# 学习

学习参考：<http://www.sohu.com/a/241412246_661203>

**1，架构师是什么？**

要想往架构师的方向发展首先要知道架构师是什么？架构师是一个既需要掌控整体又需要洞悉局部瓶颈并依据具体的业务场景给出解决方案的团队领导型人物。一个架构师得需要足够的想像力,能把各种目标需求进行不同维度的扩展，为目标客户提供更为全面的需求清单。

架构师在软件开发的整个过程中起着很重要的作用。

说的详细一些，架构师就是确认和评估系统需求，给出开发规范，搭建系统实现的核心构架，并澄清技术细节、扫清主要难点的技术人员。主要着眼于系统的“技术实现”。

**2，架构师的任务**

架构师的主要任务不是从事具体的软件程序的编写，而是从事更高层次的开发构架工作。他必须对开发技术非常了解，并且需要有良好的组织管理能力。可以这样说，一个架构师工作的好坏决定了整个软件开发项目的成败。

在成为Java架构师之前，应当先成为Java工程师。

熟练使用各种框架，并知道它们实现的原理。

jvm虚拟机原理、调优，懂得jvm能让你写出性能更好的代码；

池技术，什么对象池，连接池，线程池……Java反射技术，写框架必备的技术，遇到有严重的性能问题，替代方案java字节码技术；

nio，没什么好说的，值得注意的是"直接内存"的特点，使用场景；java多线程同步异步；java各种集合对象的实现原理，了解这些可以让你在解决问题时选择合适的数据结构，高效的解决问题，比如hashmap的实现原理，好多五年以上经验的人都弄不清楚，还有为什扩容时有性能问题？不弄清楚这些原理，就写不出高效的代码，还会认为自己做的很对；

总之一句话，越基础的东西越重要，很多人认为自己会用它们写代码了，其实仅仅是知道如何调用api而已,离会用还差的远。

如果你立志做架构，首先打好基础，从最底层开始。然后发展到各种技术和语言，什么都要懂两点，要全面且不肤浅。为什么不是懂一点？你要看得透彻，必须尽量深入一些。别人懂一点，你要做架构师，必须再多懂一点。比如你发现golang很流行，别人可能写一个helloworld就说自己玩过golang，但你至少要尝试写一个完整的应用。不肯下苦功，如何高人一头？

另外你要非常深入地了解至少一门语言，如果你的目标是java，就学到极致，作为敲门砖，先吃饱了才能谈理想。

**3，架构师都是从码农过来的**

而Java学到极致势必涉及到设计模式，算法和数据结构，多线程，文件及网络IO，数据库及ORM，不一而足。这些概念放之一切语言都适用。先精一门，为全面且不肤浅打基础。

另外就是向有经验的架构师学习，和小伙伴们讨论辩论争论。

其实最重要的能力就是不断学习。在思考新的技术是否能更好地解决你们遇到的问题之前，你首先得知道并了解新的技术。

架构师都是从码农过来的，媳妇熬成婆。

千万不要成为不写代码的架构师，有些公司专门产不写技术的架构师。

所谓架构师，只是功底深厚的程序员而已。个人认为应该扎扎实实学习基础知识，学习各种规范，架构，需要广泛的知识面，懂的东西越多视野越开阔，设计的东西当然会越好越全面。

成为架构师需要时间的积累的，不但要知其然还要知其所以然。平时的一点一滴你感觉不到特别用处，但某天你会发现所有东西都没有白学的。

**4，架构师知识体系**

下面是我总结多年经验开发的架构师知识体系

一、分布式架构

架构分布式的英文（ Distributed computing 分布式计算技术）的应用和工具，成熟目前的技术包括 J2EE，CORBA 和 .NET（DCOM），这些技术牵扯的内容非常广，相关的书籍也非常多。

本文不介绍这些技术的内容，也没有涉及这些技术的细节，只是从各种分布式系统平台产生的背景和在软件开发中应用的情况来探讨它们的主要异同。

分布式系统是一个古老而宽泛的话题，而近几年因为“大数据”概念的兴起，又焕发出了新的青春与活力。

除此之外，分布式系统也是一门理论模型与工程技法。并重的学科内容相比于机器学习这样的研究方向，学习分布式系统的同学往往会感觉：“入门容易，深入难”的确，学习分布式系统几乎不需要太多数学知识。

分布式系统是一个复杂且宽泛的研究领域，学习一两门在线课程，看一两本书可能都是不能完全覆盖其所有内容的。

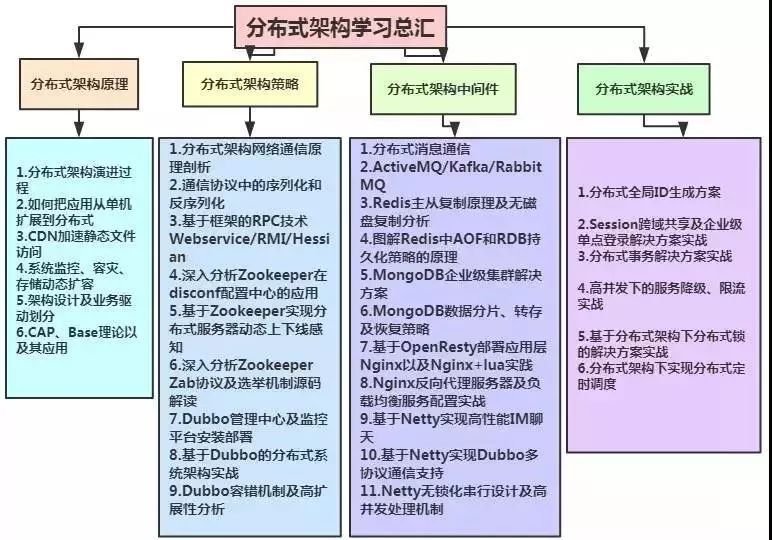
总的来说，分布式系统要做的任务就是把多台机器有机的组合，连接起来，让其协同完成一件任务，可以是计算任务，也可以是存储任务。

如果一定要给近些年的分布式系统研究做一个分类的话，我个人认为大概可以包括三大部分：

1. 分布式存储系统

2. 分布式计算系统

3. 分布式管理系统



二、微服务

当前微服务很热，大家都号称在使用微服务架构，但究竟什么是微服务架构？微服务架构是不是发展趋势？对于这些问题，我们都缺乏清楚的认识。

为解决单体架构下的各种问题，微服务架构应运而生。与其构建一个臃肿庞大，难以驯服的怪兽，还不如及早将服务拆分。微服务的核心思想便是服务拆分与解耦，降低复杂性。

微服务强调将功能合理拆解，尽可能保证每个服务的功能单一，按照单一责任原则（Single Responsibility Principle）明确角色。将各个服务做轻，从而做到灵活，可复用，亦可根据各个服务自身资源需求，单独布署，单独作横向扩展。

微服务架构（Microservice Architecture）是一种架构概念，旨在通过将功能分解到各个离散的服务中以实现对解决方案的解耦。你可以将其看作是在架构层次而非获取服务的

类上应用很多 SOLID 原则。微服务架构是个很有趣的概念，它的主要作用是将功能分解到离散的各个服务当中，从而降低系统的耦合性，并提供更加灵活的服务支持。

概念：把一个大型的单个应用程序和服务拆分为数个甚至数十个的支持微服务，它可扩展单个组件而不是整个的应用程序堆栈，从而满足服务等级协议。

定义：围绕业务领域组件来创建应用，这些应用可独立地进行开发，管理和迭代在分散的组件中使用云架构和平台式部署，管理和服务功能，使产品交付变得更加简单。

本质：用一些功能比较明确，业务比较精练的服务去解决更大，更实际的问题。



三、源码分析

从字面意义上来讲，源文件的英文指一个文件，指源代码的集合。源代码则是一组具有特定意义的可以实现特定功能的字符（程序开发代码）。

源码分析是一种临界知识，掌握了这种临界知识，能不变应万变，源码分析对于很多人来说很枯燥，生涩难懂。

源码阅读，我觉得最核心有三点：技术基础+强烈的求知欲+耐心。

我认为是阅读源码的最核心驱动力我见到绝大多数程序员，对学习的态度，基本上就是这几个层次（很偏激哦）：

1，只关注项目本身，不懂就百度一下。

2，除了做好项目，还会阅读和项目有关的技术书籍，看维基百科。

3，除了阅读和项目相关的书外，还会阅读IT行业的书，比如学的Java的时，还会去了解函数语言，如LISP。

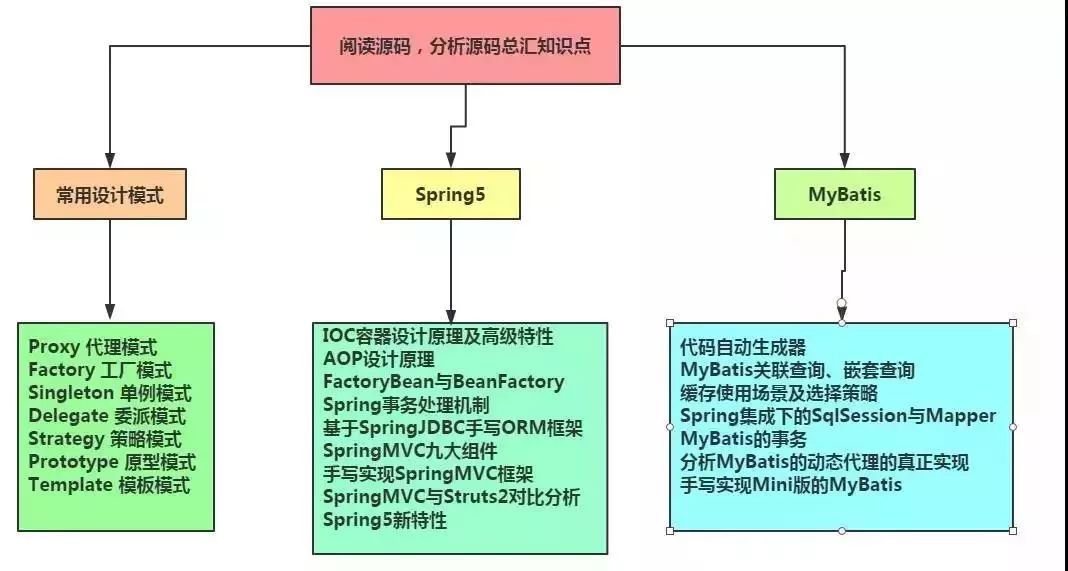
4，找一些开源项目看看，大量试用第三方框架，还会写写演示。

5，阅读基础框架，J2EE 规范，调试服务器内核。

大多数程序都是第1种，到第5种不光需要浓厚的兴趣，还需要勇气：？我能读懂吗其实，你能够读懂的

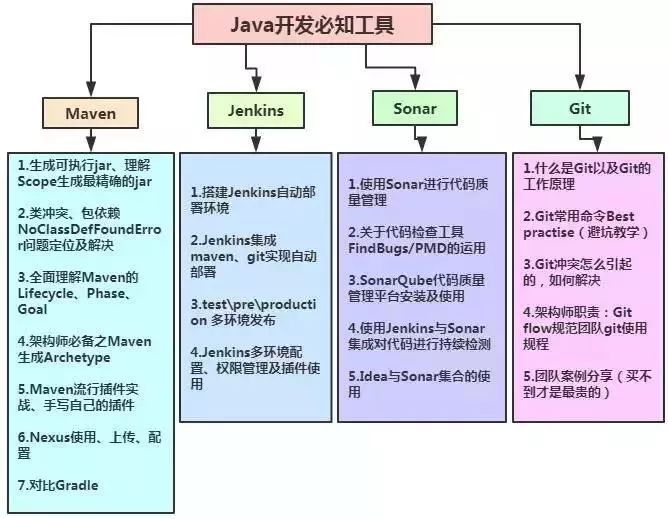
耐心，真的很重要。因为你极少看到阅读源码的指导性文章或书籍，也没有人要求或建议你读。

你读的过程中经常会卡住，而一卡主可能就陷进了迷宫这时，你需要做的，可能是暂时中断一下，再从外围看看它：如API结构，框架的设计图。



四、工具使用

工欲善其事必先利其器，工具对 Java 的的程序员的重要性不言而喻现在有很多库，实用工具和程序任的 Java 的开发人员选择。下图列出的工具都是程序员必不可少的工具



五、性能优化

不管是应付前端面试还是改进产品体验，性能优化都是躲不开的话题。

优化的目的是让用户有“快”的感受，那如何让用户感受到快呢？

加载速度真的很快，用户打开输入网址按下回车立即看到了页面

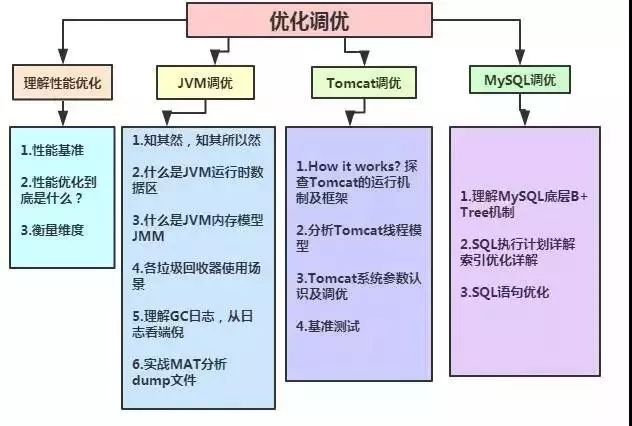
加载速度并没有变快，但用户感觉你的网站很快

性能优化取决于多个因素，包括垃圾收集，虚拟机和底层操作系统（OS）设置。有多个工具可供开发人员进行分析和优化时使用，你可以通过阅读爪哇工具的源代码优化和分析来学习和使用它们。

必须要明白的是，没有两个应用程序可以使用相同的优化方式，也没有完美的优化的 Java 应用程序的参考路径。

使用最佳实践并且坚持采用适当的方式处理性能优化。想要达到真正最高的性能优化，你作为一个 Java 的开发人员，需要对 Java 的虚拟机（JVM）和底层操作系统有正确的理解。

性能优化，简而言之，就是在不影响系统运行正确性的前提下，使之运行地更快，完成特定功能所需的时间更短。性能问题永远是永恒的主题之一，而优化则更需要技巧。



## 程序员之路

如下是我做开发这几年经常使用一些技术和工具，当然这些技术也都是需要加强的（有些是我一直使用的，不过不深入，有些内部的原理等等不是很清楚）

前端部分：

       1）HTML：网页的核心语言，构成网页的基础

       2）CSS：使网页更加丰富多彩灿烂的利器

       3）JavaScript：使网页动起来的根本，加强了网页和用户之间的交互

       4）HTML DOM：换一种更加形象化的角度来看待网页，让我们更加方便的控制网页

       5）HTML BOM：与浏览器交互不可或缺的工具

       6）JavaScript库，主要是：jQuery及其插件、YUI及其插件，使编写网页更加的方便快捷和健壮的强大工具

       7）AJAX：异步提交，增强了用户使用网页的良好交互体验

       8）JSON： 比 XML 更小、更快，更易解析的数据传输工具

       9）FLEX：提供丰富多彩的动画效果

       10）普元工作流：更加清晰明了的帮助用户处理业务流程上面的工作

       11）JSP：Servlet的展示层，使网页逻辑与网页设计的显示分离

       12）JSTL：加强和简化了JSP页面的开发

       13）EL：使JSP页面写起来更加简单

后台部分：

       1）JAVA语言编程基础部分：内容丰富是Java知识体系结构的核心和基础

       2）JDBC：提供了一种基准，据此可以构建更高级的工具和接口，使Java开发人员能够编写数据库应用程序

       3）JavaMail：用于电子邮件的相关的编程工作

       4）JUnit：单元测试，整个变成工作测试的地位始终非常重要

       5）Log4j：使我们能够更加细致地控制日志的生成过程

       6）Servlet：JavaWeb的核心

       7）Struts2：JavaWeb编程中明星级的框架，优点多功能强使编程工作更简单，主要用于控制跳转

       8）Spring：JavaWeb编程中明星级的框架，同样优点多功能强使编程工作更简单，主要用于管理对象

       9）iBatis：JavaWeb编程中明星级的框架，同样也是优点多功能强使编程工作更简单，主要用于程序和数据库之间的交互

       10）SQL：与数据库交互式编程的必备工具

版本控制：

      1）SVN：版本控制，方便团队协同工作

WEB服务器：

      1）Tomcat：优秀免费的中小型WEB服务器

      2）Weblogic：功能很强大的WEB服务器

开发工具：

      1）Eclipse：开源强大的Java编程工具

      2）MyEclipse：在eclipse 基础上加上自己的插件开发而成的功能强大的企业级集成开发环境

数据库：

     1）Oracle：数据库业界老大，这个常用一点当然，也仅仅是常用一些常用的功能而已，需要加强

     2）MySQL：最好的关系型数据库之一

     3）SqlServer：最好的关系型数据库之一

数据库客户端：

     1）Toad：非常棒的数据库客户端软件

操作系统：

     1）Windows

     2）Linux

远程登录的会话工具：

    1）SSH：一直在用，非常不错

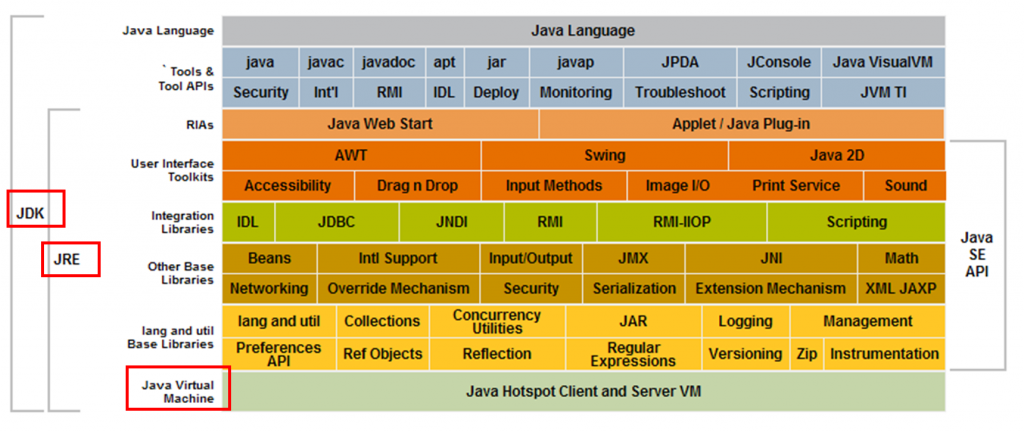
其他：

      各种种类和版本的浏览器（推荐使用FireFox）、office办公软件系列、各种文本编辑器（推荐Sublime）、各种阅读器等等

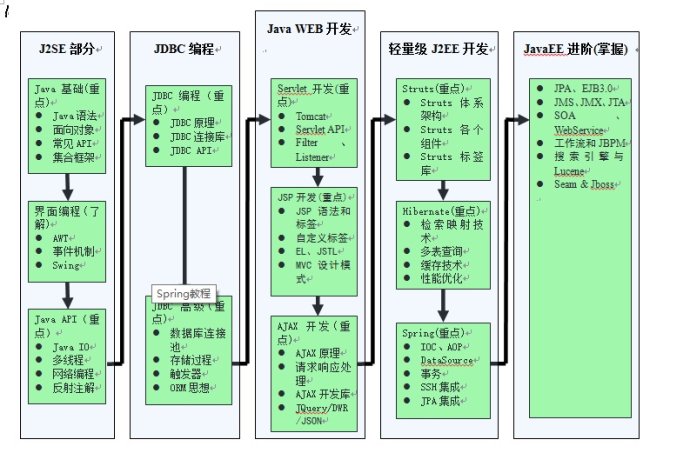
恩，回头看，针对开发工作自己使用过的东西还是不少的，不过每一样都称不上是精通的，仅仅是常常使用，使用那些常用的功能，我在这里记录下来就是要一个个补上来，当然，肯定也需要不断的学习其它一些新的技术和工具的！是对自己的一个督促吧！指明了自己努力的方向！

下面是从网上找的三幅图片，比较好的说明了作为一个Java程序员的学习路线图，我放在这里作为一种自我提醒的工具，不时看看，并且会赶紧修补自己的知识树！

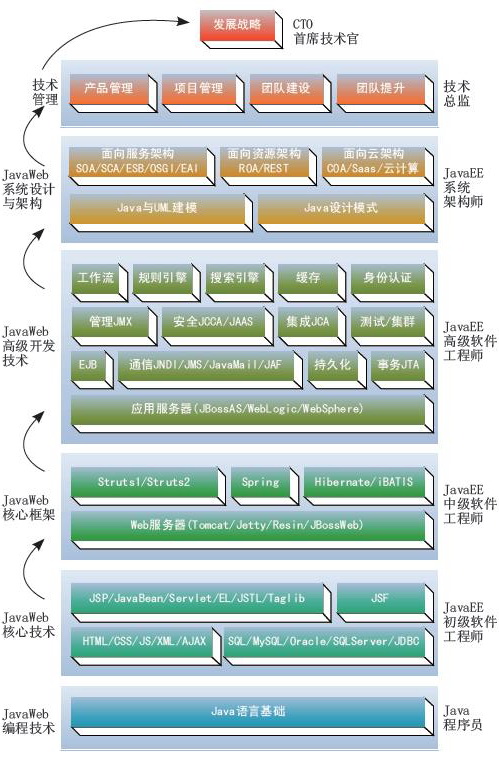
**1：Java的知识体系结构**



**2：JavaWeb开发的基本学习路线图**



**3：JavaWeb开发的成长路线图**



恩，今年也看了不少的书，感觉收获挺大的，2018年期待有个更加美好的开始！目标已经有了，下面就看行动了！记住：学习永远是自己的事情，你不学时间也不会多，你学了有时候却能够使用自己学到的知识换得更多自由自在的美好时光！时间是生命的基本组成部分，也是万物存在的根本尺度，我们的时间在那里我们的生活就在那里！我们价值也将在那里提升或消弭！少年加油吧！

最后的建议：学习各种技术的时候去对应技术的官方网站（百度一下就知道了！），买专业出版社和专业人士出的书籍，英文是一大关，也是区分各级程序员的一道网，我的英语也比较差，不过我在持续的学习中，我相信我能拿下它，最近我学习技术的时候也比较喜欢去官网硬着头皮啃英文，并且觉得国外这些做技术的太厉害啦！当然，他们是鼻祖嘛！加油！