33 | 桌面开发篇：回顾与总结

2019-08-16 许式伟

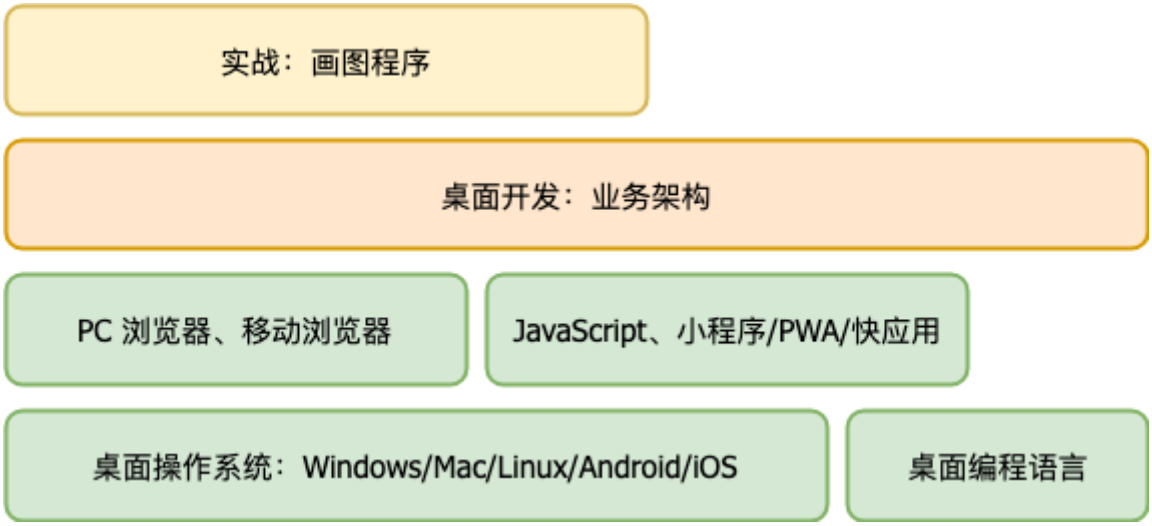
许式伟的架构课 进入课程 >

讲述：姚迪迈

时长 09:16　大小 8.50M

你好，我是七牛云许式伟。

到今天为止，我们第二章 “桌面开发篇” 就要结束了。今天，让我们对整章的内容做一个回顾与总结。本章我们主要涉及的内容如下。



这一章的内容主要分为三类。

一类是基础平台，也就是上图中的浅绿色背景部分，谈的是 Native 桌面操作系统和浏览器的演变过程。

一类是业务架构，也就是上图中的浅棕色背景部分，谈得是如何开发一个桌面软件。

最后一类是实战，也就是上图浅黄色背景部分，我们以画图程序作为例子谈业务架构，并对需求进行了多次的迭代。

通过本章的内容，我们总结一下桌面开发的特点。

首先从基础平台看。它的特点是：种类多、迭代快、知识有效期短。让桌面开发工程师（大前端）痛苦的是，时不时就有各种新平台、新语言、新框架冒出来，让人应接不暇。

其次从要开发的产品本身看。它的特点是：需求多、迭代快。桌面开发（大前端）负责的是和活生生的个体打交道，我们的开发人员需要为了功能丰富，体验便捷做各种努力。

为了让产品有竞争力，很多团队的发布周期都是至少一个月迭代一个版本，有的甚至是一周发布一个版本。而 Web 前端就更夸张了，一些公司甚至没有统一的发版概念，只要某个功能产品经理验收了，测试验收了，就可以发。

最后我们从对程序员的技能要求看。它的特点是门槛极低，但天花板又极高。

桌面开发（大前端）的代码量大，代码变更又很频繁，所以它对程序员的第一要求，不是质量，而是数量上的需求极大。为什么 GitHub 的语言排行榜总是 JavaScript 排名第一？这不是别的原因，是市场需求所致。

与之相对的，服务端开发则非常不同。服务端开发并不是一上来就有的，是互联网出现后产生的新分工。它并不负责用户交互，所以在需求提炼时可以做到极强的可预测性。因而服务端的第一挑战往往不是快速响应，而是性能和稳定性等质量需求。

桌面开发的客观需求量大，这决定了它的门槛要求必须极低。我在描述桌面开发的未来也提到过，桌面开发技术的演进方向，是 7-8 岁的儿童也可以开发生产级的应用。这是门槛低的极致状态。

但是为什么我又说桌面开发的天花板又极高呢？因为桌面开发的团队人数多、人员质量参差不齐、代码量大、迭代变更频繁，这意味着桌面软件工程项目的管理难度极高。所以桌面开发对架构师能力、软件工程的水平要求之高，要远高于服务端开发。

当然，从国内的现状来说，凡是堆人和加班可以解决的，最终都是用堆人和加班解决。架构师能力培养和软件工程能力提升？对大部分公司来说，他们的想法可能是：这太慢了，等不起。

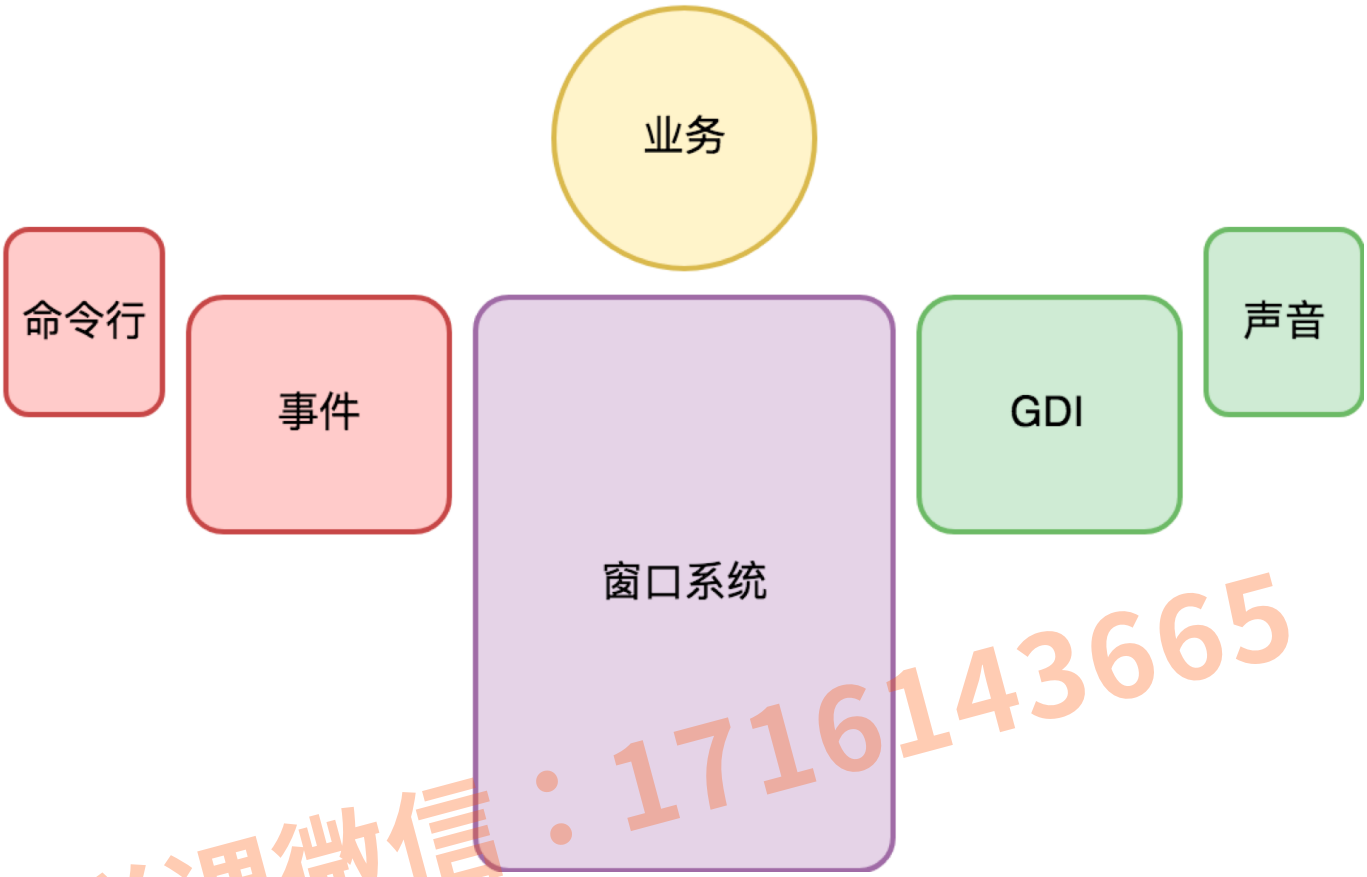
桌面开发篇的内容回顾

这一章前面我们讲了些什么？为了让你对第二章内容有个宏观的了解，我画了一幅图，如下。

我们首先从单机软件开发讲起。我们开篇第一讲首先回顾了桌面开发关于交互方式的变更。从最早命令行程序，到 2D/3D GUI 图形界面程序，到智能交互程序的萌芽。

为什么我们从交互变更谈起？因为这是桌面系统迭代的根源。每一次桌面系统大的变更周期，都是由一场新的交互革命所驱动。

随后，我们介绍了今天仍然处于主流地位的图形界面操作系统提供的编程框架。尽管使用接口各不相同，但是今天主流桌面操作系统的框架本质大同小异，都是基于事件分派做输入，GDI 做界面呈现。



互联网的出现，衍生出了浏览器，它支持了一种新的应用形态：Web 应用。这意味着在操作系统之上，产生了一个新操作系统。Web 应用也在演变，从静态页，到以 Gmail 为代表的 AJAX 应用，到 PWA，到小程序。

PC 浏览器之争已经结束，但移动浏览器的竞争才刚开始。

怎么做一个桌面程序？标准的套路是 MVC 架构。无论是单机还是 Web 应用，它都是适用的，只是 Web 程序需要考虑客户端与服务端的分工，需要引入网络协议。

跨平台开发，是桌面程序开发绕不过去的问题。几年前也许不明显，这得益于 Android 和 iOS 的垄断。但是现在又回到了群雄逐鹿的时期。Native 手机操作系统、传统 Web、众多的小程序种类、国际市场的 PWA 等等，需要综合考虑进行取舍。

聊完单机软件和 Web 应用，我们也探讨了桌面开发的未来趋势。桌面开发技术的演进，目标是越来越低的门槛，它和儿童编程教育相向而行，有一天必然汇聚于一点上。

为了让你更好地理解桌面开发的架构逻辑，我们引入了一个长达 5 讲的实战案例。这个案例建议深度消化。

为什么实战是很重要的？

学架构，我个人强调的理念是 “做中学”。

首先还是要勤动手。然后配合本专栏去思考和梳理背后的道理，如此方能快速进步。

我们不能把架构课学成理论课。计算机科学本身是一门实践科学，架构经验更是一线实战经验的积累和总结。

通过这个实战案例，我们也探讨了辅助界面元素，也就是控件的架构。控件架构没有什么特别的地方，唯一需要注意的是支持多实例。用多实例去思考你的应用程序架构的合理性，会有助于你对架构设计中的一些决策提供帮助。

当然更重要的，其实是让你有机会形成更好的架构设计规范。

作为最后收官，我们聊了架构第二步：系统的概要设计，简称系统设计。我们这个阶段关注的是全局性的风险，怎么保证项目可以按时、按质、高度并行化地被执行。

系统架构打的是地基。

这个阶段需要选择操作系统、选择语言、选择主框架，选择项目所依赖的最核心的基础设施。这就是我说的有关于基础架构的工作。

这个阶段也需要分解业务系统。我们一般以子系统为维度来阐述系统各个角色之间的关系。对于关键的子系统，我们还会进一步分解它，甚至详细到把该子系统的所有模块的职责和接口都确定下来。

这个阶段我们的核心意图并不是确定系统完整的模块列表，我们的焦点是整个系统如何被有效地串联起来。如果某个子系统不作进一步的分解也不会在项目上有什么风险，那么我们并不需要在这个阶段对其细化。

为了降低风险，概要设计阶段也应该有代码产出。

这样做的好处是，一上来我们就关注了全局系统性风险的消除，并且给了每个子系统或模块的负责人一个更具象且确定性的认知。

代码即文档。代码是理解一致性更强的文档。

桌面开发篇的参考资料

桌面开发的知识迭代更新非常快，所以很难去列经典书籍。

这里我列一下我认为值得重点关注的技术：

JavaScript。毫无疑问，这是当前桌面开发的第一大语言，务必要精通。这方面我推荐程劭非（winter）的极客时间专栏“重学前端”。

微信小程序。这方面资料比较少，我推荐高磊的极客时间视频课“9 小时搞定微信小程序开发”。

React 和 Vue。这应该当前比较知名的两大前端框架，可以学习一下。前者可以看下王沛的“React 实战进阶 45 讲”，后者可以看下唐金州的“Vue 开发实战”。

Flutter 和 SwiftUI。这两个技术很新，其中 Flutter 已经有一些资料，比如陈航的“Flutter 核心技术与实战”。SwiftUI 与 Swift 语言关联很紧，在张杰的“Swift 核心技术与实战”中有所涉略。

PWA 和 WebAssembly。这方面图书还比较少，不妨看官方材料结合实战来学习。

当然，经典的 Android、iOS 方面的开发资料，也值得看看。这方面资料非常多，我就不再去提名了。

结语

今天我们对本章内容做了概要的回顾，并借此对整个桌面开发的骨架进行了一次梳理。

这一章我们开始聊业务架构。学业务架构最好的方式是：“做中学”。做是最重要的，然后要有做后的反思，去思考并完善自己的理论体系。

如果你对今天的内容有什么思考与解读，欢迎给我留言，我们一起讨论。下一讲我们开始进入第三章：服务端开发篇。

如果你觉得有所收获，也欢迎把文章分享给你的朋友。感谢你的收听，我们下期再见。

上一篇　32 | 架构：系统的概要设计

精选留言 (12)

写留言

靠人品去赢
2019-08-16

关于对人加班以及门槛低天花板高的总结，简直真实的一匹。

2

CALL ME
2019-08-18

很期待后续的内容，感谢前辈的付出。

展开

亚宁
2019-08-18

许老师，辛苦了。

展开

Jian

2019-08-17

从需求着眼，讲到技术的发展，揭示了造成前端技术现状的原因。层层递进，最后还推销了一把Geek的课程----简直润物细无声，老师注定要成为---技术布道人

展开

leslie

2019-08-16

前进中提升：做这行停不下来的事情只有学习和实践；跟着课程一起学习、实践，提升自己的实力。

许童童

2019-08-16

跟着老师一起精进。

展开

Smallfly

2019-08-16

辅助元素的多实例设计应该怎么理解，比如说相同样式的辅助元素，只是数据源不同，是否需要多实例呢？（在辅助元素不会同时显示的前提下）

不是特别理解多实例和应用架构之间的具体关系。

展开

作者回复: 如果应用存在单例性质的全局变量，就比较难做到多实例。

Aaron Cheung

2019-08-16

打卡33 详实的总结

展开

义明
2019-08-16

老师，关于go语言常用的几种设计模式有介绍吗？如果编写游戏服务端相关的项目是否有开源项目可以参考？

作者回复: 一般设计模式是领域性的，和语言关系不大。游戏领域我关注不多。

海贼王
2019-08-16

老师您这么多年肯定有好多经验，晚辈想向您请教，面对未知的流量暴增，可以预先怎么处理？？？？

作者回复: 这个后面服务端开发会涉及

黄强
2019-08-16

从头跟到现在最有感触的还是许老师的那句，思考中记忆，一边看一边思考一边梳理这10几年来的经历，感触良多，收获良多，谢谢许老师！

展开

Geek_架构师
2019-08-16

咦，参考资料怎么没Qt呢？

展开