52讲门面模式：如何设计合理的接口粒度以兼顾接口的易用性和通用性

前面我们已经学习了代理模式、桥接模式、装饰器模式、适配器模式，这4种结构型设计模式。今天，我们再来学习一种新的结构型模式：门面模式。门面模式原理和实现都特别简单，应用场景也比较明确，主要在接口设计方面使用。

如果你平时的工作涉及接口开发，不知道你有没有遇到关于接口粒度的问题呢？

为了保证接口的可复用性（或者叫通用性），我们需要将接口尽量设计得细粒度一点，职责单一一点。但是，如果接口的粒度过小，在接口的使用者开发一个业务功能时，就会导致需要调用n多细粒度的接口才能完成。调用者肯定会抱怨接口不好用。

相反，如果接口粒度设计得太大，一个接口返回n多数据，要做n多事情，就会导致接口不够通用、可复用性不好。接口不可复用，那针对不同的调用者的业务需求，我们就需要开发不同的接口来满足，这就会导致系统的接口无限膨胀。

那如何来解决接口的可复用性（通用性）和易用性之间的矛盾呢？通过今天对于门面模式的学习，我想你心中会有答案。话不多说，让我们正式开始今天的学习吧！

门面模式的原理与实现

门面模式，也叫外观模式，英文全称是Facade Design Pattern。在GoF的《设计模式》一书中，门面模式是这样定义的：

> Provide a unified interface to a set of interfaces in a subsystem. Facade Pattern defines a higher-level interface that makes the subsystem easier to use.

翻译成中文就是：门面模式为子系统提供一组统一的接口，定义一组高层接口让子系统更易用。

这个定义很简洁，我再进一步解释一下。

假设有一个系统A，提供了a、b、c、d四个接口。系统B完成某个业务功能，需要调用A系统的a、b、d接口。利用门面模式，我们提供一个包裹a、b、d接口调用的门面接口x，给系统B直接使用。

不知道你会不会有这样的疑问，让系统B直接调用a、b、d感觉没有太大问题呀，为什么还要提供一个包裹a、b、d的接口x

呢？关于这个问题，我通过一个具体的例子来解释一下。

假设我们刚刚提到的系统A是一个后端服务器，系统B是App客户端。App客户端通过后端服务器提供的接口来获取数据。我们知道，App和服务器之间是通过移动网络通信的，网络通信耗时比较多，为了提高App的响应速度，我们要尽量减少App与服务器之间的网络通信次数。

假设，完成某个业务功能（比如显示某个页面信息）需要“依次”调用a、b、d三个接口，因自身业务的特点，不支持并发调用这三个接口。

如果我们现在发现App客户端的响应速度比较慢，排查之后发现，是因为过多的接口调用过多的网络通信。针对这种情况，我们就可以利用门面模式，让后端服务器提供一个包裹a、b、d三个接口调用的接口x。App客户端调用一次接口x，来获取到所有想要的数据，将网络通信的次数从3次减少到1次，也就提高了App的响应速度。

这里举的例子只是应用门面模式的其中一个意图，也就是解决性能问题。实际上，不同的应用场景下，使用门面模式的意图也不同。接下来，我们就来看一下门面模式的各种应用场景。

门面模式的应用场景举例

在GoF给出的定义中提到，“门面模式让子系统更加易用”，实际上，它除了解决易用性问题之外，还能解决其他很多方面的问题。关于这一点，我总结罗列了3个常用的应用场景，你可以参考一下，举一反三地借鉴到自己的项目中。

除此之外，我还要强调一下，门面模式定义中的“子系统（subsystem）”也可以有多种理解方式。它既可以是一个完整的系统，也可以是更细粒度的类或者模块。关于这一点，在下面的讲解中也会有体现。

1.解决易用性问题

门面模式可以用来封装系统的底层实现，隐藏系统的复杂性，提供一组更加简单易用、更高层的接口。比如，Linux系统调用函数就可以看作一种“门面”。它是Linux操作系统暴露给开发者的一组“特殊”的编程接口，它封装了底层更基础的Linux内核调用。再比如，Linux的Shell命令，实际上也可以看作一种门面模式的应用。它继续封装系统调用，提供更加友好、简单的命令，让我们可以直接通过执行命令来跟操作系统交互。

我们前面也多次讲过，设计原则、思想、模式很多都是相通的，是同一个道理不同角度的表述。实际上，从隐藏实现复杂性，提供更易用接口这个意图来看，门面模式有点类似之前讲到的迪米特法则（最少知识原则）和接口隔离原则：两个有交互的系统，只暴露有限的必要的接口。除此之外，门面模式还有点类似之前提到封装、抽象的设计思想，提供更抽象的接口，封装底层实现细节。

2.解决性能问题

关于利用门面模式解决性能问题这一点，刚刚我们已经讲过了。我们通过将多个接口调用替换为一个门面接口调用，减少网络通信成本，提高App客户端的响应速度。所以，关于这点，我就不再举例说明了。我们来讨论一下这样一个问题：从代码实现的角度来看，该如何组织门面接口和非门面接口？

如果门面接口不多，我们完全可以将它跟非门面接口放到一块，也不需要特殊标记，当作普通接口来用即可。如果门面接口很多，我们可以在已有的接口之上，再重新抽象出一层，专门放置门面接口，从类、包的命名上跟原来的接口层做区分。如果门面接口特别多，并且很多都是跨多个子系统的，我们可以将门面接口放到一个新的子系统中。

3.解决分布式事务问题

关于利用门面模式来解决分布式事务问题，我们通过一个例子来解释一下。

在一个金融系统中，有两个业务领域模型，用户和钱包。这两个业务领域模型都对外暴露了一系列接口，比如用户的增删改查接口、钱包的增删改查接口。假设有这样一个业务场景：在用户注册的时候，我们不仅会创建用户（在数据库User表中），

还会给用户创建一个钱包（在数据库的Wallet表中）。

对于这样一个简单的业务需求，我们可以通过依次调用用户的创建接口和钱包的创建接口来完成。但是，用户注册需要支持事务，也就是说，创建用户和钱包的两个操作，要么都成功，要么都失败，不能一个成功、一个失败。

要支持两个接口调用在一个事务中执行，是比较难实现的，这涉及分布式事务问题。虽然我们可以通过引入分布式事务框架或者事后补偿的机制来解决，但代码实现都比较复杂。而最简单的解决方案是，利用数据库事务或者Spring框架提供的事务（如果是Java语言的话），在一个事务中，执行创建用户和创建钱包这两个SQL操作。这就要求两个SQL操作要在一个接口中完成，所以，我们可以借鉴门面模式的思想，再设计一个包裹这两个操作的新接口，让新接口在一个事务中执行两个SQL操作。

重点回顾

好了，今天的内容到此就讲完了。我们来一块总结回顾一下，你需要重点掌握的内容。

我们知道，类、模块、系统之间的“通信”，一般都是通过接口调用来完成的。接口设计的好坏，直接影响到类、模块、系统是否好用。所以，我们要多花点心思在接口设计上。我经常说，**完成接口设计，就相当于完成了一半的开发任务。只要接口设计得好，那代码就差不到哪里去。**

接口粒度设计得太大，太小都不好。太大会导致接口不可复用，太小会导致接口不易用。在实际的开发中，接口的可复用性和易用性需要“微妙”的权衡。针对这个问题，我的一个基本的处理原则是，**尽量保持接口的可复用性，但针对特殊情况，允许提供冗余的门面接口，来提供更易用的接口**。

门面模式除了解决接口易用性问题之外，我们今天还讲到了其他2个应用场景，用它来解决性能问题和分布式事务问题。

课堂讨论

1. 适配器模式和门面模式的共同点是，将不好用的接口适配成好用的接口。你可以试着总结一下它们的区别吗？
2. 在你过往的项目开发中，有没有遇到过不合理的接口需求？又或者，有没有遇到过非常难用的接口？可以留言“吐槽”一下。

欢迎留言和我分享，如果有收获，也欢迎你把这篇文章分享给你的朋友。

精选留言

小兵

适配器是做接口转换，解决的是原接口和目标接口不匹配的问题。
门面模式做接口整合，解决的是多接口调用带来的问题。

2020-03-02 07:58

Frank

以前在做Activiti工作流开发时知道该工作流引擎提供了诸多门面接口供外部使用，以前只知道这样设计是对很多细节做了包装，提供友好易用的接口供用户使用。今天学习了本章内容，加深了对门面模式的理解。门面模式从定义上来看是为接口设计而提出的，所以在开发中我们在设计接口时可参考该模式。该模式对应到了之前学习过的一些设计原则和思想，如封装，迪米特法则。
对于课堂讨论：
1. 适配器模式与门面模式的区别：（a）适配器主要是为了解决接口不兼容的问题，而门面模式主要用于设计接口的易用性问题。（b）适配器在代码结构上主要是继承加组合，门面模式在代码结构上主要是封装。（c）适配器可以看作是事后行为，是一种“补偿模式”，主要是用来完善设计上的不足，而门面模式是在设计接口时就需要考虑的，是一种事前行为。
2. 在过往的开发中，自己在写接口时除了满足需求外大部分考虑是接口的幂等性，限流，安全等。对于接口的可复用性考量的不是很好，还需要大量的实践来加深。

2020-03-02 08:22

黄林晴

吐槽不存在的，我只知道我现在做的app刚启动的时候 要调用五六个接口...，之前没了解过门面模式，不过我在想，我去说服

务端改成门面模式之前，要确定一个问题，那就是门面模式是将很多接口整合在一起，那么势必，牵扯到传参变多，以及返回数据量多的因素，这种情况下应该也比较影响效率，比如一个接口是从student表中查询，一个是从course表还有一个是从teacher表中查，门面模式和直接写一个接口sql查询这么多的效率是一样的吗

2020-03-02 20:05

progyoung

解决分布式事务问题的应用场景中，如果用户和钱包并没有公用同一个数据库，那么是不是门面模式也不适用了呢?

2020-03-02 09:00

下雨天

门面为了"偷懒"用起来更方便；适配器是不得已，老接口已经不可用或者不好用了。

2020-03-02 09:55

小刀

适配器--继承+组合

门面---封装

2020-03-04 08:22

小晏子

适配器模式和门面模式要的解决问题就不一样，适配器模式为了适配两个不兼容的系统，关联两个不兼容的接口，当程序必须遵循特定的接口并且必须支持多态行为时使用适配器；而门面模式要提供一个更简单易用的接口，比如你有一个开关可以控制打开你家的电视，空调，灯，等，这就是门面模式：一个按钮或功能需要一系列更复杂的步骤。

在实现上也有区别：门面模式定义了新的接口，而适配器模式使用旧的接口，适配器模式使两个现有接口同时工作而不是定义一个全新的接口。

2020-03-02 10:06

test

适配器模式将原来不统一的每个接口统一，门面模式将一组接口统一暴露为同样一个接口

2020-03-02 08:53

杨杰

其实吧，我想吐槽的是各种查询接口。就拿用户来说吧，可以按照手机号码、可以按照邮箱或者其他方式查询，有可能要查询已经停用的或未停用的。在复杂的查询场景下，很多时候不得不单独写查询接口。有一种方案是写一个相对通用的queryparam，通过传入不同的查询方式来实现不同的查询方式。。不知道这样是不是好的实践。

2020-03-03 20:31

Dimple

适配器模式将接口间貌似不能对接东西，通过适配完美地对接在一起，实现从上到下连贯；

门面模式，将多个独立的、又可以先后完成的事项统一打包起来，以此达到连贯的目的；

2020-03-03 15:54

每天晒白牙

适配器模式主要解决的是解决接口不兼容问题

门面模式主要解决接口易用性问题，同时还能解决多个接口调用的性能问题还有分布式事务问题

对门面模式有个很深的印象是之前做第三方支付时，需要调用渠道的接口，然后在前面封装了一个接口，供交易系统调用，接口名字就带facade，就是用到了门面模式，把子系统给屏蔽起来，对使用方很友好

2020-03-03 15:31

墨雨

建议老师可以给出一下典型的实现代码，这样会更直观一些

2020-03-03 09:00

平风造雨

适配器模式更多的解决原接口的定义不适合在当前业务中满足统一抽象的需求而产生的，门面模式更多的是为了解决调用方简单易用，把细粒度的多个内部接口重新编排成粗粒度的接口。

2020-03-02 23:29

zengjian

每天除了打卡还有一个乐趣就是点大家的头像，哈哈哈哈

2020-03-02 22:54

Fstar

适配器模式和门面模式的区别：适配器模式用于事后的补救，是对设计失败的接口进行的补救的方法；门面模式是将细粒度的接口进行组合，封装细节，提高调用者调用接口的体验。

2020-03-02 22:38

荀麒睿

问题1：门面模式主要是进行封装，给上层提供必要易用的接口，而适配器模式主要是为了弥补兼容性问题，个人认为门面模式主要着重与设计阶段，而适配器模式主要着重于事后的一个补偿重构阶段，一个是为了上层业务进行适配，一个是为了弥补缺陷而进行适配。

问题2：仔细想想之前自己写的接口有些与项目过于冗余，只考虑了完成任务，现在回想一下有些接口都是做了一些重复劳动，需要反思

2020-03-02 21:39

Ken张云忠

适配器模式和门面模式的区别:

适配器模式为了使不兼容的接口变得兼容,使不能用的接口变得可用.

门面模式主要为了提升接口的易用性,使能用的接口变得更加好用.

2020-03-02 15:55

往事随风，顺其自然

门面模式怎么实现，代码结构如何

2020-03-02 14:23

Jxin

1.门面模式应该算是适配器模式的一个子集。

2.我们的项目dao层即是rpc接口，里面即包含db操作的方法，也包括业务方法，同时全部是public，即使它只在本项目使用。使用放的感受就是打个.三四十个方法，都不知道用啥，找个功能费劲。然后insert方法内嵌套业务代码，导致一开始用被坑，后面还得自己再拆个干净的insert。使用方感受就是，每个方法都得看源码，接口是骗人的，用他接口不如自己写舒坦。

2020-03-02 13:10

sabo

1.适配器模式与门面模式的共同点都需要二次封装，隐藏内部细节。不同点为适配器是为了统一格式，门面是为了简单易用。

2.我自己就开发过恶心的接口，最近做的这个项目就是考虑了很多复用性的问题，api设计的比较松散，一个注册登录的过程需要前端调用三次接口。接口是通用了，但是感觉前端人员非常不高兴。。。使用起来不太方便，且在前端需要的业务逻辑也更多了。这种情况还是很有必要使用门面进行包装一下。

2020-03-02 11:08