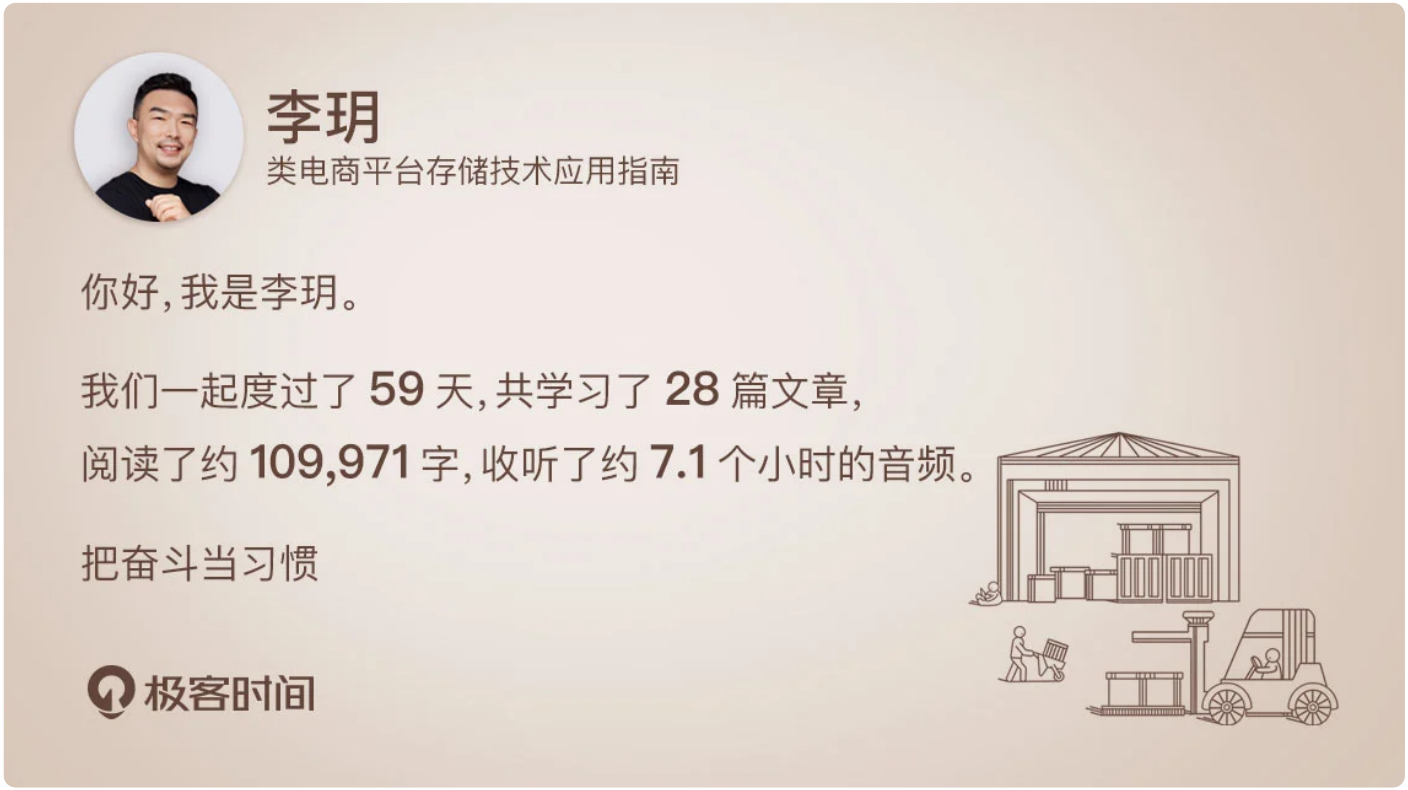
结束语 | 把奋斗当习惯

李玥 · 后端存储实战课



你好，我是李玥。

今天这节课，是我们这系列课程的最后一讲，我想跟你聊聊我对个人技术成长的感悟。

程序员是一个特别依赖个人技术能力的职业，不同的程序员之间，技术能力的差别也非常大。一个大神程序员的产出，可以抵得上好几个普通程序员。一个技术差还自以为是的程序员，他的产出更是能抵得上几十个程序员，不过这个产出是负的。所谓一人写 Bug，大家加班来找茬，相信很多人都有过这样的经历。

相应的，程序员的收入差距也非常大，从年入几万到几百万的都有。同样是应届生，从 CRUD（增删改查）开始做起，几年之后有些人还在“CRUD”，只是更熟练了而已。而有的人技术成长得非常快，职位和收入也随之水涨船高。

为什么有些人的技术能快速成长？这里面的原因很复杂，每个人的天赋、工作经历、选择甚至于运气都是影响因素。但除此之外，还是会有一些方法和经验，可以为你的技术成长提速。

把奋斗当成习惯

技术的原始积累，是个人技术能力的基础。这个“积累”主要指的是，你要有足够多的技术经历，这里面包括你读过的书、写过的代码、做过的项目、解决过的 Bug、用过的框架、踩过的坑儿以及遇到的各种线上问题，等等。

为什么说技术积累这么重要呢？没有技术积累我直接去学习技术原理，去刷题，通过这些手段去提升技术不行么？

那我给你举个例子，包括很多学习计算机专业的同学，刚毕业的时候，觉得大学里面学到那些专业课和实际工作脱节的厉害，还没有培训班讲的实战课有用。反而很多有多年技术经验的资深程序员，开始捡起大学那些课本去回炉重学。越是大厂技术面试，基础知识、算法、设计模式这些占比越重，这里面很多都是大学专业课学过的知识。

为什么会有这样的现象？因为如果没有足够的技术积累，你很难理解书本上的技术知识和原理它的用途在哪儿，所以会觉得没用。只有你遇到过这样的问题，有过困惑，再去看书上讲的知识，立刻就会有一种恍然大悟的感觉。所以说，原始的技术积累非常重要。

对于技术积累，没有捷径可以走，如果想做到快速积累，只有多写代码、多做项目这一条路。具体的做法很简单，也很难，就是去主动地去多做事情，不要去管这些事情是不是职责范围内的、有没有报酬、会不会有收获、对技术有没有提升。不计得失，任劳任怨。

这个做法说来简单，它真的就很简单，只要想去做，每个人都做得到。但是，它也很难。因为并不是每个人都打心里认同这个做法，**如果你没有内心的认同，又强迫自己这么做，是非常痛苦的，并且很难坚持下来**。所以，问题的关键是寻求内心的认同。

我刚毕业那几年，对这个观点就是非常不认同。我当时的想法是这样的：每个月才给我这么一点儿钱，凭啥我要主动去做这么多事儿？我对公司产生的价值，绝对对得起我的工资，再多干活，公司又不会给我钱。

我估计很多年轻的朋友也会有同样的想法。我们不能说这个想法就不对，实际上这里面涉及到人生观的问题。比如说，有的人清楚地知道自己想要什么，“我不追求什么技术，也不在乎职务收入，工作只是我谋生的手段，我更看重的是诗和远方”。

但我是一个“俗人”，希望不断提升自身的技术能力，获得自我认同，同时也获得更好的收入和体面的生活。如果你也和我一样不能免俗的话，我建议你在内心上尽快做一个转变。什么转变呢？**从“凭什么要我做？”，到“愿意主动去多做事儿”，再到“把奋斗当做习惯”。**

我的转变来自一顿酒局。记得当时也是一个前辈在一次一起喝酒的时候提点了我。当我借着喝点儿酒，和他抱怨当时的工作钱少事儿多没技术含量，他一句话点醒了我：“如果你是老板，当你有一个重要的任务，你是愿意交给那个只做分内事和你斤斤计较的人，还是交给那个不计得失兢兢业业的人呢？”

其实你真正应该较劲的人，不是那个扣扣索索不舍得给你发钱的老板，而是那些和你一起竞争有限发展空间的同龄人。主动去多做一些事儿，不仅能获得更多的成长和锻炼，更重要的是获得周围人对你的认同，这里面也包括你的领导和老板。这样你就会获得更重要的任务和更多的锻炼机会，才能相比同龄人更快速地成长，用更短的时间快速实现技术积累。

想通了这个道理，即使去做一些没有意义的脏活累活，心里也不会感觉那么痛苦了。

思考沉淀，让点成势

你的技术能力，会随着你的技术积累线性增长。当你的技术经验积累到一定程度的时候，你需要**停下来**，给自己几天时间，什么都不做，放空一下自己，利用这段时间去思考，问自己两个问题：

> 这段时间我都做了什么？
>
> 技术上我都学到了什么？

然后，在脑海中把这两个问题的答案再梳理一下，这个时候你就有可能会发现，你之前积累的零散的知识，它们之间其实是有联系的，然后再通过总结，你就有可能会在某一个小的技术领域上，构建出一个知识体系。

原来看不清楚脉络的技术，有可能就看清楚了。反过来，理清了技术脉络，构建起知识体系之后，也会极大地加快你继续学习和积累的速度。

这么说有点儿抽象，我们还是通过一个例子来看。比如说学习一门新的编程语言，这个对很多同学来说都是一个挺大的挑战，但我现在是可以做到用一周的时间来学会一门全新的编程语言。

达到什么程度呢？精通和熟练是肯定谈不上的，但至少可以做到写出规范和合格的代码，去开发一个真正可以用于生产的系统，这个是没问题的。

我在上一门课程《消息队列高手课》中，有很多的示例代码，用了 Java、Scala、Python 和 Golang 四种比较流行的编程语言。我日常工作使用的是 Java 语言，Python 偶尔会到。当时在编写这门课程的时候，Scala 和 Golang 这两门语言都是现学现卖的。

我之所以能够做到快速学习，一个前提是，我之前在熟练地掌握了二、三门语言之后，经过了思考和总结，理清楚了编程语言的技术体系是什么样的，以及我已经掌握的这几门语言，它们之间有哪些共通的知识。

当我再学习一门全新的语言时，我首先会去看一下，这门语言和其他语言有什么不一样的特性，这些特性往往是，为了解决其他语言中那些不容易解决的问题而诞生的。比如，最近特别火的 Rust 语言，它之所以这么火的原因是，它采用了所有者模型来解决内存管理的老大难问题。如果你经历过用 C++ 内存泄漏的痛苦，也体会过 Java 以及 Golang 动不动 Stop the world 的垃圾回收器，那你一下子就能理解 Rust 语言的这个特性有多可爱。

你可以看到，快速理解这些新特性的基础，还是要有足够的技术积累做支撑，如果你没有 C++、Java 和 Go 这些语言的使用经验，你的感受可能是：为什么会有所有者模型这么奇葩的设计？这个垃圾的编译器为什么总是编译不通过？

再说回来学习编程语言这件事，了解完一门新语言的特性之后，我会看一下它的基本语法、线程模型以及内存管理模型是什么样的，是不是和已有的语言是一样的机制；再看一下它的基础类库，包括常用的集合类、如何读写文件、如何处理输入输出这些；再有就是它的源代码如何组织，编译构建系统是什么样的，如何处理类库之间的引用依赖这些编译运行的问题；最后还要看一下这门语言的生态系统，比如最常用的 Web 框架、RPC 框架是什么，一些常用的场景下，配合哪些中间件最合适等等。

当然这么多内容，不可能一下子都记住，但你会发现，这里面绝大部分内容都是和其他语言差不多的，我们只要记住这个语言独有的那些特性就好。

了解了以上这些内容，基本上我们就算是初步掌握一门语言了。不过我可能还没有那么熟练，写几行代码就得去看看文档和例子，写的还比较慢，但我们写出来的代码规范性和正确性是可以保证的。剩下的就交给时间，逐步去练熟直到精通。

所以，**停下来，去思考沉淀，让点成势，构建出自己的知识结构，是技术成长的捷径。**

洞见技术本质

如果说，我们能够不断地积累，思考，再积累，再思考，那不仅你的技术成长会非常快，反复地总结和思考，也会在无形中逐渐提升你的思考能力。

随着你的知识体系越来越完备，总结和思考的能力越来越强大，那你也就会越来越容易看清一项新技术的本质和原理，这又非常有助于你快速地学习一些新的技术。这个时候，你会有一种学习任何技术都很容易很轻松的感觉，恭喜你，你已经完成了技术的升华和蜕变。

但这个时候，我还要打你一棒子，那个很轻松、很容易的感觉其实是个错觉。为什么这么说呢？因为技术的原理或者说是本质，它本来就是很简单的，并不是我们有厉害。真正复杂和难的是工程实践中的细节。

比如说，汽车发动机的原理大家都知道，汽油燃烧热胀冷缩推动活塞做功。但是，这个地球上真正能造出可靠耐用的汽车发动机的公司并不多，原因就是光掌握原理是不够的，还要解决很多复杂的工程问题。

看清一项技术的原理之后，会利于我们快速学习这项技术，但要想达到精通并熟练的应用，还是要沉下心来，去深入学习、研究、使用和总结，这个功夫是少不了的。

好的，以上这些就是我对个人技术成长的一点点感悟。其实这些知识道理并不高深，只是能够做到，并且将这些道理变成自己的信条时，你才真正拥有了它。

到此，我们的《后端存储实战课》也就告一段落了。课程结束并非终点，我们还可以在留言区互动交流，也祝你享受成长，学有所成。

再见。

精选留言 (75)

每天晒白牙
2020-04-25

这篇结束语让我收获很多，谢谢老师


36

kyll
2020-04-28

受益匪浅， 自从来了极客时间， 我都不敢说自己学过计算机。
基础和细节都是魔鬼。在写了N年代码后，从计算机组成原理开始补课了。

共 2 条评论>


20

Middleware
2020-04-25

把奋斗当习惯，一直觉得自己的工作，对不起工资😂

共 7 条评论>


15

每天晒白牙
2020-04-25

很多人都觉得自自己做的事儿已经对得起工资了，不愿多做，甚至有了"面向RMB编程"这种调侃，你是不是也经常这样想?

老师说要做到技术技术积累唯有多写代码，多做项目。做法比较简单，但思维的障碍却是拦路虎。大家都是普通人，难免去计较得失，就是这样才限制了自己的成长。

老师说很关键的一点是获得内心的认同，怎么做到?

要在内心做一个转变:

"从凭什么要我做？" 到 "愿意主动去多做事儿"，再到"把奋斗当做习惯"

对我很受启发的就是后面这句话，其实就是主动去做，多做。

我的领导之前分享过一个观点，大意是如果你能站在他的位置思考问题，那么以后你就很可能取代他的位置。

两者虽不同，但觉得还有些相通

我身边的同事有很多都是比较主动型的，他们会主动去做一些事儿，而不仅限于产品提出的业务需求，在多做事儿的同时，成长也可想而知了

15

Jxin

2020-06-02

1.看了几十个专栏，栏主的这个结束语算得上佳作了。也不是说其他栏主的不好，主要其他栏主的结束语都落在感谢和技术展望上。而栏主的这段肺腑之言，无疑是故清流。

2.内心的抵触是个拦路虎，这个确实是个问题。所幸我好像没有过这个纠结。主动去做有意义的事，这一直都是一个习惯。不过，技术变化快，公司环境又往往比较慢，除了埋头奋斗，选择也很重要，甚至更重要。

3.我曾用近一年的下班时间，重构了项目过半的代码。有自己的，也有别人的。这个事情吃力不讨好，价值也不好说，毕竟我重构的部分代码可能还是垃圾甚至更糟糕。但仅是“我写过的代码就是我的名片”这么一个追求，我坚持去做了这么一件事，并且这一年学习的重心也落在了编码规范，面向对象，设计原则，设计模式，领域驱动，接口测试，整洁架构等等的书籍上（知行合一，很多事都需要，写代码尤其需要）。结果就是现在写代码思路很清晰，慢慢可以做较为复杂系统的设计和分析，而且看设计和代码质量好的开源项目也很顺（看差的就骂骂咧咧了。。）。讲心里话，这个成长我很满意。但还是被定义为“技术能力不够”。倒也没啥好抱怨，但这也提醒了我一点。长线收益和短线收益要管理好，打基础功重要，但齐备深入的技术栈也重要，特别在所处环境一般时，短线收益可能更重要。

4.奋斗是必须的，但根据现状调整路线，可以更顺心，也可能事半功倍。感谢栏主分享，共勉。

作者回复: 确实选择和努力同样重要。低头奋斗，也不要忘了抬头看看世界的变化，你会发现更好的选择。

共 2 条评论> 13

Geek_e23d98
2021-12-07

后端存储实战课知识总结

创业阶段

保证数据准确（ 具有幂等性的服务）

流量大，数据多如何存

– 分而治之–按数据特点分别存储

购物车如何设计?

对账问题

– 采用数据库事务的思想解决

分布式事务

– 2PC，3PC，TCC

    – 2PC:准备阶段和提交阶段

ES––全文搜索的分布式内存数据库

MySQL HA

– 高可用依赖的是数据复制，数据复制本质是从一个库备份数据恢复到另一个库中去
– 多种HA方案，在高可用，数据可靠性，性能中做取舍

高速增长

数据库访问超时

- 系统架构层的改进，让系统更加健壮
- 系统关键部分自我保护，首页降级减少影响

如何避免写出慢sql

- 定量认识mysql
    - 1w,百万，千万
    - 百万以内可认为是安全的，百-千万需要评估，千万以上非常危险
- 使用索引，避免全表扫描
- 分析sql执行计划
    - explain sql语句
    - 关注rows 和type

sql在数据库中是如何执行的

- 数据库由执行器和存储引擎组成
- 逻辑执行计划->物理执行计划
- B+树

MySQL如何应对高并发

- 使用Redis做前置缓存，使用Cache Aside模式更新缓存
- 避免大量缓存穿透引起雪崩，可采用灰度发布或者缓存预热
- 首选数据库扩容
- 读写分离
    - 手动配置数据源
    - 组件方式
    - 代理方式
    - 推荐采用读写分离组件做改造

数据库同步如何实现呢?

- 同步复制
- 异步复制
- 半同步复制
- 复制状态机-全量备份和Binlog

数据库越来越慢怎么办

- 查询耗时取决于：查询时间复杂度和数据总量
- 查找时间复杂度取决于：查找算法和数据存储结构
- 对数据进行拆分-学名：分片(Shard)

    - 首选进行历史数据归档

海量数据

存储海量数据最后一招：分库分表

- 原则：能不拆就不拆，能少拆就少拆
- 数据量大就分表，高并发就分库
- 越简单的设计可靠性越高

用Redis构建缓存集群

- Redis Cluster

    - 中小规模
    - 去中心化设计 --流言（Gossip）协议-避免中心节点的单点故障，但传播慢

- 超大规模集群

    - 基于代理模式
    - 寻址功能移到客户端中

如何做MySQL to Redis

- 使用Binlog 实时更新Redis缓存
- 常用开源项目：Canal

分布式存储：对象存储

– 近乎无限的存储容量，超高的读写性能，数据可靠性，服务高可用
– 如：ES，Ceph
– 对大文件进行拆分

跨系统实时同步数据

– Binlog + MQ

不停机，安全更换数据库

– 设计技术方案时要保证安全性，每个步骤可回滚
– 参考步骤：1：上线同步程序，从旧库复制数据到新库；2上线双写服务，只读写旧库；3.开启双写，同时停止同步程序；4.开启对比和补偿程序，确保新就数据库数据完全一致；5.逐步切量读请求到新库；6.下线对比补偿程序，关闭双写，读写切换到新库；7.下线旧库和服务的双写诶功能

点击流，监控，日志等海量数据如何存储

– 海量数据中的海量数据
– 先存储再计算，计算结果保存到业务库，供使用
– kafka

    – 读写性能好

– HDFS

    – 近乎无限存储容量，对查询友好

– 分布式流数据存储
– 时序数据库存储

海量数据如何查的快

– 根据数据规模不同有小到大可以选择：关系型数据库，列式数据库如Hbase和一些大数据存储系统，TB及以下数据可以选择ES,超过TB级的数据通常事定期聚合计算，保存结果，配合

大数据系统
– 根据查询选择存储系统

NewSQL

– 新一代分布式数据库
– 高速发展阶段，不建议做小白鼠

RocksDB

– 高性能持久化的KV存储

4

Jackey
2020-05-03

老师这结束语走心了

3

Alan
2020-04-25

谢谢老师，让我学习了很多。江湖再见👋

3

丁丁历险记
2020-05-22

每天尽力的工作，赚取的时间摸鱼做死，成熟后反馈到生产上来

2

阿斯蒂芬
2020-05-19

“思考沉淀”，读起来多简单的四个字，实践起来却是不易。前期看着还有一堆堆的这个那个知识、技术、框架不懂，总想着不停去快速学习，越学越焦虑，反而成了一个“恶性焦虑循环”。
老师苦口婆心安利“停下来”，我觉得是非常有用的，特别是对于基础的“技术本质”知识，一定要舍得花时间去啃，能啃多透啃多透，能多几遍就多几遍，所谓常读常新，学过的知识不一定就没有重新学习的价值，重复总结，提炼，思考，才能达到“知识体系”
所谓的”技术捷径“，我想可能并不是捷径，而是一种方法论，核心就是坚持，带着”内心认

同“的坚持。
谨以此留言再次告诫自己，警惕”无用的焦虑“，多”沉淀“。
也感谢老师的精品专栏。
业精于勤而荒于嬉，行成与思而毁于随。与君共勉。

作者回复: 是这样的，战略思考永远是高于战术的运用。

2

第一装甲集群司令克莱...
2020-09-26

通过技术的第一性原理知道，技术是前赴后继，互相弥补的，哪有那么多的新技术，都是基础技术的模型演变而来的。

1

djfhchdh
2020-09-22

后几章有些拉跨啊，可以再详细些。不过，每课之后的思考题解答好评，非常及时高效。

1

jacky
2020-06-16

对任劳任怨要看年龄和阶段，还要看是否是枯燥的重复，不能做无助与自己进步的累砖头吧?

共 1 条评论>

1

QQ怪
2020-05-14

老师受教了，感激!

作者回复: 希望这门课程能对你有所帮助。

1

aoe
2020-05-06

高并发、大数据下性能调优的时候，才感觉到理论基础的重要性

1

而立斋
2020-04-26

虽然课程还没有看完，但是看到老师的【把奋斗当成习惯】，这句有一些共鸣了，我从18年开始树立了【把学习当成习惯】的slogan，几乎每天都会花上那么一至两个小时用来学习，并没有觉得痛苦，但是越学越觉得焦虑，抬头一看我中，还有这么多东西是我所不知道的。甩了甩一头乌黑而茂密的秀发，赶紧记下来继续学吧！早晚我会有所收获的

1

Forr
2020-04-26

非常赞同老师观点，停下来，去思考沉淀，让点成势，构建出自己的知识结构，是技术成长的捷径。我对技术成长的捷径的理解是通过思考总结可以减少实践次数，当弄透彻一项技术，类似的技术也就更易理解，试错成本大大地降低了

1

一步
2020-04-25

思考沉淀，构建自己的知识体系。 这个过程真的是辛苦的

1

达芬奇
2023-06-25 来自北京

感谢老师的分享

ifelse
2022-12-19 来自浙江

感谢老师