01讲为什么说每个程序员都要尽早地学习并掌握设计模式相关知识

我相信，很多程序员都已经意识到基础知识的重要性，觉得要夯实基础，才能走得更远，但同时对于如何将基础知识转化成开发“生产力”仍然有些疑惑。所以，你可能看了很多基础的书籍，比如操作系统、组成原理、编译原理等，但还是觉得很迷茫，觉得在开发中用不上，起码在平时的CRUD业务开发中用不上。实际上，这些基础的知识确实很难直接转化成开发“生产力”。但是，它能潜移默化地、间接地提高你对技术的理解。

不过，我觉得，设计模式和操作系统、组成原理、编译原理等这些基础学科是不一样的。它虽然也算是一门基础知识，但是它和数据结构、算法更像是一道儿的，相比那些更加基础的学科，设计模式能更直接地提高你的开发能力。我在开篇词里也说了，如果说数据结构和算法是教你如何写出高效代码，那设计模式讲的是如何写出可扩展、可读、可维护的高质量代码，所以，它们跟平时的编码会有直接的关系，也会直接影响到你的开发能力。

不过，你可能还是会觉得设计模式是把屠龙刀，看起来很厉害，但平时的开发根本用不上。基于这种观点，接下来，我们就具体地聊一聊，我们为什么要学习设计模式?

1. 应对面试中的设计模式相关问题

学习设计模式和算法一样，最功利、最直接的目的，可能就是应对面试了。

不管你是前端工程师、后端工程师，还是全栈工程师，在求职面试中，设计模式问题是被问得频率比较高的一类问题。特别是一些像BAT、TMD这样的大公司，比较重视候选人的基本功，经常会拿算法、设计模式之类的问题来考察候选人。

所以，我在求职面试的时候，都会提前准备、温习一遍设计模式。尽管并不是每次面试都会被问到，但一旦被问到，如果回答得不好，就是一个败笔，这场面试基本上也就凉凉了。所以，为了保证万无一失，摆脱一旦被问到答不出来的窘境，对于设计模式这种大概率被问到的问题，我都会未雨绸缪，提前准备一下。

当然，我并不是临时抱佛脚。我平时就比较重视设计模式相关知识的积累，所以底子比较好，只需要在每次面试前花很短的时间，重新温习一下，便可以自信满满地去面试，而不是心里老是担心被问到，影响正常的面试发挥。

所以，如果你也不想让设计模式相关问题成为你面试中的短板，那跟着我把专栏中的知识点都搞清楚，以后面试再遇到设计模式相关的问题，就不会惧怕了，甚至还会成为你面试中的亮点。

2. 告别写被人吐槽的烂代码

我们经常说，“Talk is cheap，show me the code。”实际上，代码能力是一个程序员最基础的能力，是基本功，是展示一个程序员基础素养的最直接的衡量标准。你写的代码，实际上就是你名片。

尽管我已经工作近十年，但我一直没有脱离编码一线，现在每天也都在坚持写代码、review指导同事写代码、重构遗留系统的烂代码。这些年的工作经历中，我见过太多的烂代码，比如命名不规范、类设计不合理、分层不清晰、没有模块化概念、代码结构混乱、高度耦合等等。这样的代码维护起来非常费劲，添加或者修改一个功能，常常会牵一发而动全身，让你无从下手，恨不得将全部的代码删掉重写！

当然，在这些年的工作经历中，我也看到过很多让我眼前一亮的代码。每当我看到这样的好代码，都会立刻对作者产生无比的好感和认可。且不管这个人处在公司的何种级别，从代码就能看出，他是一个基础扎实的高潜员工，值得培养，前途无量！因此，代码写得好，能让你在团队中脱颖而出。

所以，我的专栏，不仅仅只是讲解设计模式，更加重要的是，我会通过实战例子，手把手教你如何避免刚刚提到的代码问题，告别被人诟病的烂代码，写出令人称道的好代码，成为团队中的代码标杆！而且，写出一份漂亮的代码，你自己也会很有成就感。

3. 提高复杂代码的设计和开发能力

大部分工程师比较熟悉的都是编程语言、工具、框架这些东西，因为每天的工作就是在框架里根据业务需求，填充代码。实际上，我刚工作的时候，也是做这类事情。相对来说，这样的工作并不需要你具备很强的代码设计能力，只要单纯地能理解业务，翻译成代码就可以了。

但是，有一天，我的leader让我开发一个跟业务无关的比较通用的功能模块，面对这样稍微复杂的代码设计和开发，我就发现我有点力不从心，不知从何下手了。因为我知道只是完成功能、代码能用，可能并不复杂，但是要想写出易扩展、易用、易维护的代码，并不容易。

如何分层、分模块？应该怎么划分类？每个类应该具有哪些属性、方法？怎么设计类之间的交互？该用继承还是组合？该使用接口还是抽象类？怎样做到解耦、高内聚低耦合？该用单例模式还是静态方法？用工厂模式创建对象还是直接new出来？如何避免引入设计模式提高扩展性的同时带来的降低可读性问题？……各种问题，一下子挤到了我面前。

而我当时并没有对设计模式相关的知识（包括设计模式、设计原则、面向对象设计思想等）有太多的了解和积累，所以一时间搞得我手足无措。好在因此我意识到了这方面知识的重要性，所以在之后很多年的开发中，我都一直刻意锻炼、积累这方面的能力。面对复杂代码、功能、系统的设计和开发，我也越来越得心应手，游刃有余。写出高质量代码已经成为了我的习惯，不经意间写出来的代码，都能作为同事学习、临摹的范例，这也成为了我职场中最引以为豪的亮点之一。

4. 让读源码、学框架事半功倍

对于一个有追求的程序员来说，对技术的积累，既要有广度，也要有深度。很多技术人早早就意识到了这一点，所以在学习框架、中间件的时候，都会抽空去研究研究原理，读一读源码，希望能在深度上有所积累，而不只是略知皮毛，会用而已。

从我的经验和同事的反馈来看，有些人看源码的时候，经常会遇到看不懂、看不下去的问题。不知道你有没有遇到过这种情况？实际上，这个问题的原因很简单，那就是你积累的基本功还不够，你的能力还不足以看懂这些代码。为什么我会这么说呢？

优秀的开源项目、框架、中间件，代码量、类的个数都会比较多，类结构、类之间的关系极其复杂，常常调用来调用去。所

以，为了保证代码的扩展性、灵活性、可维护性等，代码中会使用到很多设计模式、设计原则或者设计思想。如果你不懂这些设计模式、原则、思想，在看代码的时候，你可能就会琢磨不透作者的设计思路，对于一些很明显的设计思路，你可能要花费很多时间才能参悟。相反，如果你对设计模式、原则、思想非常了解，一眼就能参透作者的设计思路、设计初衷，很快就可以把脑容量释放出来，重点思考其他问题，代码读起来就会变得轻松了。

实际上，除了看不懂、看不下去的问题，还有一个隐藏的问题，你可能自己都发现不了，那就是你自己觉得看懂了，实际上，里面的精髓你并没有get到多少！因为优秀的开源项目、框架、中间件，就像一个集各种高精尖技术在一起的战斗机。如果你想剖析它的原理、学习它的技术，而你没有积累深厚的基本功，就算把这台战斗机摆在你面前，你也不能完全参透它的精髓，只是了解个皮毛，看个热闹而已。

因此，学好设计模式相关的知识，不仅能让你更轻松地读懂开源项目，还能更深入地参透里面的技术精髓，做到事半功倍。

5. 为你的职场发展做铺垫

普通的、低级别的开发工程师，只需要把框架、开发工具、编程语言用熟练，再做几个项目练练手，基本上就能应付平时的开发工作了。但是，如果你不想一辈子做一个低级的码农，想成长为技术专家、大牛、技术leader，希望在职场有更高的成就、更好的发展，那就要重视基本功的训练、基础知识的积累。

你去看大牛写的代码，或者优秀的开源项目，代码写得都非常的优美，质量都很高。如果你只是框架用得很溜，架构聊得头头是道，但写出来的代码很烂，让人一眼就能看出很多不合理的、可以改进的地方，那你永远都成不了别人心目中的“技术大牛”。

再者，在技术这条职场道路上，当成长到一定阶段之后，你势必要承担一些指导培养初级员工、新人，以及code review的工作。这个时候，如果你自己都对“什么是好的代码？如何写出好的代码？”不了解，那又该如何指导别人，如何让人家信服呢？

还有，如果你是一个技术leader，负责一个项目整体的开发工作，你就需要为开发进度、开发效率和项目质量负责。你也不希望团队堆砌垃圾代码，让整个项目无法维护，添加、修改一个功能都要费老大劲，最终拉低整个团队的开发效率吧？

除此之外，代码质量低还会导致线上bug频发，排查困难。整个团队都陷在成天修改无意义的低级bug、在烂代码中添补丁的事情中。而一个设计良好、易维护的系统，可以解放我们的时间，让我们做些更加有意义、更能提高自己和团队能力的事情。

最后，当你成为leader、或者团队中的资深工程师、技术专家之后，你势必要负责一部分团队的招聘工作。这个时候，如果你要考察候选人的设计能力、代码能力，那设计模式相关的问题便是一个很好的考察点。

不过，我也了解到，很多面试官实际上对设计模式也并不是很了解，只能拿一些简单的单例模式、工厂模式来考察候选人，而且所出的题目往往都脱离实践，比如，如何设计一个餐厅系统、停车场系统、售票系统等。这些题目都是网上万年不变的老题目，几乎考察不出候选人的能力。在我的专栏中，有200多个真实项目开发中的设计模式相关问题，你跟着看下来，足以让你成为设计模式方面的大牛，再来面试候选人的时候，就不用因为题目老套、脱离实践而尴尬了！

重点回顾

今天，我们讲了为什么要学习设计模式相关的知识，总结一下的话，主要有这样五点：应对面试中的设计模式相关问题；告别写被人吐槽的烂代码；提高复杂代码的设计和开发能力；让读源码、学框架事半功倍；为你的职场发展做铺垫。

投资要趁早，这样我们才能尽早享受复利。同样，有些能力，要早点锻炼；有些东西，要早点知道；有些书，要早点读。这样在你后面的生活、工作、学习中，才能一直都发挥作用。不要等到好多年后，看到了，才恍然大悟，后悔没有早点去学、去看。

设计模式作为一门与编码、开发有着直接关系的基础知识，是你现在就要开始学习的。早点去学习，以后的项目就都可以拿来锻炼，每写一行代码都是对内功的利用和加深，是可以受益一整个职业生涯的事情。

课堂讨论

今天课堂讨论的话题有两个：

1. 聊一聊你对设计模式相关知识的重要性的看法；
2. 在你过往的项目开发中，有没有用过某种设计模式？是在什么场景下应用的？解决了什么问题？

欢迎在留言区发表你的观点，积极参与讨论。你也可以把这篇文章分享给你的朋友，邀请他一起学习。

精选留言

丁丁历险记

笔记：

1 作者反复解释了下学好dp 的重要性。

映像深刻的：

基本功不够，把一台战斗机放你面前，你都不知道如何欣赏和品味。

其它职能：

1 面试所需（适当的区分度）

2 告别烂代码，让实现优雅起来。老司机后，要参与指导菜鸟，也要会。

3 show me the code ,你牛不牛，终归还需要代码的展现，把框架说得头头是道又如何，技术看技术，硬核不行，外表的东西白搭。没法成为别人心中的大牛的。

作业：聊看法。

一句话，简直太tmd 的重要了。

以前重构过一个p2p 客户投资后奖励活动【放心，平台未跑路，老板是用心做事的人】。 刚开始，他们真的是 if eles 的去写每一个活动。

我去了后。主要就是参考yii 框架的实现方法。

做了以下解藕，把购买后的奖励分为四块。

通过配置 rules 来确认是否有奖励资格。 【首次，】

清算出奖多少，奖给谁（通常会带上推荐人）【固定额，阶梯算法，比例值，vip 等级等】

创建出奖励执行类， （红包，现金，抽奖券， 积分等） 并执行奖励

发送通知 （站内信，短信，微信，邮件， ） （通知会在通知里挂接广告）

离开那公司时特意查了一下，公司共发布了1700 条个奖励项，给客户返利约900 万。

2019-11-04 17:55

嚴脂红.*

做游戏开发相关的工作，日常用到非常多的设计模式，比如:

对于游戏的设置，ui和scene等等各种manager管理类都要用到单例模式。

创建游戏中各种角色的各种工厂模式还有对象池。

处理游戏角色的各种状态的有限状态机要用到状态模式。

在优化复杂游戏场景时会用到享元模式。

还有游戏引擎本身就用到的组件模式。

.....

2019-11-04 19:10

jony

设计模式，两眼一抹黑，从0开始

2019-11-05 00:48

乐

个人认为设计模式主要解决的是扩展和耦合问题

日常使用：

使用代理模式进行共性化处理，比如说 AOP 思想，将非业务功能和业务功能解耦

* 事务的处理@Translation
* 系统间上下文的传递 ThreadLocal + restTemplate#intercept 等等

使用工厂 + 策略

* 不同优惠种类的计算
* 定制化功能的解耦

观察者模式：这个模式的思想，我觉得非常的重要，你可以在许多中间件（mq、zookeeper、netty 等等）乃至生活中都能看到它的影子

* 通过领域事件解耦业务
* 理解 eventloop、epoll 等等
* 通过 watch 实现动态配置、HA 等等

责任链模式：pipeline 思想

* filter
* 理解 netty 中的各种 handler

2019-11-04 20:24

谭棋钊

只用过单例

2019-11-04 17:41

梁军

个人感觉，设计模式和数据结构算法结合，再去读源码或者框架，很清晰，一个招式一个内功

2019-11-04 19:08

编程界的小学生

1.设计模式很主要，很主要，很主要。他是一个能够长久迭代开发的必要条件，也是一个提高开发效率的必要条件。当我第一次用设计模式的时候我激动的一晚上没睡好，反反复复去看我的代码，喜欢的不得了。我会对照我的代码思考需求变更我的代码应该改什么，那种解耦合，可扩展的架构真的喜欢！！！

2.单例、工厂、模板、策略
基本的套路就是：单例的工厂类负责创建策略类，但是每个策略类都有共同特性，所以用到了模板模式。
类关系就是每个策略类实现策略接口并继承模板类。交由单例的工厂来管理。
也有人说这就是模板。跟策略没关，但我认为确实也是策略。
场景:医疗系统，药品分为中药，西药，医疗器械等等不同类别，每种类别计算价格方式由相同的算法和不同的算法组成，所以我用了模板和策略。
补充：其实最后我发现spring有依赖搜索，直接注入map就行了。完全不用自己写工厂管理

2019-11-04 21:14

仰望星空

老师能不能讲讲函数式编程思想，设计模式都是基于面向对象的，而现在更流行函数式编程。

2019-11-04 20:05

作者回复

函数式编程感觉还是没面向对象流行 所以专栏中没讲

2019-11-05 11:00

大王拍我去巡山

CRUD写的太多了， 通过设计模式也能简化业务代码，释放生产力。期望能多有些代码场景的演练，跟随学习

2019-11-05 00:05

汤小高

课后讨论：我的看法

1.聊一聊你对设计模式相关知识的重要性的看法；

（1）设计模式能让程序员编写出可读性高，易维护，易拓展的代码，避免烂代码。

（2）利用好设计模式能让复杂功能的设计的代码可复用性，可拓展性，可维护性，可读性更高。达到高内聚、低耦合的目的。

（3）设计模式能提高程序员的自我修养

2.在你过往的项目开发中，有没有用过某种设计模式？是在什么场景下应用的？解决了什么问题？

用到过模板模式，单例模式。

2.1）模板模式应用场景：在一个项目的规则引擎中，一个规则引擎有一系列规则过滤，这个过滤步骤基本上是确定的，只是某些步骤在不同的场景下需要相互替换，模板方法定义了方法调用顺序，需要用到一个钩子，让子类去实现这个方法。

模板模式解决问题：解决了以后可拓展的问题，如果以后需要在新场景下新增规则方法，只需新增一个类，实现钩子方法即可，不需改动既有代码。

2.2）单例模式应用场景：用于加载项目中需要的配置文件的资源类。

单例模式解决问题：解决了资源共用，避免创建出大量资源对象，节省了JVM内存资源。

2019-11-04 21:58

jiji87432

1、设计模式的重要性看法：

(1)可以使得代码编写更优雅

(2)对学习一些开源框架有很大帮助

(3)程序的可扩展性、维护性更好，系统解耦

2、基于上面的看法结合实际项目说明：

(1)在新老系统接割的项目中，进行数据回写，用到了策略模式，根据不同的接口码值，进行不同的策略回写。

(2)在银行前置系统中，用到了模板设计模式，对接不同的三方通道，在模板中定义某些特定的步骤，并加上hook，具体步骤子类实现。

(3)责任链模式，在网关系统中用到的filter就是责任链模式，把一个请求依次的在过滤器链上进行传输

(4)装饰器模式，在普通的微服务提供的restful接口中，在请求返回的时给对象添加一些额外的职责

(5)门面设计模式，对外提供统一的接口，例如在springcloud中提供统一的feinclient接口，所有外部系统都通过feinclient接口进行接入，从而不关心内部接口的调用实现。

(6)还有命令设计模式，原来在看工作流的时候不知道这个模式，看的云里雾里，后来才知道原来整个框架用到了命令设计模式，一下子就明朗了很多。所以学习设计模式很重要。

还有单例模式、工厂模式等，以上大概就是在工作中用到的设计模式，分享一下，如有不对请指正。

2019-11-11 17:03

条

1.个人觉得设计模式就是对一些写比较好的代码的总结，其实代码看的写的多了，也能写出自己的设计模式。

2.最近重构了一块生成excel的代码，因为相似的代码很多，就用到了简单工厂模式和策略模式，工厂模式让对象的创建更加明确，策略模式让代码的逻辑看起来更加清晰、也解决一些代码重复的问题。

2019-11-04 23:58

SweetyTang

用的最多的是观察者模式，其它模式都不了解，有些太失败了

2019-11-04 18:05

摸爬滚打三十年

关于为什么早学设计模式我很有发言权，我们公司的食堂有一道名菜:肥牛金针菇，去的早的吃肥牛，去的晚的吃金针菇。

2019-11-14 19:41

时光之刃

希望老师能举一些开源代码中的设计模式，比如netty或者ES等

2019-11-05 23:10

作者回复

可以的 我后面可以考虑多写几篇加餐

2019-11-06 06:41

于留月

设计模式的重要性：
1、提高设计和开发能力，从更高层次考虑问题：可读性、可维护性、可扩展性、模块化、组件化；
2、读源码、学框架，深有体会，Android源码到处充斥着各式各样的设计模式和设计原则；
3、职场发展，个人希望能够在架构和技术leader方面深入发展；
4、高质量代码，避免坏代码，不想被同行嘲笑，不想被同事鄙视，不想被后期维护的人吐槽，想被人称赞；
5、作为专业话术，更方便同行业间交流，更容易跟领导解释清楚，更轻松带领团队完成任务；

使用经验：

策略模式：
解决开发、测试、预发布、生产环境不同的数据来源、不同的数据处理方式，以及不同的图片加载方式

建造者模式：
网络通信协议，非常规意义上的http请求，更多是Socket通信，需要处理大量的参数传递，包装，解析

2019-11-05 23:09

彦祖
之前读完了大话设计模式，入职新公司后，为了完成新的需求复用旧类，采用了模板方法模式。为了把那些好几百行的代码拆开重新梳理，也只是能用到建造者模式来解决方法参数贼多的问题。
2019-11-04 17:51

啊哈哈哈怕
是不是适用设计模式也应该一分为二的看，如果业务非常简单就没必要上设计模式增加系统的复杂性了。
2019-11-07 16:07

帆大肚子
我是做iOS开发的，之前一直有了解过设计模式，也在自己的一个框架中应用了组合模式。

重要性那就更不用说了，上个坑是一个专门定制ERP的公司，需要从APP上录入很多信息。我接手的时候，准备一根绳子随时准备上吊。我就连类似于把姓名输入项和性别输入项换一个位置，都要换一个上午。
学习了设计模式之后，我对应公司的开发场景，应用组合等模式把公司的iOS开发组件化（最快的时候一天写了20个表单页面）。
虽然自己有学过，但是如王争老师所说的，自学的效果真的有些不尽人意，并缺少反馈。
熟悉OC的同学可以去看看我的作品，希望大家给予指点
https://github.com/xiefans/FFormLib
但是因为自己比较懒，没有给文档补全。
2019-11-04 18:55

落叶飞逝的恋
学完这个，再去重温数据结构，大有裨益。
2019-11-04 17:38