料并重新学习让你感到卡壳的那部分, 直到你领会得足够顺畅, 顺畅到可以在纸上解释这个部分为止.

第四步 – 为了让你的讲解通俗易懂，简化语言表达

最终的目的, 是用你自己的语言, 而不是学习资料中的语言来解释概念. 如果你的解释很冗长或者令人迷惑, 那就说明你对概念的理解可能并没有你自己想象得那么顺畅 — 你要努力简化语言表达, 或者与已有的知识建立一种类比关系, 以便更好地理解它。

http://jingyan.baidu.com/article/dca1fa6f59896bf1a440528d.html  费曼学习法的详细讲解

我看了这本书的中文版，觉得还是比较有道理的。发现还有一个很重要的概念，就是通过联想不断的建立你掌握知识的关联，这个跟思维导图的逻辑有些想通。举个例子，程序相关的吧：

缓存 其实是一个攻防战，用户是进攻方，架构师防守方，第1层阵地就是用户的浏览器端，第2层是cdn，第3层是nginx等web容器静态缓存，等等 这是讲解。

当然缓存还会跟其他的知识进行关联 http协议、db除了缓存还可以进行分布式等等。你发现整个大领域的知识都是相关的，你不停的关联，你的知识体系就会越牢固。

如果这个学习法，我早些学到，哇哈哈！

# 架构师速成4.3-幼儿园要学会查找资料

有很多人整天用电脑，但是碰到问题就不知所措，好像从来没有用过互联网一样。

互联网给了我们最多的知识，但是也给我了最大的诱惑，也给了我们最多的垃圾。如果迅速找到自己想要的知识，取其精华，弃其糟粕是一个非常有用的学问。还是举例子：

比如我要学习一门新的语言scala，

* 列出对自己帮助比较大的几种资料或途径
  + 书籍
  + 视频
  + 别人学习的心得和经验
  + 相关论坛和答疑群、博客，有疑问可以咨询
* 光撒网，搜集所有相关资源
  + 书籍 从百度文库或者百度网盘，另外还有一个资源网站 http://www.lefashn.com/other/resource.htm
  + 从豆瓣搜索相关书籍和别人评价
  + 视频 从优酷或百度网盘，或it相关的视频网站
  + 心得或经验， 就需要baidu或google 看有没有相关人的博客或论坛
  + 论坛或答疑群、博客 同上
* 去粗取精
  + 书籍 前期肯定找入门和基础的，口碑特别好的，如何排除就看别人心得、豆瓣评价、销量
    - 快学scala
    - scala编程
    - scala程序设计
    - effective scala
    - twitter的入门教程
  + 视频 也粗略的看一下目录，找一套就可以
  + 论坛和博客都找最近有更新，比较活跃的
* 速读
  + 迅速的过一下所有的书籍，明确知识概况，确定重点难点。
  + 制定学习计划
* 学编并行
  + while(!ok){
  + 先看书
  + 敲代码
  + 如果有视频就看视频，可以没有
  + }

[架构师速成5-小学](http://blog.csdn.net/for5million/article/details/46572933)

很高兴你很快的进入了小学，小学的东西会让你更加的耀眼。

* 阶段： 小学
* 学时：2-3个月
* 升学标准
  + 能自己制定目标及计划，get thing done。
  + 可以轻松制作一个让你身旁人惊叹的ppt
  + 能做一个简单的网站（或者客户端软件），数据能保存到数据库。

实践经验干货来了。

先说ppt吧，这个上一期已经讲了，如果你ppt做到出神入化，基本不需要做架构这么苦逼的事情了。因为你很容易成为老板的心腹，军师，走上人生正道。作为一个苦逼的小学程序员，很羡慕吧。那就再努力学一下，不用多久,你就会升职加薪,当上总经理,出任CEO,迎娶白富美,走上人生巅峰。想想还有点小激动呢,嘿嘿！

* [《别告诉我你懂PPT》思维导图读书笔记](http://www.write.org.cn/do-not-tell-me-you-understand-ppt-mind-map-reading-notes.html)
* [《说服力：工作型PPT该这样做》思维导图读书笔记](http://www.write.org.cn/persuasive-ppt-the-type-of-work-to-do-mind-map-reading-notes.html)
* [《说服力：工作型PPT应该这么做》PPT制作七步法](http://www.write.org.cn/persuasive-ppt-type-of-work-should-do-so-ppt-produced-seven-step-method.html)
* [《写给大家看的PPT设计书》思维导图读书笔记](http://www.write.org.cn/we-wrote-the-book-to-see-the-ppt-design-mind-map-reading-notes.html)
* [《PPT，要你好看》读书笔记思维导图](http://www.write.org.cn/ppt-want-you-to-look-good-reading-notes-mind-map.html)
* [《乔布斯的魔力演讲》思维导图读书笔记](http://www.write.org.cn/jobs-of-the-magic-of-speech-mind-map-reading-notes.html)
* [Robin Williams《写给大家看的设计书》思维导图读书笔记](http://www.write.org.cn/robin-williams-xie-gei-da-jia-kan-di-she-ji-shu-si-wei-dao-tu-du-shu-bi-ji.html)
* [《PPT演示之道》思维导图](http://www.write.org.cn/ppt-yan-shi-zhi-dao-si-wei-dao-tu.html)

再说gtd，这个是你人生的利器，掌握了他，基本就会比同龄人花更少的时间，赚更多的钱，升更快的职，玩更多的

–

–

–

–

-，Dota。想歪了的，先站一旁反省一下。

用比较通俗的说法说一下，人生就像一次旅行，说走就走。有人坐飞机（x二代），有人坐豪车（还是x二代），在这看我文章的人，肯定不是什么二代，所以只能靠自己的双脚。小A和小C都想靠自己的双脚来一次说走就走的旅行，小A真的说走就走了，啥都不带就走了。小C不着急，先找来地图，确定要去北京（为啥是北京啊！），然后查了一下大约多少公里，每天走多少公里，需要带什么东西。准备完毕，小C也出发了。那问题来了，挖掘机哪家强？谁先到北京？

有人这时候就要开始抬杠了，争论了，辩论了，我只想说别扯蛋。有闲功夫扯蛋的，都不适合做架构师。

gtd就是目标+规划+执行。怎么掌握呢？

1. <小强升职记>看10遍。跟你身旁最亲的人讲2遍。
2. 推荐一个工具，绝对没有收钱的推荐。doit.im，免费的，每天都用，坚持1个月。
3. 找你的前辈帮你提出目标的建议。
4. 坚持执行你的计划，坚持！

怎么做一个网站或客户端呢？

我就以网站为例讲解一下，因为网站更难，客户端更简单一些。

1. 快速掌握语言 html,css,javascript的基础知识，这些因为已经有了之前的基础，花费时间应该不多，具体方式参考java学习。不要求做到精通，仅要求掌握。
2. 快速掌握mysql的安装，使用，掌握sql语句。具体方式参考java学习。
3. 掌握一个web容器，如tomcat，jetty，学会如何启动，关闭，部署。
4. 综合起来进行web开发
   1. 找一本做web网站的书或视频（针对你会的语言找），速度过一遍，
   2. 下载demo跑起来。
   3. 修改前端，看看改了之后的效果。
   4. 改后端代码，打印一些日志。
   5. 再看配置文件（xml,properties,ini)之类的，搞明白每个部分的作用。
   6. 找一个从前到后例子，自己实现一遍。至少实现5个类似的功能。
   7. 自己从头到尾创建项目进行开发，直到跑起来为止。

简单吧，请抓紧时间完成你的学业。

www.100j.top

www.sishuoou.com

# 架构师速成5.1-小学gtd进阶

人生没有理想,那和咸鱼有什么区别。

有了理想如何去实现，这就是gtd需要解决的问题。简单说一下gtd怎么做？

1. 确定你的目标，如果不能确定长期目标，至少需要一个2年到3年的目标。
   1. 目标必须是可以衡量的，不能模棱两可。
   2. 目标必须符合你的价值观，如果确定是否符合价值观，假设这件事给你10万，让你不要继续做，你说滚。
2. 针对你的目标，确定需要做的事情，脑子里想到的所有事情都放到收集箱，放空自己。
   1. 必须把所有的事情放进来，让大脑做到空无一物。
   2. 放到收集箱不行，必须确定有解决方案，否则大脑潜意识还是会纠结。
3. 收集箱的事情，进行分类整理，分成马上执行、项目、日程、下一步计划、将来也许
   1. 马上执行的事情，马上解决掉
   2. 项目必须进一步细化，细化成日程或者下一步计划
   3. 日程设定定时执行
4. 每天早上把脑子里的事情放到收集箱，同上整理一遍，然后过一下所有的下一步计划和日程，今天需要做的放到今日执行。另外过一遍所有的目标，看看有没有新的想法需要实施。
5. 将今天需要做的事情分清优先级，按照优先级逐个完成。
   1. 优先级，请按照四象限方法进行安排，一定要注意，不要被紧急不重要的事情占用时间，尽管我这样说了，但是还是有很多人整天被紧急不重要的事情占得满满的。
   2. 一旦确定优先级，就一定要按照优先级执行，不要被其他人打断。
   3. 如果有人要打断你，一定不要被打断，尽量完成手头的工作。如果非要打断不可，那一定要保存好断点，让你回来马上能够接上。
6. 晚上过一遍今天的工作，反省一下。
   1. 确定一下自己的计划是否完成
   2. 是否被人打断
   3. 今天有哪些做的不够好的地方需要改进
   4. 今天有哪些做的好的地方，需要坚持

有人生来就是当臭咸鱼的，但你不是，那就get things done！

# 架构师速成5.2-如何掌握综合性技能

买菜、洗菜、切菜很多人都会，但是把几种菜混在一起做成一道可口的佳肴就不是一般人能掌握的。

今天就讲解一下如何学会做一道好菜。

举个例子，做网站，其实跟做菜一样。

1. 先看菜谱，看需要哪些配料，做网站，当然需要html,css,js,java,mysql这些菜，当然还需要设计模式、面向对象、框架这些调料，eclipse这样的刀，另外还需要tomcat这样的锅。
2. 磨刀 基本工具要首先用熟，快捷键都记住了吧。
3. 洗菜 主菜要多花些功夫，配菜只需要一点。html，javal算作前后端主菜，要反复洗脸。css，js，mysql算作配菜，稍作处理即可。
4. 放料 菜好了，料也不能少，配料要少而精，什么火候放都是学问啊。
5. 开炒 准备工作都ok了，放入锅中，一起翻炒。翻炒时要不断的研究，仔细的观察，弄懂主次搭配，调料的作用，前后贯通。
6. 好菜出锅，如果不行就多来几次，一盘好菜就ok了。

学习其他综合性技能也是类似的思路，先根据菜谱，分别准备，有主有次，从前到后分析流程原理，熟能生巧。

[架构师速成6-初中](http://blog.csdn.net/for5million/article/details/46572965)

* 阶段： 中学
* 学时：6-12个月
* 升学标准
  + 学会如何思考，读每本书都能整理思维导图，能使用思维导图思考。
  + 学会知识整理，使用知识管理工具整理自己的知识。
  + 掌握设计模式，可以设计一套开发框架，使用此框架开发一个产品，而且新人也可以快速使用此框架进行开发。
  + 掌握软件设计理念，有一套自己的设计、开发思路，并使用此思路完整的完成一个产品设计。
  + 精通linux

中学自然要学习的东西更多，不知道你能不能坚持下来，如果不能坚持，请放弃这个职业。

思维导图又一个神器，好好掌握，后面单独一个篇幅讲解。

知识管理工具，wiz(为知)，evernote等，后面单独一个篇幅讲解。

linux 作为一个牛逼开发人员，不得不面对。

小学你已经学会如何从0搭建一个简单网站，那么你已经可以找到一份工作。找到工作后，你肯定会被安排做更大的网站。

这时候你应该沉浸在学新知识的快乐中，抓紧一切时间学习你身边可以学到的东西。同时你需要思考，如果从零开始做你目前的这个工作，应该怎么做？你会想到下面这些：

* 需要人员管理
* 需要了解需求
* 需要制定计划
* 需要进行设计
* 需要实现
* 需要测试
* 需要发布
* 需要收集反馈

或者更多的事情。

中学其实就希望你能学会整个软件的开发过程，形成自己的一套设计、开发思路。另外有思路还是不够的，希望你能沉下心把做的事情简化掉，对已有的轮子进行封装，让你从0开始做一个网站时易如反掌

# 架构师速成6.1-思维导图

思维导图者，所谓神器也！

先推荐思维导图的工具，xmind，这个是免费的。如果你是有钱人，可以用mindmananger。是不是给点推广费啊，家里揭不开锅了！

文章我觉得还是不要去重复造轮子了：

http://www.read.org.cn/html/542-use-mind-mapping-to-help-you-study-and-pass-the-exam-express-1.html

http://www.read.org.cn/html/544-use-mind-mapping-to-help-you-study-and-pass-the-exam-express-2.html

另外看看

http://www.read.org.cn/html/category/mandmap 这一类的知识之后，我就不需要再讲什么了。

将思维导图和前面的费曼学习法结合起来。

# 架构师速成6.2-知识什么时候才是你的？

很多人上了很多年学，读了很多年书，他们其中的很多人压根就没有用上这些知识，我就是其中的一员。当时我还是化学课代表，化学牛人，高三的化学卷，别人做1个半小时，我只需要半小时，而且顶多会错2个选择。然而并没有什么卵用，我发现上了大学之后，化学、物理统统交回去了。所以学过!=你的知识。

另外我还看过很多书，然而并没有什么卵用，都忘了。所以看过!=你的知识。

参加工作以来，我用过的语言也有不下10种，然而并没有什么卵用，现在很多都忘了。所以用过!=你的知识。

那什么才算是你的知识呢？就像吃饭一样，你吃了很多，但是最终留下的才是你的，其他的都变成了大便。

你总结的思想，你掌握的技能，你可以讲授的东西，才是你的知识，才是你的价值。

前面的费曼学习法就是很好的总结自己知识的方法，思维导图是进行总结归纳的好工具，另外你要有自己的知识管理工具，类似“为知”之类。

知识要挑肥拣瘦，不要全盘接收。打个比方，你去吃自助餐，海鲜，肉类，菜类，甜点，饮料，水果应有尽有。有人就上来喝了2杯啤酒，吃了一个西瓜，后来只能干瞪眼看别人吃，自己喜欢吃的吃不下了。有人喜欢吃肉，来了就吃肉，吃的很尽兴，有人喜欢吃海鲜，就主攻海鲜，吃的也很happy。

* 要有自己的目标，知道自己需要什么知识，专门去整理和学习这些知识。

还有人猛吃便宜的，但是很占肚子的，比如来了就啃面包，那还不如在家自己啃面包，何必来吃自助。

* 对于字典型知识，只需要知道，什么时候会用到他，我该如何去查就ok了，而不需要记住。不要因为字典型知识的学习占用过多的时间。

吃完了，要总结归纳，下次来怎么吃才爽，分享给后来人。

* 比如你爱吃肉，随着你多次吃自助，总结出规律，先吃牛肉，再吃五花，再吃鸡翅比较过瘾。总结下来，归纳下来，顺便给同样是吃货的朋友分享，碰撞。画一个吃自助攻略思维导图，或许更好。

不断的跟其他的饭店进行比较和串联，吃自助优势在哪，吃大酒店怎么样，一周7天该怎么吃。

* 把你的知识跟其他知识串联起来，说不定就可以出一本《吃遍中国》

一个喜欢的店，要多去才能显示你的真爱。

* 要不断的回顾和反思你总结的知识，归纳整理分类。

我都觉得我像个吃货了，减肥去了，100个俯卧撑！

# 架构师速成6.3-设计开发思路

面向对象，是一个伟大的设计思想，应该是软件开发史上的一次革命。

当然理解面向对象也很难，有好多人用着面向对象的语言，写着面向过程的逻辑，而且一写就是好多年。但是有高手，用c照样可以写出很牛的面向对象的程序。面向对象其实是一种思考问题的方式，重点如下：

1. 面向对象是用来反映显示世界的，而不是强行创造世界。
   * 这句话，说起来简单，但是做起来很难。现实世界中你绝对不会把狗腿，按在一个人身上，但是写程序的时候，你常常会创造出一个狗腿人。
   * 有人还会创造一些一些稀奇古怪的万能类，或者融合了n种物种的怪物。或者只有一条腿的狗。
   * 一定要记住，只有反映显示的才是长久的。
2. 是我的就是我的，不是我的就不是，不增不减。
   * 有时候有些属性，或者方法，不知道该如何放就随便放在一个类里，这就大错特错。
3. 真正的对象是有血有肉的，而不是只有一堆属性，或者只有一堆方法的怪物。
4. 按照现实世界的关系安排对象之间的交互。

怎样锻炼你的面向对象思考能力，很简单，每天把和你产生交互的人和物，写成代码，让他们和你的交互变成方法的调用。例如今天你坐车去了超市，买了一堆东西。抽象出 car，person，market，goods，cash 等等，用代码实现他。

设计模式的确是很好的，它们是前人给我们总结了的一些秘诀，这是国人最喜欢的了，对吧，所以记住什么时候使用他们。

有了上面的知识，就可以思考如何有一套自己的设计开发思路了。做任何事，都有一套最适合自己的方法，你要试着摸索出来，并固化到你的血液里。比如现在给你一个项目，做一套简单的进销存系统，需要你设计出来，并进行编码，你会怎么办？

想一想？有很多种方式去进行设计，但是你总得有你的一套，所以你自己总结吧，我不会把我的秘诀告诉你的。

# 架构师速成6.4-开发框架

经历了很多公司，看过好多代码。传统行业公司普遍都有自己一套统一的开发框架，封装的非常傻瓜化，门槛极低，便于不同的项目快速开发上线。比如有一个银行业框架封装到开发人员只需要在界面上拖拽就完成大部分工作，然后在拖拽好的模块里面添加一些业务代码就ok了。用友，华为也是类似。当然传统行业缺点是，一套框架用n年，老掉牙了也在用，有时候跟不上时代。

小的互联网公司有时候就不太注重，基本就以快速上线为主，草草完成功能就可以了。阿里巴巴在开发框架封装方面做的也不是很好，跟发展历史有关，也跟商业驱动有关，这个就不多谈了。

开发框架一说，有做java就认为spring就ok了，或者spring我简单封装一下就ok了。大错特错，做框架的目的是把与业务无关的所有细节隐藏掉，让开发人员以最快最简单的方式完成业务。拿做一个网站来说，抛开架构，仅就单机开发一个java网站，需要做的事情就很多。

我觉得起码需要规范或封装的部分：

1. 后端
   * 后端编码规范
   * 分层规范 controller service dao。这个非常重要
   * 基础工具类 xxUtil
   * Ioc aop spring可以实现
   * mvc spring可以实现
   * 数据库 数据源 事务 这个可以通过spring实现
   * 日志记录 通过spring aop记录耗时日志，参数日志等
   * 模板封装 jsp或vm等
   * 数据校验
2. 前端
   * 前端js/css编码规范
   * 前端统一字体、样式、控件规范
   * 前端模块化开发 例如:sea.js
   * form 校验
   * ajax
   * 无js化，参见angularjs jui(国内前端框架)
3. 前后端结合
   * 分页
   * 异常处理及展示
   * 页面跳转规范或ajax load规范
4. 业务
   * 单点登录
   * 权限判定
   * 评论
   * 等等

总之有很多，一定需要封装到让一个刚毕业的学生在2-3周就可以熟练开发。而且框架代码由专门的框架开发人员进行升级维护及培训，保持框架的活力。

封装框架可以增强对设计模式及面向对象，aop的理解，只有当你成功的完成了一个框架的封装，你才可以继续往架构的方向前进。

当然对此有开发人员颇有争议，我见过很多程序员，当他从一个公司离开后，发现脱离了公司的框架自己什么也不会。这个说明2个问题：

1. 他们公司框架做的好，让他了解的细节足够少。
2. 他没有合理的利用时间，去把框架掌握到手。

跟阿里的同事也有讨论，结果大家都比较认同：

1. 一个团队必须有自己的开发框架，虽然再去为阿里全集团做一套统一的框架很难，但自己团队内部必须有一套框架，可以帮助团队快速开发。其实支付宝是有一整套开发框架的，这是因为支付宝沿袭自金融行业。
2. 另外要组织全团队的框架培训，让每个人都对框架有足够的熟悉，甚至提出改进意见。
3. 通过框架节省下的时间，可以研究更先进的技术，反哺框架本身。

# 架构师速成6.5-也谈设计模式

回头来回顾之前写的文章，发现初中阶段的内容缺了一块很重要的内容就是设计模式。设计模式是程序员的另一个G点，无论吹牛，还是面试，还是设计都会秀一下设计模式，这也奠定了设计模式不可动摇的地位。

我主要讲一下设计模式如何学习，哪些虚的我就不讲了。

1. 找2本书就可以了，大话设计模式和head first设计模式
2. 速读2遍，对设计模式有总体的概念
3. 整理思维导图，先把印象最深的5个列下来，用你的语言通俗易懂的给别人讲清楚，另外需要重点侧重于什么场景下使用。
   * 举个例子，代理模式，怎么跟人讲呢，http://yangguangfu.iteye.com/blog/815787 看看这篇文章，很黄很暴力
   * 当然你还要侧重一下怎样用，在需要用比较通用和复杂的对象指针代替简单的指针的时候，使用 Proxy模式。下面是一些可以使用Proxy模式常见情况：
   * 远程代理（Remote  Proxy）为一个位于不同的地址空间的对象提供一个本地的代理对象。这个不同的地址空间可以是在同一台主机中，也可是在另一台主机中，远程代理又叫做大使(Ambassador)
   * 虚拟代理（Virtual Proxy）根据需要创建开销很大的对象。如果需要创建一个资源消耗较大的对象，先创建一个消耗相对较小的对象来表示，真实对象只在需要时才会被真正创建。
   * 保护代理（Protection Proxy）控制对原始对象的访问。保护代理用于对象应该有不同的访问权限的时候。 智能指引（Smart Reference）取代了简单的指针，它在访问对象时执行一些附加操作。
   * Copy-on-Write代理：它是虚拟代理的一种，把复制（克隆）操作延迟到只有在客户端真正需要时才执行。一般来说，对象的深克隆是一个开销较大的操作，Copy-on-Write代理可以让这个操作延迟，只有对象被用到的时候才被克隆。
4. 直到你讲的一个不懂开发的人也很清楚的时候，开始整理下5个。
5. 整理完所有的之后，理清所有的模式之间的联系和区别，例如
   * 适配器Adapter 为它所适配的对象提供了一个不同的接口。相反，代理提供了与它的实体相同的接口。然而，用于访问保护的代理可能会拒绝执行实体会执行的操作，因此，它的接口实际上可能只是实体接口的一个子集。
   * 装饰器模式Decorator：尽管Decorator的实现部分与代理相似，但Decorator的目的不一样。Decorator为对象添加一个或多个功能，而代理则控制对对象的访问。
6. 整理到1个思维导图上，做到只要看到这份思维导图，你就能讲出所有。
7. 试着重新画这副思维导图，多画几次，直到你很容易的就能画出。
8. 尝试用代码描述你日常见到的所有可以使用设计模式的地方，每天做1个例子。

ok 不出半个月你已经是设计模式高手了。请让我膜拜一下奥！

# 架构师速成6.6-知识的收集整理学习

知识如何学习前面已经讲了2节，这节主要讲知识的整理和沉淀。

其实我之前也一直没有好好的思考过这个问题，今天在整理自己的wiz知识库的时候突发灵感，所以有了这一节。

其实知识获取的过程分为搜索->收集->整理->精化->应用->分享，前一部分跟时间管理的收集也很相近吧。知识获取的思路适用于有目的的知识收集和日常的备忘性的知识收集。当然你随机收集一些资料记录下来其实效果并不是很理想，重要的是你要有目的的学习才能最大的发挥你的心智以及潜意识。当你主动要学习一项知识时，你的潜意识会主动关注一切相关数据和知识，举个例子，当你要学架构师相关知识时，当你在随便浏览网页时，如果出现架构师这几个字时，你会一眼就发现，而且会特别关注，这就是潜意识。当然今天不是讲潜意识，而且你的主动意识。

我说一下我是怎么进行管理的，其实我发现需要2套工具来进行知识管理，1是wiz类知识收集工具，2个云盘用来存放大的文件，如pdf，视频等等。但是2者的目录结构是完全一致的，这样便于管理和分享。

先抛砖一下我的目录结构

├─01收集箱  
│ ├─团队分享  
│ └─股票  
├─02有效资料  
├─03我要学习  
│ ├─01马上学习  
│ └─02即将学习  
├─04我的能力  
│ ├─01学习能力  
│ │ ├─01阅读技巧  
│ │ ├─02记忆力  
│ │ └─03学习方法  
│ ├─02思考能力  
│ ├─03沟通能力  
│ └─04管理能力  
│   ├─01时间管理  
│   ├─02知识管理  
│   └─03财务管理  
├─05我的知识  
│ ├─01家庭生活  
│ ├─02健康养生  
│ └─03工作必备

我说一下，如果我要学一门新的知识我会怎么办，比如要学习scala吧，我会先搜集入门资料：

1. 去搜相关的论坛，豆瓣，收集资料，有很好的经验、心得就用wiz抓下来，放到收集箱
2. 确定要看的书，确定好之后下载电子版，电子版放到云盘里面，如果没有电子版就买纸质的。
3. 确定相关的好的项目，比如twitter的util库，akka等，在看完书之后实践用。也下载

整理资料：

1. 在我要学习–马上学习目录里建立scala入门，scala进阶的文件夹，把相关pdf，代码放到里面
2. 在wiz的相同目录，将抓取得文章，放到scala入门，scala进阶目录。

开始学习：

1. 先快速浏览一下所有的入门文章，在我的知识–工作必备-scala 建立目录，将吸收的知识转化为自己的语言整理到里面。当然看懂的资料先放一下，等后面再回头看。
2. 速读相关书籍，整理xmind读书心得
3. 进行练习，整理练习心得，所有的都沉淀下来，变成自己的。
4. 反复进行

整理自己的知识&分享：

1. 学一段时间后，回顾自己整理的心得，知识，做进一步总结概况，并能给别人讲解，放到网站上或者论坛上。
2. 把已经完全吸收的文章或者书籍，进行删除，仍有价值或者可以备查的资料，放到我的知识–工作必备-scala-归档目录进行保存。

大概就是这样了，肚子饿了，赶紧洗洗睡了。

# 架构师速成6.7-设计开发思路-uml

uml是什么东西？统一建模语言，一门语言，是用来进行软件设计的一门语言。

其实一门语言的诞生并不伟大，让大多数人都使用才足够伟大。uml就是一门伟大的语言，因为目前软件设计的唯一语言就是它。

UML其实还是比较简单的，就那么几个图形，那么几种模式，但是因为他是唯一的语言，所以有设计能力的人都能很容易看懂你说的什么，这就是他的伟大之处。

我说一下在软件设计中最常用的几个，以及我的心得：

1. 用例图，在了解用户需求时非常有效，他仅用来描述系统需要提供的功能，本身没有顺序，不要用来描述流程。注意使用扩展和包含。那个小人即可以是使用者也可以是其他系统。
2. 类图，这是面向对象设计的真谛，不要和ER图混为一谈，类图是用来描述类与类之间的交互关系，本身可以没有任何属性。当然也可以有很多属性，但是不要用设计数据库的思路来设计类图。类图只是用来反映现实，在设计类图时，可以认为数据会存储在DB中，也可能存储在XML中，也可以存储在文件中，不要去考虑存储。
3. 对象图，用的不太多
4. 序列图，描述对象之间的交互顺序，着重体现对象间消息传递的时间顺序，这个比较有用，但是不是很难。
5. 状态图，状态机就是它了，当你被复杂的状态搞晕的时候，用它来画清楚，实现就用状态模式，perfect。
6. 活动图，表示两个或多个对象之间在处理某个活动时的过程控制流程，这个也很重要，但是不难。

其他我用的就不多了，学习这门语言真的很重要，请重视。掌握他之后，学习设计模式会更加得心应手！

# 架构师速成6.8-设计开发思路-领域驱动（转帖）

领域驱动设计简称DDD,很好的名字，先来普及一下相关的名词缩写：

测试驱动设计 TDD，行为驱动设计 BDD，面向对象设计 OOD，面向过程设计 OPD。

设计思路和方法是一项专门的技能，区别于设计模式，编程语言。UML是设计的工具，设计方法是设计的灵魂，而且设计方法并没有好坏之分。关键是你需要掌握各种设计方法，在做项目时信手拈来，才是真正的高手。为什么要讲领域驱动设计，因为在做大型系统时，领域驱动设计会让你事半功倍，得心应手。

网上找到一篇总结的文章，写到比我写的好，那就直接转帖吧http://www.cnblogs.com/netfocus/p/4492486.html，原帖地址。

1. 领域驱动设计（DDD）是一种基于模型驱动的软件设计方式。它以领域为核心，分析领域中的问题，通过建立一个领域模型来有效的解决领域中的核心的复杂问题。Eric Ivans为领域驱动设计提出了大量的最佳实践和经验技巧。只有对领域的不断深入认识，才能得到一个解决领域核心问题的领域模型。如果一个应用的复杂性不是在技术方面的，而是在领域本身，即领域内的业务很复杂，那这种应用，使用领域驱动设计的价值就越大。
2. 领域驱动开发也是一种敏捷开发过程（极限编程，XP），强调迭代开发。在迭代过程中，强调开发人员与领域专家需要保持密切的合作关系。极限编程假设我们能通过不断快速重构完善设计。所以，对开发人员的要求非常高。
3. 领域驱动设计提出了一套核心构造块（Building Blocks，如聚合、实体、值对象、领域服务、领域工厂、仓储、领域事件，等）,这些构造块是对面向对象领域建模的一些核心最佳实践的浓缩。这些构造块可以使得我们的设计更加标准、有序。
4. 统一语言（Ubiquitous Language），是领域驱动设计中一个非常重要的概念。任何一个领域驱动设计的项目，都需要一种通用语言，一套通用的词汇。因为没有通用的语言，就没有一致的概念，沟通就会遇到障碍，最后的领域模型和软件也就无法满足领域内的真实业务需求。通用语言是领域专家和开发人员在对领域问题的沟通、需求的讨论、开发计划的制定、领域模型的设计，以及开发人员之间对领域模型的具体编码落地实现，等一系列过程中，所有人员使用的一种通用语言。话句话说，就是无论是沟通时所用的词汇、还是领域模型中的概念、还是代码中出现的类名与方法，只要是相同的意思，那就应该使用相同的词汇。可以看出，这种通用语言不是一下子就可以形成，而是在一个各方人员讨论的过程中，不断发现、明确，与精炼出来的。
5. 领域模型是领域驱动设计的核心。统一语言中的所有关键词汇，在领域模型上应该都能找到。各方人员沟通时，都应该以领域模型为基础。通过讨论的不断深入，大家对领域的认识也会不断深入，领域模型也会不断得到完善，统一语言的词汇也会不断丰富和精准。需要特别强调的是，开发人员应该尽量保证代码实现和领域模型相绑定，时刻保持代码与模型的一致。如果不绑定，那代码就会慢慢和模型相脱节，就会出现像我们以前那样的设计文档和代码相脱节一样的问题，甚至模型还会起到误导作用。通过这样一种思路，我们确保语言、模型、代码三者紧密绑定，确保最后实现出来的软件可以准确无误的实现业务需求，并且还能让我们的软件可以快速的和业务同时演进。而不像传统的开发方式那样，分析、设计、实现三个阶段完全脱节，最后出来的软件没有很好的满足业务需求，也不能在未来很快的跟业务需求一起演进。所以，领域模型同时承载了分析的结果和设计的结果，这里的分析是指对领域内业务需求的分析，设计是指对模型的设计以及软件的设计。所以，我们的领域模型，不能只考虑业务需求，还要同时考虑软件设计的原则，是一种综合考虑的、平衡的设计结果。
6. 领域模型可以复用，因为特定的领域模型解决的都是某个特定的问题域；比如淘宝网有个商品中心，有个商品模型，核心概念有商品分类、商品；商品模型负责解决电子商务领域中的商品目录（Product Catalog）子域。后来阿里又出了个天猫，也会有商品中心，但是这两个商品中心基本是一样的问题域。所以，我们可以复用之前淘宝实现的商品中心领域模型，并复用之前淘宝商品中心的解决方案，来解决天猫的商品维护和展示。当然，这个只是我个人的认识，一个例子。具体阿里是否是一个商品中心同时解决淘宝和天猫的业务，没具体调研过。
7. Bounded Context，属于一种软件构件，作用是用来对领域模型进行划分。Bounded Context有两层含义：
   * Bounded，即有边界的，表示领域模型有边界；这个边界定义了模型的适用范围，以便让负责该模型的团队知道什么该在模型中实现，什么不该；
   * Context，即领域模型的产生是在某个上下文中产生的；上下文是一个和环境相关的概念。比如一次头脑风暴会议大家达成了一个模型，那这次会议的讨论就是该模型的上下文；比如某本书中谈到了某个东西，那这个东西的上下文就是那本书，那个东西要有意义的前提离不开那本书这个上下文；所以，上下文是模型有意义的前提；
8. 领域建模的方法有很多种，我分享一下自己的一种基于场景为核心的分析方法。大概的思路是：
   * 通过与领域专家和业务需求人员沟通，找出领域中的关键业务场景；
   * 针对每个业务场景分析出有哪些场景参与者，哪些参与者以对象（聚合）的形式参与，哪些参与者以服务的形式参与；
   * 分析每个场景参与者对象的基本状态特征；
   * 分析每个场景参与者对象分别扮演什么角色参与场景，整个场景的完整交互过程是怎样的，对象在参与场景的过程中执行了哪些交互行为；
   * 分析如何记录和跟踪这一次交互行为，分析这次交互行为会产生哪些额外的信息；
   * 上面，只是简单列了一下条目，具体的描述，请参看我的[另一篇文章](http://www.cnblogs.com/netfocus/archive/2011/09/18/2180656.html)，有详细的叙述。
9. 关于领域（Domain）、领域模型（Domain Model）、边界上下文（Bounded Context）的关系
   * 领域就是问题域，问题空间；
   * 领域模型是一种模型，表达了领域中哪些业务需求以及业务规则必须被满足；
   * 每一个领域中的问题，都会有一个对应的领域模型去解决；
   * Bounded Context的作用是用来对领域模型进行划分；
   * 划分领域就是对问题空间的划分，通俗的理解，就是将大问题拆分为小问题；
   * 划分Bounded Context就是将一个大的领域模型划分为多个小的领域模型；
   * 可以把Bounded Context看成是一种解决方案空间，所以，Bounded Context也可以理解为是对解决方案空间的划分；
   * 理论上，一个Domain可能会对应多个Bounded Context；同样，一个Bounded Context可能也会对应多个Domain；所以他们之间没有绝对的关系。主要是他们划分的依据不同，一个是针对领域（问题空间），一个是针对领域模型（解决方案空间）；理想情况，一个Domain最好对应一个Bounded Context；
10. 关于Domain、Sub Domain、Core Domain、Generic Domain，以及Shared Kernal的理解：
    * 一个领域（Domain）会拆分为多个子领域（Sub Domain）；
    * 子领域中最核心（最重要）的那个叫Core Domain；我们应该讲团队的核心资源用在核心子域上，因为它是产品成败的关键；
    * 除了Core Domain外，其他的是支撑子域（Supporting Subdomain）；
    * 有些支撑子域比较特殊，因为它解决的是一类通用问题，比如账号和权限；这类子域我们叫做通用子域（Generic Subdomain）；通常，通用子域对应的Bounded Context，会跨域多个子域；
    * 多个子领域有时会有相交的部分，我们称作共享内核（Shared Kernel）；体现到代码上，就是同一份代码，在两个领域模型中复用；
    * 一般只有Domain比较大的时候，我们才会划分出Sub Domain；
11. 为什么一个大的领域模型需要划分？因为，通常一个大的领域模型需要多个团队合作完成。如果多个团队基于一个共同的领域模型工作，由于每个团队的关注点不同，且一些看似叫法一样的概念，对于不同的团队，其背后的意思完全不同。所以，这样的概念含义模糊会给团队以及成员之间的合作带来很大的困扰。所以，我们需要通过一种手段（Bounded Context），将领域模型划分为不同的部分，确保同一个Bounded Context内的领域模型所表达的概念含义明确。然后，同一个Bounded Context下面，相关人员都使用一种统一的语言，以此来保证团队成员之间沟通能畅通无阻；

当然，如果有时间还是看看《领域驱动设计》这本书吧。

[架构师速成8.1-谈做技术人员的态度](http://blog.csdn.net/for5million/article/details/46938677)

谦卑，永远保持谦卑。

热情，永远保持热情。

同行相轻在中国好像是一个惯例，互相看不起，互相贬低，也充斥了软件行业的每个角落。想成为一个架构师请先保持你的内心的谦卑，永远不要嘲笑或贬低任何一个人。因为当你嘲笑或者贬低一个人时，世界就给你关了一扇门。其实群体的意志才是最强大的，当你有一个团队，一般情况下团队的力量总是会比一个人要强大。所以要时刻想着如何激发整个团队的热情，而不是轻视你的队友，当你谦卑时，你就会拥有整个团队的力量。同行相轻这些事情在大公司的同学身上尤为明显，大公司的同学会不自然的体现一种优越感，可能本人无法察觉，尤其是在面对其他小公司的同学时。但是我见过真正的牛人，他们反而是最没有架子，最谦卑、最低调的人。这也说明为什么牛人能做到那样的地步，先做人再做事，古人诚不欺我。

热情，是前进的动力，是团队的火种。保持对新技术的热情，保持对知识的热情，保持对同伴的热情，用你的热情感染周围的一切，世界终将因你而变。

怎么一股浓浓的鸡汤味！这段时间晚上没有饭吃的原因吧，已经2周多没有吃晚饭，减了6斤了，请祝贺我。

[架构师速成8.2-架构师要懂产品](http://blog.csdn.net/for5million/article/details/46938679)

产品和架构两个截然不同的职业，好像风马牛不相及，其实不是这样的。产品的思想需要经过技术的手来成为现实，在成为现实之前，需要技术理解、评估、碰撞、优化、把控、验证等等。当然架构师就承担了这一系列技术的责任，而且在一个产品的实现过程中，技术架构并不是很重要的，前期可以没有架构，简单快速验证，只有在用户多了之后，架构才有真正的用处。在初创公司，很多架构师都等不到用户多了的那一天，来实现自己的架构梦。所以产品这一关架构一定要把好，只有你把好了，后面才有机会让你去架构。

当然架构师的懂产品，是懂产品的生命周期，产品发展的客观规律，产品的评估及验证。其实架构师承担的是一个美食家的角色，美食家并不需要去做菜，但是要知道如何评价菜的味道，懂得菜是如何做出来的。你不需要去养猪，只管吃肉就行了，听上去很美好，非常美好！

1. 产品提出初步思路，这时候你就需要参与进来，用客观的角度去评价思路的正确性，而不是觉得这个做产品的人跟我很熟，我觉得产品思路就不错。这个非常重要，目标错了，200%的努力去实现，只会更糟。如何用客观的眼光去评价呢，我有一个诀窍，就是代入法，假设这个产品是你的竞争对手做的，你如何干掉他。
2. 产品需求调研，你要亲自参加，使用精益创业的手段来进行理性的验证。

后面不写了，再写可以专门写一本了，反正就是你要懂产品，不是让你去做原型，而是让你去品这个产品！

# 架构师速成8.3-可用性

作为一个软件系统可用性是第一位的，如果一个系统不可用，你其他的地方做的再怎么好，然并卵。

一般什么情况下软件会不可用：

我方发生故障，导致系统不可用，当然会出现单机的不可用及n多机器群的全部不可用。

1. 程序故障 功能错误、程序退出
2. 系统故障 CPU超负荷、内存超负荷、网络超负荷
3. 物理故障  机器死机 断电 断网
4. 不可恢复故障 地震、海啸等等

客户方也会发生相同故障，导致系统不可用，当然会出现个别用户的不可用及区域性用户均不可用。

对于我方发生的问题，我们必须通过架构的方式进行解决，对于客户方发生的问题，我们尽量找方法解决，先解决区域性问题，再解决个别用户问题。解决方案必须要考虑到成本及战略来进行取舍，比如创业初期，根本没有大量资金，要解决不可恢复故障基本不太可能。

我们先试图从架构的方式来解决我方发生的故障，这种解决方案类似于设计模式，故称之为架构模式。

针对单机的不可用，有一个专业术语叫做单点故障，最好的方式就是部署多机器，通过多机器负载均衡，来规避单点故障。

1. 分布式
2. 负载均衡

针对多机的不可用，我们需要分类看如何解决：

1. 程序故障 功能错误、程序退出，这种错误有同学说，可以加单元测试、功能测试，让测试来发现问题。是的，但是那是开发流程，我们先不讨论那个，我们从架构的角度讨论，主要的解决方案如下：
   * 分批自动化发布
   * 灰度发布
   * 异常监控
2. 系统故障 CPU超负荷、内存超负荷、网络超负荷
   * 流量控制
   * 功能降级
   * 动态扩容
   * 异常监控
3. 物理故障  机器死机 断电 断网
   * 异地多活
   * 异地热备or冷备
   * 异地数据同步
4. 不可恢复故障 地震、海啸等等
   * 同上

后面我会针对每个专题跟大家仔细讲解。

# 架构师速成8.3-可用性之分布式

分布式算是软件界发展的一个里程碑，它开辟一个新的软件时代，其他的溢美之词我就不再乱说了。

分布式按照我的观点，应该分为有状态和无状态2种：

1. 有状态
   * 分布式数据库
   * 分布式存储
2. 无状态
   * 分布式计算
   * 分布式web系统

当然分布式尽量做成无状态的分布式，但是存储最终因为最终存储的是有状态的数据，所以不得不变的有状态。当然web系统也可以是有状态的，但是最好做成无状态的，因为无状态可扩展性更强，而有状态必须维护和确定状态和机器的对应关系，无谓增加了很多复杂性。例如：

无状态的集群，我随意增加或减少一台机器，对原有机器无任何影响。有状态的，我就需要知道新增或减少的机器，应该分配哪些数据哪些数据。

我们先从简单的无状态分布式系统说起，分布式计算我就不再展开了，针对web系统如何架构为无状态讲解一下其中的关键：

1. web系统有状态的原因是什么？如果用户都不需要登录，系统本身就是无状态的，所有人的看到都是一致的。但是用户登录之后，你要一直显示这个登录用户的相关数据，导致了系统变为有状态。
2. 如何解决这种有状态呢？如果这种状态可以作为数据每次通信时传递给web系统，那web系统就可以变为无状态。这也是常见的有状态变为无状态系统的解决方案，例如web系统将有状态的数据存储在db。解决方案很简单，就是cookie记录用户id。有人说用户登录之后有好多数据，我cookie存不下，而且有安全问题，那就加一个无状态的cache集群，存储所有的用户数据。cookie的id来了之后就去cache拿一下所需的数据，针对这个有个专业的名词叫做 share nothing architecture，sna架构。
3. 有了无状态的web系统，如果请求量增加，很easy，加机器就可以了。另外有些同学说加的机器怎样感知到，如果有机器宕机，又是如何感知到的，这个是负载均衡干的事情，后面会单独再讲。

在阿里巴巴有一个专门的架构方案来解决web架构的有状态，叫做tbsession。

无状态的讲完了，那再讲讲有状态的分布式，这个可能需要的篇幅比较多了，让我再单开一章吧。

刚才查了一下，果真有人已经提出了有状态和无状态，呵呵，看来还是有人更专业。

# 架构师速成8.3-可用性之分库分表

有状态分布式，涉及的知识就比较多了，不过我们可以拿几个现实的例子由浅入深的来理解。

## 数据库的分库分表

1. 假设你是一个开发负责人，开始使用单机的数据库，突然一天数据库硬盘挂掉了。你没有做备份，然后就没有然后了。
2. 进入第2个公司，你意识到备份的重要性，每天定时备份到另一台机器，突然有一天，数据库硬盘挂掉了。你心想幸好我有备份，然后巴拉巴拉的恢复起来，用了2个小时。老板说不错，但是—-我们因为宕机造成大量用户流失，信誉下降，然后就又没有然后了。上面说的就是单点的问题。
3. 进入第3个公司，你觉得单点很可怕，所以主备做起来，数据自动同步到备库，做到随时准备切换。突然有一天，主数据库硬盘挂掉了，你从容的修改数据库连接指向备库，重启系统恢复了，只用了5分钟。此时掌声一片，你沉浸在无比的欢乐中，老板说不错，但是—就在这5分钟我们丢了一个上亿的单子。我擦，你不是故意的吧！（其实这有可能是真实的片段，我们创业时，就30分钟断网，结果正好在举行一个大型的营销策划，不说了，我擦一会眼泪），然后就又没有然后了。其实当你用上主备时，说明数据库已经有状态了，必须要区分谁是主，谁是备。
4. 进入第4个公司，你不但做了主备，还做了高可用，通过HA实现了瞬时切换。突然有一天，主数据库硬盘挂掉了，你从容的端起了你的屌丝杯，世界清静了。老板说不错，小子我看好你。从此你走向人生巅峰，出任CTO，迎娶白富美。但是没过多久问题来了，随着用户不断的增加，你的数据库摇摇欲坠，不时就抽疯。老板说搞定他，不然我就搞定你。
5. 咋办，分库分表啊！如何分，这就涉及到更多的规则了，比如按照用户id是最常见的做法。此时你不但需要管主备而且还需要在程序中确定如何路由，结果集合并，如果再有机器增加，还要涉及数据迁移，另外还要防止出现重复id的脏数据，需要全局唯一主键，等等。亚美蝶！知道有状态的痛苦了吧。这也是为什么有些同学转投nosql的存储的很大原因，nosql替你屏蔽了这些规则，他在内部实现了路由、分库、合并等等。
6. 提到这里不得不提一下淘宝的牛逼产品–drds（沈公子是不是应该给些广告费啊）。
   * 分布式SQL引擎
     + 将数据按照条件分散到多个数据节点(分库分表)，对于数据操作sql进行分布式优化，获得最佳执行效率
   * 自主运维
     + DRDS的用户运维平台提供DRDS接入、分布式DDL、拆分信息维护、平滑扩缩容、分布式DML、监控等常用功能，让运维工作变得更简单
   * 小表复制
     + 对于配置表，常量表等不经常变化的表进行多节点对等同步，加速该类表与其他拆分表做关联查询的速度
   * 分布式全局唯一id
     + 提供全局唯一数字id服务，帮助您在分布式环境下，继续保持类似唯一键、主键等数据的全局(所有节点)唯一性
7. 看到了吧，这就是有状态带来的痛苦。为了把有状态变为无状态有时候你需要做大量的工作。

有关分库分表的关键点和难点，我新一章统一讲解。

# 架构师速成8.3-架构师必须要了解的规则（转）

作为一个架构师，有些规则是必须要掌握的，这就想软件的公理，如果你学物理不知道牛顿定律，那就不要学了。在软件行业也有类似的东西，我称之为软件定律。例如：

ACID,CAP,BASE

## ****ACID****

传统数据库系统中，事务具有ACID 4个属性

(1)原子性（Atomicity）：事务是一个原子操作单元，其对数据的修改，要么全都执行，要么全都不执行。

(2)一致性（Consistent）：在事务开始和完成时，数据都必须保持一致状态。这意味着所有相关的数据规则都必须应用于事务的修改，以保持数据的完整性；事务结束时，所有的内部数据结构（如B树索引或双向链表）也都必须是正确的。

(3)隔离性（Isolation）：数据库系统提供一定的隔离机制，保证事务在不受外部并发操作影响的“独立”环境执行。这意味着事务处理过程中的中间状态对外部是不可见的，反之亦然。

(4)持久性（Durable）：事务完成之后，它对于数据的修改是永久性的，即使出现系统故障也能够保持。

可以说，数据库系统是伴随着金融业的需求而快速发展起来。对于金融业，可用性和性能都不是最重要的，而一致性是最重要的，用户可以容忍系统故障而停止服务，但绝不能容忍帐户上的钱无故减少,而强一致性的事务是这一切的根本保证。

## ****CAP****

在2000的PODC（Principles of Distributed Computing）会议上，Brewer提出了著名的CAP理论。CAP指的是：Consistency、Availability和Partition Tolerance。

（1）Consistency（一致性）：一致性是说数据的原子性，这种原子性在经典的数据库中是通过事务来保证的，当事务完成时，无论其是成功还是回滚，数据都会处于一致的状态。在分布式环境中，一致性是说多个节点的数据是否一致。

（2）Availability（可用性）：可用性是说服务能一直保证是可用的状态，当用户发出一个请求，服务能在有限时间内返回结果。

（3）Partition Tolerance（分区容错性）：Partition是指网络的分区。可以这样理解，一般来说，关键的数据和服务都会位于不同的IDC。

CAP理论告诉我们，一个分布式系统不可能同时满足一致性，可用性和分区容错性这三个需求，三个要素中最多只能同时满足两点。三者不可兼顾，此所谓鱼与熊掌不可兼得也！而对于分布式数据系统而言，分区容错性是基本要求，否则就不称其为分布式系统了。因此架构设计师不要把精力浪费在设计如何能同时满足三者的完美分布式系统上，而是应该进行权衡取舍。这也意味着分布式系统的设计过程，也就是根据业务特点在C（一致性）和A（可用性）之间寻求平衡的过程，要求架构师真正理解系统需求，把握业务特点。

后来：CAP理论的作者终于给了我长久以来想要的答案：CAP理论并非严格的三选二，大多数情况下，A和C是可以兼得的，因为大多数情况下，P都不存在。  
P只有在结点之间通信延迟大于可接受的范围时才出现（结点之间开始近似隔离，状态开始不一致），即P一旦出现，我们选择继续提供服务那么状态就肯定不一致，也就等于放弃了C；我们选择不提供服务，那么就等于放弃了A。通俗一点，P并不是目标，也不是手段，它是伴随着“多结点，网络，数据，共享”的要求而必然出现的，出现的原因是因为网络的不可靠性及结点通信延迟（延迟的原因可能是由于硬件，网络，或者压力太大而无法及时响应)。

弄清楚了CAP的P，也就弄清楚了CAP理论的实质，戴在头顶的紧箍咒便永久摘掉了。

CAP理论并不是要求我们悲观地放弃A和C任何一方，相反，它可以乐观地指导我们将C和A最大化；ACID和BASE分别处于CAP理论的两个极端，ACID强调强一致性，BASE强调高可用性，两者把重点都放在A和C上，淡化了P也可变的事实；通过对P的出现检测，发现P之后的限制和约束，P结束之后的补偿和恢复，通过采用千差万别的策略，我们可以避免P带来的C和A的严重损失，实现A和C的最大化来提高整个系统的正确性和可用性。ACID和BASE并非水火不容，我们可以在同一个系统中，既使用ACID，又使用BASE。

## ****BASE****

BASE来自于互联网的电子商务领域的实践，它是基于CAP理论逐步演化而来，核心思想是即便不能达到强一致性(Strong consistency)，但可以根据应用特点采用适当的方式来达到最终一致性(Eventual consistency)的效果。BASE是Basically Available、Soft state、Eventually consistent三个词组的简写，是对CAP中C & A的延伸。BASE的含义：

（1）Basically Available：基本可用；

（2）Soft-state：软状态/柔性事务，即状态可以有一段时间的不同步；

（3）Eventual consistency：最终一致性；

BASE是反ACID的，它完全不同于ACID模型，牺牲强一致性，获得基本可用性和柔性可靠性并要求达到最终一致性。

后面我会不断充实这本软件定律。  
作者：[arrowcat](http://www.cnblogs.com/hustcat/)  
出处：<http://www.cnblogs.com/hustcat/>  
本文版权归作者和博客园共有，欢迎转载，但未经作者同意必须保留此段声明，且在文章页面明显位置给出原文连接，否则保留追究法律责任的权利。

本文为Gleasy原创文章，转载请指明引自[Gleasy团队博客](http://rdc.gleasy.com/)

# 架构师速成8.4-分库分表的关键点

我们还是由浅入深（这个词我喜欢，你呢？）的讨论一下，分库分表的关键点（本故事纯属虚构，仅为搞笑）：

1. 当你的系统很小的时候，只有一个数据库，每个表的主键都是自增的，你都不去关心主键变成了多少，反正db保证自增，小日子过的很是惬意。但惬意的日子总是短暂的，你因为DB宕机被老板fire 3次（见上一个故事）。
2. 进入第4个公司的时候，你发粪涂墙，将集群改成主备HA,结果顺利出任CTO，迎娶白富美，走向了人生巅峰。当然这中间也出过一些小插曲，比如：张三注册时，刚点击完注册，DB主机宕机了。张三发现刚注册的账号不能登录了，张三很生气。你说这这不算啥，让他重新注册一下账号吧。但是李四刚付款买了网站上一款价值9.98的玉镶金的超级玉佩，刚过1秒，订单就没有了。李四不愿意了，他还等着这个9.98的玉佩3天内升值500%大赚一笔呢。你灵机一动，我们写一个数据订正的程序吧，对于宕机丢失的数据进行订正，另外加送李四同学一块999的超级金树叶。问题都摆平了，你果真是维护世界和平的正义使者！
3. 由于只要9.98的超级玉佩口碑传播太好了，有无数人等着购买，甚至有人肯出更高的价购买，但是我们是有操守的，只卖9.98,9.98你买不了吃亏，买不了上当，现在购买还可以….此处省略1000字广告。于是网站的注册用户暴涨，数据库时不时卡死。
   * 老板发飙了：“怎么回事,我造福全人类的大业，要毁在你手里，你马上解决，要是解决不了，你就是人民公敌，社会败类！我会让无数大爷大妈一人一口唾沫吐死你！”。
   * 你：看来我们要分库存储了，一台数据库，完全抵挡不住大爷大妈们的热情啊，马上给点经费吧。
   * 老板对你一阵痛骂：“这得多少金镶玉啊，要不用金镶玉付款吧”。
   * 你：老板英明。
4. 又增加2台机器之后，突然发现你是的世界完全崩塌了。
   * 哪些数据需要分库呢？
   * 原来数据怎么分到这2台机器上呢?
   * 我查询的时候怎么知道查哪一套集群？
   * 自增主键太坑了，自增完都重复了，怎么办？
   * 原来的关联查询（我无数牛叉的sql），分库之后怎么办？
   * 我的事务一致性怎么办？
   * 原来的count，sum,group怎么办？
   * 要是需要再分我怎么办？但是你被fire3次之后，早已练成神功之————-死猪不怕开水烫，既然天降大任于我，我就全力去搞，顺便鄙视一下这个老板。
5. let‘s google，我靠，有专门的资料，http://blog.csdn.net/column/details/sharding.html，还有专门的书籍《MySQL性能调优与架构设计》，照猫画虎，大功告成。
   * 不明真相的群众:这样也太坑了，直接把最关键的略过了。退货，退货，揍他，揍他
   * 作者：我只是想告诉大家怎么思考，为什么会出现这些问题
   * 不明真相的群众：狡辩，揍他
   * 作者：饶命啊，大侠，我是觉得写起太费劲了。而且我也向大家展示了面向对象威力，这个分库分表就是一个完整的对象，我只需要引用他就可以了
   * 不明真相的群众：好像比较有道理，但是就是因为你敷衍，揍他
6. 搞好之后，流量大增，老板心里乐开了花，对你大加称赞，并奖励了你一块金镶玉。
   * 老板：小子，我看好你！这块金镶玉就送给你，不日就升值1000倍，你就是大富翁了。
   * 你：呵呵，谢谢老板，老板英明
   * 老板：看你嘴甜，再送你10块
   * 你：呵呵呵，证据到手
   * 你：警察局吗，有人在诈骗
   * 老板：亚美蝶……，我是在造福全人类
   * 你：对不起，我是卧底！

# 架构师速成8-本科

* 阶段： 本科
* 学时：6-12个月
* 升学标准
  + 掌握做一个成功的产品的理念，不需要画原型，但是思路要有。
  + 定制或开发一套自己的软件生命周期管理系统，进行整个生命周期的管理。
  + 开发一套基于现有框架的或理论的辅助开发系统，可以进行基本代码的生成及更新。
  + 掌握架构的基本模式，可以根据项目确定项目的架构选型，并完成架构实现及最佳实践案例。

工欲善其事，必先利其器。软件开发是一个系统性的工程，目前市面的产品，大都仅仅满足一个方向的需求，例如 redmine，hudson，maven等等。大公司都会定制自己的软件生命周期管理系统，比如淘宝的aone，就是淘宝自己定制的软件生命周期管理系统。

该学一学怎么做产品了，这是一门高深的学问，如果要深度掌握怕是很难。但是作为一个合格的架构师，必须要有产品理念和基本的做产品的思路。如何掌握做产品：

* 看一些做产品的书籍，人人都是产品经理、精益创业等等。
* 有机会就和用户沟通，而且经常站在用户的角度思考，评判做产品的人思路是否正确
* 和做产品的人沟通，学习他们的思考方式
* 思考曾经伟大或正在伟大的产品，为什么伟大，为什么消亡

架构师掌握了产品理念，是用来和做产品的同学进行碰撞及升华，用理性思维来验证产品的合理性。同上能够把握产品节奏，推进项目的进展。当然也可以在没有产品的时候，顶上去，完成一个产品的职能。

一个软件的完整生命周期，可以分为如下几个阶段：

* 构想
* 立项
* 需求&验证
* 产品原型&验证
* 架构
* 设计
* 开发
* 测试
* 自动化发布
* 产品验证
* 用户反馈
* 下线

如何做一个适合自己公司的系统，那就是你的事情了。

要有代码生成器，而不止于代码生成器。你懂了吧！

架构模式是一个很大的话题，还需要展开讨论。

# 架构师速成9-研究生

* 阶段： 研究生
* 学时：2-5个月
* 升学标准
  + 分析总结出各大网站架构的优缺点，分析其底层的实现机制，形成每个网站的分析文档
  + 实践各个架构模式，进行性能测试，进行深入分析，明确思想及原理。

他山之石可以攻玉

这句话看上去很牛b吧，但是我也不懂啥意思。

学习一下老外及国内网站的架构吧，分析一下他们的优劣，对自身提升很有帮助。顺便又能力分享一下心得。当然不要迷信所谓的大公司，其实一般情况下，是流量造就了他们的架构，如果你的网站也有那么多流量，可能你做的会更好。

实践是检验真理的唯一标准

实践一下你会懂的更多。

# 架构师速成10-博士

* 阶段： 博士
* 学时：6-8个月
* 升学标准
  + 根据自己实践确定合理的架构，开源自己的解决方案，获得认可。

予人玫瑰 手有余香

经过前面的练习，或许你已经成功，或者走在成功的路上。那希望你能分享你的心得，你的收获给更多的人。