64 | 不断完善的架构范式

2019-12-13 许式伟

许式伟的架构课 进入课程 >

讲述：姚迪迈

时长 09:11 大小 8.42M

你好，我是七牛云许式伟。

我们在 “62 | 重新认识开闭原则 (OCP)” 这一讲中介绍了开闭原则。这篇内容非常非常重要，可以说是整个架构课的灵魂。

总结来说，开闭原则包含以下两层含义：

第一，模块的业务要稳定。模块的业务遵循 “只读” 设计，如果需要变化不如把它归档，放弃掉。这种模块业务只读的思想，是架构治理的基础哲学。我平常和小伙伴们探讨模块边界的时候，经常会说这样一句话：

> 每一个模块都应该是可完成的。

这实际上是开闭原则的业务范畴 “只读” 的架构治理思想的另一种表述方式。

第二，模块业务的变化点，简单一点的，通过回调函数或者接口开放出去，交给其他的业务模块。复杂一点的，通过引入插件机制把系统分解为 “最小化的核心系统 + 多个彼此正交的周边系统”。事实上回调函数或者接口本质上就是一种事件监听机制，所以它是插件机制的特例。

上一讲我们介绍了接口设计。到此为止，我们的架构思维篇也已经基本接近尾声。可能有人会越来越奇怪，为什么我还是没有去聊那些大家耳熟能详的架构设计原则？

实际上，并不是这些架构设计原则不好，它们之中不乏精彩绝伦、振聋发聩的总结。比如：

- 接口隔离原则（Interface Segregation Principle，ISP）：一个模块与另一个模块之间的依赖性，应该依赖于尽可能小的接口。
- 依赖倒置原则（Dependence Inversion Principle，DIP）：高层模块不应该依赖于低层模块，它们应该依赖于抽象接口。
- 无环依赖原则（Acyclic Dependencies Principle，ADP）：不要让两个模块之间出现循环依赖。怎么解除循环依赖？见上一条。
- 组合 / 聚合复用原则（Composition/Aggregation Reuse Principle，CARP）：当要扩展功能时，优先考虑使用组合，而不是继承。
- 高内聚与低耦合（High Cohesion and Low Coupling，HCLC）：模块内部需要做到内聚度高，模块之间需要做到耦合度低。这是判断一个模块是在做一个业务还是多个业务的依据。如果是在做同一个业务，那么它所有的代码都是内聚的，相互有较强的依赖。
- 惯例优于配置（Convention over Configuration，COC）：灵活性会增加复杂性，所以除非这个灵活性是必须的，否则应尽量让惯例来减少配置，这样才能提高开发效率。如有可能，尽量做到 “零配置”。
- 命令查询分离（Command Query Separation，CQS）：读写操作要分离。在定义接口方法时，要区分哪些是命令（写操作），哪些是查询（读操作），要将它们分离，而不要揉到一起。

关注点分离（Separation of Concerns，SOC）：将一个复杂的问题分离为多个简单的问题，然后逐个解决这些简单的问题，那么这个复杂的问题就解决了。当然这条说了等于没说，难在如何进行分离，最终还是归结到对业务的理解上。

这些都是很好很好的。但是，我们需要意识到的一点是，熟读架构思维并不足以让我们成为优秀的架构师。

要始终记住的一点是，我们做的是软件工程。软件工程的复杂性它自然存在，不会因为好的架构思维而消除。

所以虽然理解架构思维是非常重要的，但是架构师真正的武器库并不是它们。

那么架构师的武器库是什么？

这就要从 “架构治理” 开始谈起。

前面我们说过，“开闭原则” 推崇模块业务 “只读” 的思想，是很好的架构治理哲学。它告诉我们，软件是可以以 “搭积木” 的方式搭出来的。

核心的一点是，我们如何形成更多的 “积木”，即一个个业务只读、接口稳定、易于组合的模块。

所以，真正提高我们工程效率的，是我们的业务分解能力和历史积累的成果。

前面我们说过，架构分解中有两大难题：其一，需求的交织。不同需求混杂在一起，也就是存在所谓的全局性功能。其二，需求的易变。不同客户，不同场景下需求看起来很不一样，场景呈发散趋势。

在 “61 | 全局性功能的架构设计” 这一讲我们重点聊的是第一点。对于全局性功能怎么去拆解，把它从我们的业务中剥离出来，并无统一的解决思路。

但好的一点是，绝大部分全局性功能都会有很多人去拆解，并最终会被基础设施化。所以具体业务中我们会碰到的全局性功能并不会非常多。

比如，怎么做用户的鉴权？怎么保障软件 24 小时持续服务？怎么保障快速定位用户反馈的问题？这些需求和所有业务需求是交织在一起的，也足够普适，所以就会有很多人去思考对应的解决方案。

作为架构师，心性非常重要。

架构师需要有自己的信仰。我们需要坚持对业务进行正交分解的信念，要坚持不断地探索各类需求的架构分解方法。这样的思考多了，我们就逐步形成了各种各样的架构范式。

这些架构范式，并不仅仅是一些架构思维，而是 “一个个业务只读、接口稳定、易于组合的模块 + 组合的方法论”，它们才是架构师真正的武器库。

这个武器库包含哪些内容？

首先，它应该包括信息科技形成的基础架构。努力把前辈们的心血，变成我们自己真正的积累。光会用还不够，以深刻理解它们背后的架构逻辑，确保自己与基础架构最大程度上的 “同频共振”。

只有让基础架构完全融入自己的思维体系，同频共振，我们才有可能在架构设计需要的时候 “想到它们”。这一点很有趣。有些人看起来博学多才，头头是道，但是真做架构时完全想不到他的 “博学”。

从体系结构来说，这个基础架构包含哪些内容？

其一，基础平台。包括：冯·诺依曼体系、编程语言、操作系统。

其二，桌面开发平台。包括：窗口系统、GDI 系统、浏览器与小程序。当然我们也要理解桌面开发背后的架构逻辑，MVC 架构。

其三，服务端开发平台。包括：负载均衡、各类存储中间件。服务端业务开发的业务逻辑比桌面要简单得多。服务端难在如何形成有效的基础架构，其中大部分是存储中间件。

其四，服务治理平台。主要是以容器技术为核心的 DCOS（数据中心操作系统），以及围绕它形成的整个服务治理生态。这一块还在高速发展过程中，最终它将让服务端开发变得极

其简单。

理解了这些基础架构，再加上你自己所处行业的领域知识，基本上设计出一个优秀业务系统，让它健康运行，持续不间断地向用户提供服务就不是问题。

读到这里，你可能终于明白，为什么这个架构课的内容结构是目前这个样子组织的。因为消化基础架构成为架构师自身的本领，远比消化架构设计原则，架构思维逻辑要难得多。

消化基础架构的过程，同时也是消化架构思维的过程。

把虚的事情往实里做，才有可能真正做好。

理解了基础架构，剩下的就是如何沉淀业务架构所需的武器库。这一般来说没有太统一的体系可以参考，如果有，大部分都会被基础设施化了。

所以，业务只能靠你自己的架构设计能力去构建它。而这，其实也是架构师的乐趣所在。

还没有被基础设施化但比较通用的，有一个大门类是数据相关的体系。数据是软件的灵魂。它可能包括以下这些内容：

存盘与读盘（IO）；

文本处理；

存储与数据结构；

Undo/Redo；

……

我们在下一讲，会专门聊聊其中的 “文本处理” 这个子课题。

从完整性讲，我们的架构课并没有包括所有的基础架构。我们把话题收敛到了 “如何把软件跑起来，并保证它持续健康运行” 这件事情上。

但从企业的业务运营角度来说，这还远不是全部。 “54 | 业务的可支持性与持续运营” 我们稍稍展开了一下这个话题。但要谈透这个话题，它会是另一本书，内容主题将会是

“数据治理与业务运营体系构建”。

我希望有一天能够完成它，但这可能要很久之后的事情了。它是我除架构课外的另一个心愿。

结语

我们在 “56 | 服务治理篇：回顾与总结” 这一讲，也就是第四章结束的时候，谈到我们下一章的内容时提到：

> 我个人不太喜欢常规意义上的 “设计模式”。或者说，我们对设计模式常规的打开方式是有问题的。理解每一个设计模式，都应该放到它想要解决的问题域来看。所以，我个人更喜欢的架构范式更多的是 “设计场景” 的总结。“设计场景” 和设计模式的区别在于它有自己清晰的问题域定义，是一个实实在在的通用子系统。

> 是的，这些 “通用的设计场景”，才是架构师真正的武器库。如果我们架构师总能把自己所要解决的业务场景分解为多个 “通用的设计场景” 的组合，这就代表架构师有了极强的架构范式的抽象能力。而这一点，正是架构师成熟度的核心标志。

结合今天这一讲我们聊的内容，相信你对这段话会有新的理解。

“开闭原则” 推崇模块业务 “只读” 的思想，是很好的架构治理哲学。它告诉我们，软件是可以以 “搭积木” 的方式搭出来的。核心的一点是，我们如何形成更多的 “积木”，即一个个业务只读、接口稳定、易于组合的模块。

结合今天这一讲的内容，相信你终于完全能理解我们这个架构课的内容组织为什么是现在你看到的样子了。

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。下一讲我们的话题是 “架构范式：文本处理”。

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　63 | 接口设计的准则

下一篇　65 | 架构范式：文本处理

精选留言 (11)

写留言

Sam
2019-12-13

跟着许大这一路走来，让我在架构认知上得到前所未有的提高，更是不断的遵循“架构是门实践”的学课原则，不断的在工作中实践。眼看课程就要结束了，真担心在课程结束后遇到无法解决的架构难题时找不到方向，希望课程结束后还能以某种方式与老师保持联系和沟通。

展开

2

轶名
2019-12-13

有种拨开云雾见月明的感觉

展开

1

歌在云端
2019-12-13

买了这么多课程，感觉还是许老师的讲的最好，最有深度

1

靠人品去赢
2019-12-13

什么时候也能自己负责一个产品的架构，从业务上考虑但是呢还能在开闭原则上有考虑，不至于业务方向一变就要开始打补丁。

展开

作者回复: 加油

1

知行合一
2019-12-18

有深度，我理解不是很到位，感觉自己还得再从头刷一遍

沫沫（美丽人生）
2019-12-15

许老师好。数据治理和运营体系搭建，能否先出一个1.0版本呢，一定会给这个行业带来巨大的价值。谢谢。

Aaron Cheung
2019-12-14

业务的正交分解 理论知道了很多还需要实践 补打卡

展开

梦醒十分
2019-12-14

出书吧！买来经常复习!适合多看几遍。

展开

leslie

2019-12-13

许老师的课一路跟随至今除了桌面开发平台这块确实有些前端，没有去扩展；其它的基础架构不少是这些年工作中经历过然后觉得自己太浅或深度不够都进行了扩展，然后不知不觉中发现自己每个月都处在高压学习和自我总结中；年末看自己的学习课程发现突然比预计多了差不多10门，辛苦却收益满满。

关于老师提及的两块提出一些个人的学后感:...

展开

作者回复: 有付出必有回报

吴

2019-12-13

学习这门课收获很多

展开

:)

2019-12-13

希望老师能把这门课出本书啊！

展开