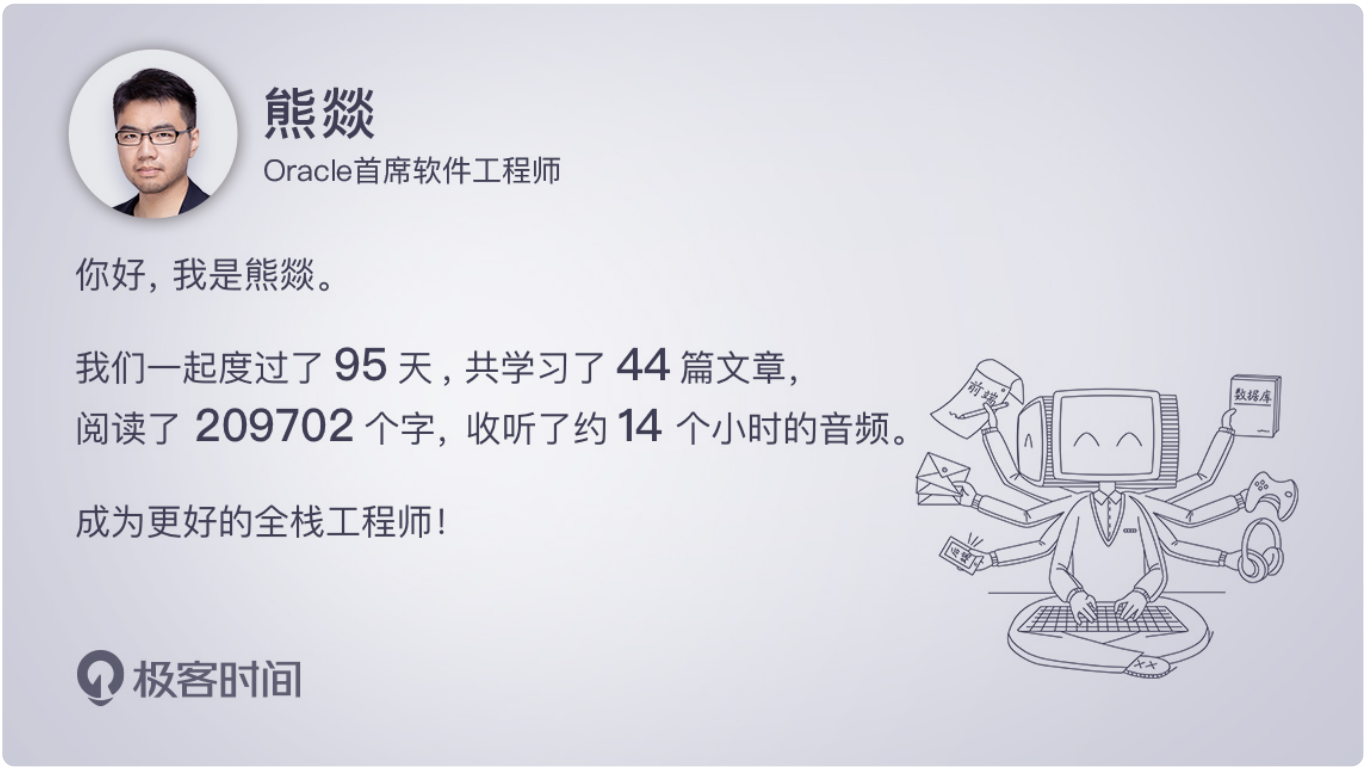
全栈回顾 | 成为更好的全栈工程师！

2019-12-13 四火

全栈工程师修炼指南 进入课程 >



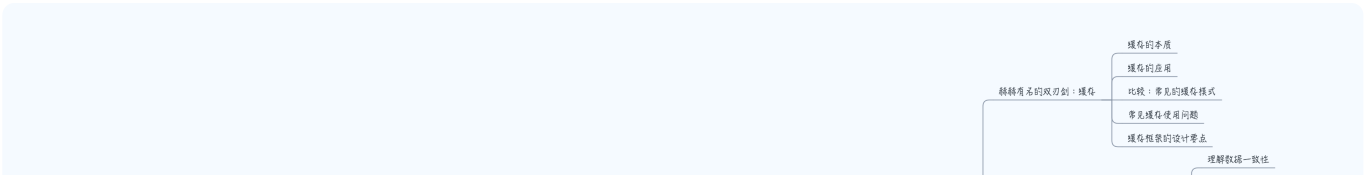
讲述：四火
时长 12:45 大小 8.77M

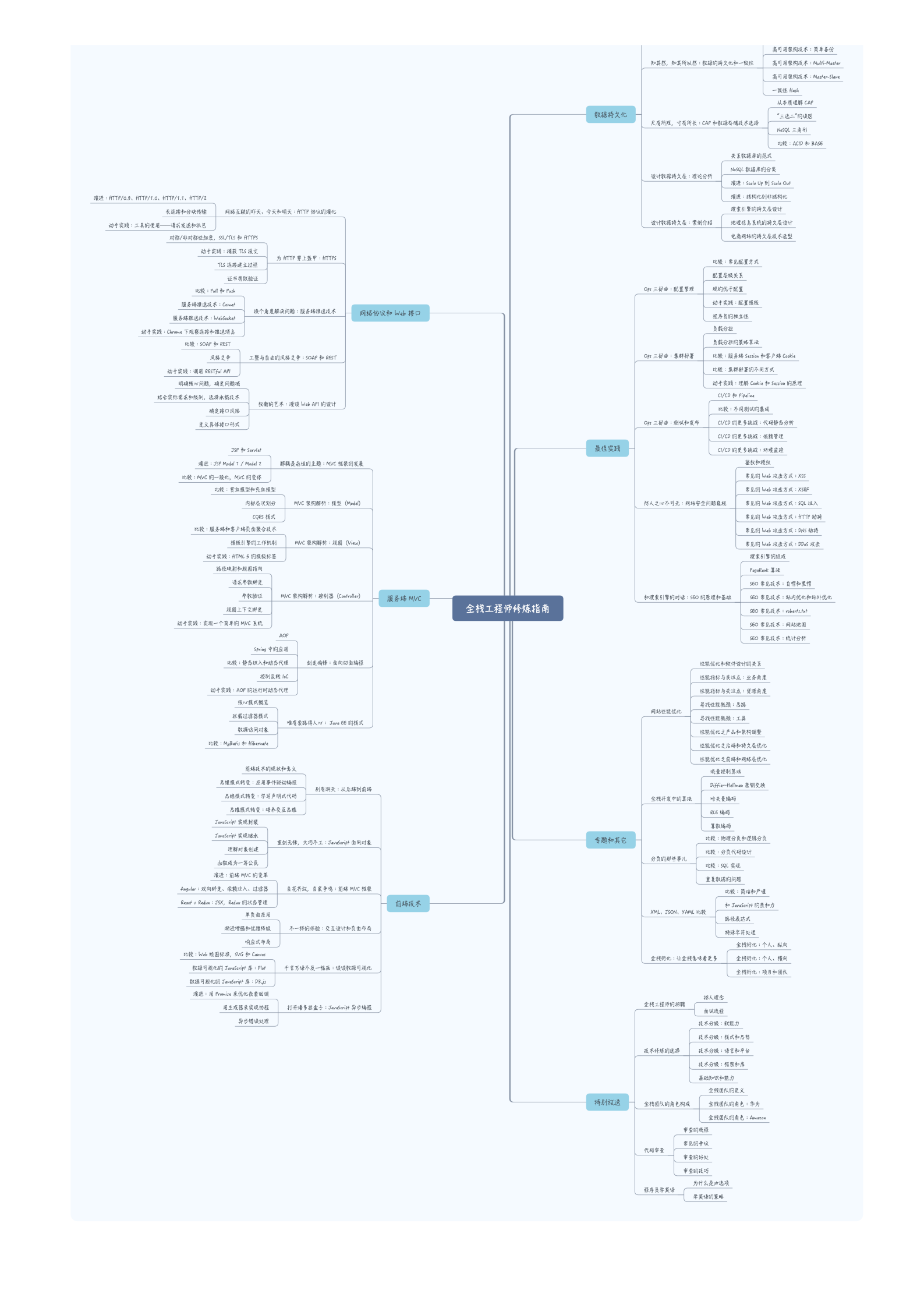
你好，我是四火。

这是这个专栏的最后一讲了。

回顾一下，我们学到了什么？

现在，是时候来回顾一下我们学过的 Web 全栈树了。这里我按章节进行划分，把每一讲的标题和主要内容做成了一张思维导图，帮助你回顾。




全栈工程师修炼指南
数据持久化
知其然，知其所以然：数据的持久化和一致性
高可用架构技术：简单备份
高可用架构技术：Multi-Master
高可用架构技术：Master-Slave
一致性 Hash
尺有所短，寸有所长：CAP 和数据存储技术选择
从本质理解 CAP
“三选二”的误区
NoSQL 三角形
比较：ACID 和 BASE
设计数据持久层：理论分析
关系数据库的范式
NoSQL 数据库的分类
渐进：Scale Up 到 Scale Out
渐进：结构化到非结构化
设计数据持久层：案例介绍
搜索引擎的持久层设计
地理信息系统的持久层设计
电商网站的持久层技术选型
最佳实践
Ops 三部曲：配置管理
比较：常见配置方式
配置层级关系
规约优于配置
动手实践：配置模板
程序员的独立性
Ops 三部曲：集群部署
负载分担
负载分担的策略算法
比较：服务端 Session 和客户端 Cookie
比较：集群部署的不同方式
动手实践：理解 Cookie 和 Session 的原理
Ops 三部曲：测试和发布
CI/CD 和 Pipeline
比较：不同测试的集成
CI/CD 的更多挑战：代码静态分析
CI/CD 的更多挑战：依赖管理
CI/CD 的更多挑战：环境监控
防人之心不可无：网站安全问题盘点
鉴权和授权
常见的 Web 攻击方式：XSS
常见的 Web 攻击方式：XSRF
常见的 Web 攻击方式：SQL 注入
常见的 Web 攻击方式：HTTP 劫持
常见的 Web 攻击方式：DNS 劫持
常见的 Web 攻击方式：DDoS 攻击
和搜索引擎的对话：SEO 的原理和基础
搜索引擎的组成
PageRank 算法
SEO 常见技术：白帽和黑帽
SEO 常见技术：站内优化和站外优化
SEO 常见技术：roberts.txt
SEO 常见技术：网站地图
SEO 常见技术：统计分析
专题和其它
网站性能优化
性能优化和软件设计的关系
性能指标与关注点：业务角度
性能指标与关注点：资源角度
寻找性能瓶颈：思路
寻找性能瓶颈：工具
性能优化之产品和架构调整
性能优化之后端和持久层优化
性能优化之前端和网络层优化
全栈开发中的算法
流量控制算法
Diffie-Hellman 密钥交换
哈夫曼编码
RLE 编码
算数编码
分负的那些事儿
比较：物理分页和逻辑分页
比较：分页代码设计
比较：SQL 实现
重复数据的问题
XML、JSON、YAML 比较
比较：简洁和严谨
和 JavaScript 的爱和力
路径表达式
特殊字符处理
全栈时代：让全栈变味着更多
全栈时代：个人、纵向
全栈时代：个人、横向
全栈时代：项目和团队
特别叙述
全栈工程师的招聘
招人理念
面试流程
技术修炼的选择
技术分级：软能力
技术分级：模式和思想
技术分级：语言和平台
技术分级：框架和库
基础知识和能力
全栈团队的角色构成
全栈团队的定义
全栈团队的角色：华为
全栈团队的角色：Amazon
代码审查
审查的流程
常见的争议
审查的好处
审查的技巧
程序员学英语
为什么是次选项
学英语的策略
网络协议和 Web 接口
网络互联的昨天、今天和明天：HTTP 协议的演化
渐进：HTTP/0.9、HTTP/1.0、HTTP/1.1、HTTP/2
长连接和分块传输
动手实践：工具的使用——请求发送和抓包
为 HTTP 穿上盔甲：HTTPS
对称/非对称性加密，SSL/TLS 和 HTTPS
动手实践：捕获 TLS 握文
TLS 连接建立过程
证书有效验证
换个角度解决问题：服务端推送技术
比较：Pull 和 Push
服务端推送技术：Comet
服务端推送技术：WebSocket
动手实践：Chrome 下观察连接和推送消息
工整与自由的风格之争：SOAP 和 REST
比较：SOAP 和 REST
风格之争
动手实践：调用 RESTful API
权衡的艺术：漫谈 Web API 的设计
明确核心问题，确定问题域
结合实际需求和限制，选择承载技术
确定接口风格
定义具体接口形式
服务端 MVC
解耦是动态的主题：MVC 框架的发展
JSP 和 Servlet
渐进：JSP Model 1 / Model 2
比较：MVC 的一脉化，MVC 的变体
MVC 架构解析：模型（Model）
比较：贫血模型和充血模型
内部层次划分
CQRS 模式
MVC 架构解析：视图（View）
比较：服务端和客户端负面聚合技术
模板引擎的工作机制
动手实践：HTML 5 的模板标签
MVC 架构解析：控制器（Controller）
路径映射和视图指向
请求参数解析
参数验证
视图上下文解析
动手实践：实现一个简单的 MVC 系统
唯有套路得人心：Java EE 的模式
核心模式概览
拦截过滤器模式
数据访问对象
比较：MyBatis 和 Hibernate
创业编锋：面向切面编程
AOP
Spring 中的应用
比较：静态织入和动态代理
控制反转 IoC
动手实践：AOP 的运行时动态代理
前端技术
别有洞天：从后端到前端
前端技术的现状和意义
思维模式转变：应用事件驱动编程
思维模式转变：学写声明式代码
思维模式转变：培养交互思维
重剑无锋，大巧不工：JavaScript 面向对象
JavaScript 实现封装
JavaScript 实现继承
理解对象创建
函数成为一等公民
百花齐放，百家争鸣：前端 MVC 框架
渐进：前端 MVC 的变革
Angular：双向绑定、依赖注入、过滤器
React + Redux：JSX，Redux 的状态管理
不一样的体验：交互设计和页面布局
单页面应用
渐进增强和优雅降级
响应式布局
千言万语不及一幅画：谈谈数据可视化
比较：Web 绘图标准，SVG 和 Canvas
数据可视化的 JavaScript 库：Flot
数据可视化的 JavaScript 库：D3.js
打开潘多拉盒子：JavaScript 异步编程
渐进：用 Promise 系优化我套回调
用生成器来实现协程
异步错误处理

下面我把它展开来，你可以顺着这个展开的内容，回顾自己所学。

第一章：网络协议和 Web 接口

第一章是网络协议和 Web 接口，我以 HTTP 为核心，介绍了它的演进历史、相关技术，以及它的局限性：

对于安全传输方面的局限，我介绍了 HTTPS 的原理；

对于交互模式上的局限，我介绍了一些服务端推送技术；

对于无状态连接的局限，在第五章我介绍了客户端和服务端的会话。

也是从这一章开始，我们开始接触了 push 和 pull 这两个“对立”的套路，对于整个专栏，我们在各个层面的技术中把它们拿出来反复比较，权衡利弊。

对于 Web 接口部分，我从 SOAP 和 REST 所代表的两大设计风格开始，梳理了 Web 接口设计的过程，以及需要考虑的零零总总各个方面。

第二章：欢迎来到 MVC 的世界

第二章我主要针对 MVC 这个对于网站和其它 Web 应用开发来说，最重要的架构和设计模式，介绍演进、分层，并逐层仔细深挖：

模型层，我介绍了贫血模型和充血模型，以及常见的 CQRS 模式；

视图层，页面聚合是我们的重点，包括客户端聚合和服务端聚合，特别是模板引擎的工作原理；

控制器层，我把这一层拆分为几个方面，梳理了控制器在 MVC 架构中的工作步骤。

之后我们将 MVC 泛化，了解了其它重要的设计模式和套路，包括 AOP 和 IoC，以及实现切面编程所需的代理技术；还有其它著名的 JavaEE 模式，特别是拦截过滤器和数据访问对象模式。

第三章：从后端到前端

第三章主要讲前端技术。首先我从一个大体上俯瞰的角度，阐述了我所理解的前端技术的“不同”，特别是思维模式的不同。

然后我介绍了一些前端技术重要的知识点，比如 JavaScript 面向对象，包括封装、继承和多态的实现方式，也包括对象创建的原理。接着我以 React 和 Angular，以及它们的重要特性为例，介绍了百花齐放的前端 MVC 框架为我们带来的变革。

在页面设计和交互布局一讲中，我使用实际例子介绍了页面设计方面的一些原则和思路，包括渐进增强和优雅降级等等。

接着在数据可视化一讲中，我仔细比较了当今最重要的两种 Web 绘图标准，SVG 和 Canvas，并介绍了 Flot 和 D3.js 这两个可视化的库。

最后是 JavaScript 异步编程的技术，我们重新回归 JavaScript，我梳理了最重要的几个技术要点，这也是这一章相对比较难的一讲。

第四章：数据持久化

继续往技术栈的下方挖，数据持久化。在这一章的一开始，我首先介绍了这一层中互联网应用最广泛的技术之一——缓存，讲了它的本质、应用和常见的坑，以及缓存框架的设计要点。

接着是数据一致性，我介绍了它的概念，还有围绕它而产生的常见架构技术。有了数据一致性的基础知识，我们就进一步学习了 CAP 的原理，包括它的本质、常见的误区，以及我们该怎样根据 CAP 去做技术选型。

最后则是数据持久层的设计，包括理论部分、关系数据库和非关系数据库的一些设计要点；以及实例部分，包括几个范例系统：搜索引擎、地理信息系统和电商网站。

第五章：寻找最佳实践

在第五章中，我们跳出了全栈技术纵深方向上具体某一层的限制，而是从一个横向的角度去分析相关的实践技术。

在 Ops 三部曲中，我分别介绍了：

配置管理，有哪些配置管理的技术，我们该选择怎样的方式去做配置管理；

集群部署，我介绍了集群部署的方式，部署下的负载分担原理和策略算法，以及集群部署下常见的话题——Session 和 Cookie；

测试和发布，围绕 CI/CD 介绍了 pipeline 的含义、内容和挑战，以及不同类型的测试在其上集成的方式。

在网络安全那一讲中，我着重介绍了常见的 Web 攻击方式的原理，包括 XSS、CSRF、SQL 注入、HTTP 和 DNS 劫持，以及 DDoS。

最后则是 SEO 这一讲，我介绍了 SEO 和白帽、黑帽的含义，以及一些常见的 SEO 技术，包括站内优化和站外优化。

第六章：专题

在第六章，我选取了四大专题分别展开介绍。

在网站性能优化一讲，首先我介绍了一些基础知识，包括性能优化和软件设计的关系，性能优化的关注点，性能测试和指标，以及寻找性能瓶颈的思路；接着按照产品架构调整、后端和持久层优化，以及前端和网络层优化这样三个部分，分别介绍相应的常见性能优化技术。

在全栈开发中的算法这一讲，我以这样几大场景为主线，介绍了一些重要的算法：

流量控制算法，包括基于固定时间窗口和滑动时间窗口的流量控制，漏桶算法和令牌桶算法；

Diffie–Hellman 密钥交换算法，从中我们学到了数学上超大质数在安全领域的一个典型应用；

数据压缩算法，包括哈夫曼编码、RLE 编码和算术编码，从而帮助我们理解，如今繁多的压缩技术的最基本的原理。

在分页这一讲中，我讲了分页的分类和几种实现分页的技术，最后则是分页的常见问题——重复数据，它背后可能的原因和相应的解决方案。

在最后的 XML、JSON 和 YAML 这一讲，我把这几个数据承载格式的优劣进行了仔细的比较，包括简洁和严谨的程度，与 JavaScript 的亲和力，schema、转义方式和路径表达式等等。

其它

以上是技术方面的“硬通货”，对于非技术方面，我通过特别放送等等的形式，讨论了一些我认为比较重要，尤其是对于 Web 全栈工程师来说比较重要的话题。

- 怎样去理解 Web 全栈技术，全栈技术应该怎样学习，需要遵从怎样的学习策略；
- 北美全栈工程师的招聘，招人会秉持怎样的理念，整个流程是怎样的；
- 技术修炼应该怎样进行取舍和选择，哪些技术应该深挖，哪些则可以点到即止；
- 全栈团队的构成一般是怎样的，有哪些重要的角色；
- 代码审查为什么不可或缺，有哪些争议和好处，又有哪些技巧；
- 程序员为什么要学英语，重要性在哪里，又有哪些好办法；
- 作为 Web 全栈工程师，我们还有哪些发展方向可以考虑，我们为什么需要让项目和团队也“全栈化”？

继续提升的重要两步

对于我们专栏已经覆盖到的这些技能树的枝枝丫丫，我还想特别强调两个方面，这两个方面其实我们已经谈到过了，但是我觉得在今天，我们需要再拿出来稍稍强化一下。因为在我看来，它们有些特殊——它们对于程序员进一步发展至关重要，它们也是“成为更好的全栈工程师”所必须迈出的两步，但是谈论的人却很少。

1. 从做项目到做产品

你还记得吗？ [第 28 讲] 的选修课堂，我介绍了程序员独立性的几个阶段，不知道你属于哪一个阶段呢？作为职业生涯进阶的一部分，如果你还停留在“做项目”的阶段，那么你需要在某一天切换到“做产品”的模式上来，完成程序员独立性的升级。事实上，“做项目”，终归只是“做产品”整个过程中很小的一部分，只有对于产品多个阶段的不断地迭代、回馈，程序员很多方面的认识才能不断进化。

举个例子，对于程序员来说，oncall（定位和修复线上问题）就是“做产品”必经的一环，也是团队“吃自己的狗食”的重要一环。这个过程可以反哺开发阶段，没有这样的过程，就不会真正体会到代码质量的重要性，就不会彻底意识到单元测试的重要性，也不会对代码评审的必要性有深刻的认识。

因此，如果你还只有“做项目”的经验，我建议你，在未来给自己寻找做产品的机会。

2. 保持技术上的包容心

在 [第 14 讲] 我就谈到过这个给自己“贴标签”的问题，有些程序员工作没几年，无论是有意识还是无意识地，就已经给自己贴上了过于明确的技术分类的标签。

比如说，给自己贴上“PHP 工程师”的标签，理由是觉得 PHP 是最好的语言，因此求职的时候非 PHP 的岗位一概不考虑；比如，只想做 Billing 系统，其它的业务领域和项目一概不考虑；再比如，极度强调自己是“技术人”，而特别排斥管理技巧、沟通合作能力等这些被鄙视为“非干货”、“非实战”的内容，可是对于几乎所有的职业生涯技术路线来说，它们其实都是非常重要，且绕不开的槛。

再说说具体的技术，每个人都有偏好，但是不要因为自己的偏好，在技术的选择上失去程序员应有的理性判断，甚至将应当去接纳和学习的技术拒之门外。因为你也不知道哪一天，一个新领域、一门新技术会像浪潮一样涌来，就像几年前的云计算，这两年的区块链。只有那些有着包容心的程序员，才能去接纳和抓住机会。这也正如我在 [开篇词] 中所说的，最终也许我们需要“学得精”，但是一开始我们一定要“学得杂”。

送君千里，终须一别

任何一段旅程都有终点，这个专栏也不例外。

今天，你学到了这里，我最想大声地和你说一句话——祝贺你！从试读到购买，再到完整学习，我相信你已经一马当先，击败了一些人。现在，我希望你可以继续一马当先，不止在这个专栏，还在你今后的学习、工作和生活中。

回头看，整个专栏正文部分共计 40 讲，全部加起来有二十多万字。如果在一开始写专栏的时候跟我说，我会做到这个程度，我一定不相信。在这个过程当中，从主题选择、大纲确定、内容试写，到正式写作、绘图、审校、录音，整个过程确实让我颇为疲惫。一篇文章耗时最长的部分并非写作，而是内容选择和构思，比如怎样让你的收获最大化，得到更加通用的原理、套路，而不是停留在某个单一的技术表面上。但是，就像我在专栏上线以前想的那样，我觉得这是一件自己可以尽量认真做好的、很有意义的事情。现在看起来，我觉得可以稍微松口气了。

在这里，我要说，感谢你可以阅读这个专栏，并且特别感谢那些积极在留言区交流的朋友，关于全栈的内容，我们还可以继续在留言区沟通，也欢迎你访问我的个人博客，和我继续交流。

世界变化太快，技术发展太快，我们“被迫”成为了全栈工程师。

全栈，不只是一种技术分类，还代表了一种态度：理性、包容、好学。希望你可以从这个专栏得到你想要的，但更多的收获，需要你在线下继续努力。毕竟，《全栈工程师修炼指南》这个专栏结束了，而你的全栈之路却才刚刚开始。

好，我是四火，我们后会有期！

最后，文末有一份调查问卷，希望你可以抽出两分钟的时间填写一下。我会认真倾听你对这个专栏的意见或者建议，期待你的反馈！

精选留言 (7)

写留言

leslie
2019-12-13

"全栈，不只是一种技术分类，还代表了一种态度：理性、包容、好学。"其实老师说漏了一点"整体性"，这是从开始学到课程结束给我的收获。

还记得课前必读章节的学习路径篇和老师在留言区沟通过：对于全栈弱的是编程，毕竟十年没写程序代码了，确实偏弱；学习课程的同时其实我已经在极客大学的算法训练营去修炼了一回，刚好两边同时结束。...

展开

作者回复: 感谢阅读和评论

1

子豪sirius
2019-12-16

感谢老师为我们贡献这么一门优秀的课程，收获了很多。我工作快十年，最近也对自己的职业感到迷茫。学了课程后，发现自己有很多知识点和领域不足，平时工作也没有思考过，这篇课程很多地方启迪了。这篇课程还有很多没看懂的地方，会反复阅读，温故知新。下一步我会尝试构建自己的全栈知识体系，寻找职业的下一个目标

展开

作者回复:

小靓仔
2019-12-16

请问老师，您说的做项目是不是就是指单纯的功能开发，而做产品就是用心去做，不只是为了单纯的完成任务，上升到另一个层次，考虑的更多

展开

作者回复: 你可以看一下第28讲的选修课堂，做产品要比做项目包含多得多的内容，而不只是“用心”两个字而已。简单来说，做产品，需要在产品的整个生命周期做不同的工作，比如我举的这

个产品维护和线上问题修复的例子，这不是做一两个项目就能覆盖得到的。再比如说，单个项目交付可能就一个月的时间，但是整个产品的生命周期可能有好几年。

koko_zhk
2019-12-13

全栈：海纳百川，有容乃大

展开

靠人品去赢
2019-12-13

绝对值回票价，但是可惜不能再追专栏，有点不舍。

展开

作者回复: 谢谢

许童童
2019-12-13

感谢老师，给我们带来这么好的课程，一路跟着老师学下来，收获是真的不少，其实人生也像是做全栈，不能把自己局限在某一个范围内，勇敢的跨出自己的舒适区，向着未知去探索自己的边界，这也许就是全栈在人生上的意义吧。

展开

作者回复: 👍

tt
2019-12-13

感谢老师一路带给我的启发！

展开

作者回复: 不客气