# 小猛 VS P8面试官

小猛

00:03

你好，面试官。

胤禛

00:05

你好。非常高兴你参加这次面试，你是，你是小猛是吗？

小猛

00:14

对对对我是小猛。

胤禛

00:16

OK，你能简短地介绍一下自己的经历吗？

小猛

00:22

你好，面试官我叫小猛，然后的话，我这最近几年的话，主要都是在那个电商和支付行业这一块工作，一直主要做的也是支付这块的一个业务。然后的话，我以前也是从那个支付的那个，就是那个类似于单体模块的这种项目，逐渐演进成那种中台化的一个。这个模式的。对于它里面的一些痛点，以及整个项目迭代的一个一些细节，都会有一些自己的一些就是实战的一些经验在里面。

小猛

01:00

的话，我们这边，主要是是电商，就是电商这一块的。相对来说的话，会接触的是业务线的话，主要接触到，比如说是电商的一些APP和线下门店两块两大业务这一块。

胤禛

01:19

明白，你介绍完了吗？

小猛

01:23

对。

胤禛

01:25

OK，我看了一下你简历，你能告诉我，因为我看你最佳最新的就是最近的一家，就是来一份的这个科技有限公司，然后从2018年到现在大概也有三年多了，对，然后你为什么，为什么会离职。

小猛

01:35

对对对。

小猛

01:40

对的。

小猛

01:44

是这样子的，在这个公司的话，我要经历过的一个过程，是从他的一个项目，从一个单体，然后就是比较偏单，偏读于那个支付能力，这个项目给他制度也升级成了一个支付的一个中台，然后到目前为止的话，项目已经趋于一个比较稳定的一个一个情况，他现在比较。就是贴近于维护的一个状态了。对于我目前来说的话，我希望自己可以去外面寻找一下更大的一个挑战，可以对我自己的一个业务认知，希望能得到一个更大的一个提升。

胤禛

02:21

你今年年龄30岁是，然后其实从刚才你的描述来看，其实是你是想还是说想在技术上再多做一些点，对，提升一下。

小猛

02:23

对对对。

小猛

02:33

对我这我我这个人的话比较偏向于一些就是业务方案的一个深深入的更深入的一个研究就是，希望把这个业务可以做到就是非常通用的然后，性能和就是迭代制度，比较比较方便这一块的

胤禛

02:53

好，OKA，那你能介绍一下，就是你的项目，你做了也有好多年，从一四年到这个，目前到就是二二年，大概也有恶恶，就是六七年的工作经验，那你这个项目里面你能说你能去说一下哈，你项目里面最难的哈，你最难遇到的项目，他的一个一个是他的背景，一个还有他的这个价值，还有它的这个解决方案吗。

小猛

03:26

我的话目前是是这样子的，因为之前我们一开始的一个项目是比较就是偏就是简单型的，它所能够承载的业务会比较少。然后。公司的话是说要对这个公司的整个的一个项目，全部项目进行一个大的升级重构，那这个重构的一个过程中的话，是有一定的一些技术挑战的。比如说我们这边的话，以前会会会会说挣下那些支付里面，我们之前是支持那种单一支付的，那我们现在要支持那种很多种组合支付，比如说我们有那种点卡加加，这些第三方支付，这种可以组合成一些组合支付，我们从架构设计上面要去满足他。这是一个点。

胤禛

04:14

怎么满足的，怎么满足的？

小猛

04:16

怎么昨天我们这边的话，第一个是首先订单这边的话，它一般默认都是会有那种父子单的一个概念的，那对于我们支付单的话也是一样，我们这边也会有一个对应的支付单的概念，它也有一个复制单，在创建订单的时候，默认会将你的这个负担的信息带到支付这边来，同时在这边生成一个预支付单，就是支付单的一个一个。

小猛

04:41

一个数据落地。那对于订单来说的话，他可以选择负担，就是整体全部，也可以选择只单进行一些支付。这里面涉及到的话，就是对应的我们这边支付单里面的一些负担，盒子单里面的一些流水信息，你如果只是支付了其中一部分，我可以选择对里面某一个子弹进行一些支付金额的一些变更和和，就是完成这个流程，如果说你是选择整单的话，他就可以对所有的支付子丹进行一个完整的一个。

小猛

05:17

支付流程的一个迭代吗。

胤禛

05:20

明白整个流程听起来还挺顺畅，没有太大的问题，但是，我毕竟是处在一个高并发的一个环境，对吗？就是互联网的高频环境，我确认一下，你的整个的solution是建立在微服务上吗？还是说是个单体的。

小猛

05:38

微微服微服务，我们最近的话，升级是说使用的那个scarecrow的这套微服务的一个技术体系。

胤禛

05:47

我确认一下哈，就是因为我刚才听起来你其实，这个交这个支付的流程，也是蛮复杂的，并且用了这个微服务，那你是怎么保证整个过程的数据哈是保证一致的，一致性的，因为是因为是跨进程，跨微服务的这种调用，那你怎么保证数据的一致性。

小猛

06:05

对。

小猛

06:07

我们这边的话，数据主要是和订单这边做一些交互的，也就是说比如说正向的正向的话，刚刚说了在创建订单的时候会有一个预支付的，一个就是创建支付单的一个动作，当你创建支付单之后，这个时候会有收银台，收银台进行一个选择，其中的一个支付方式进行支付。一旦他收到了一个支付回调通知之后，等于说我这边的一个支付可能结果就完成了，但是此时他会有一个有有刚刚说的这种场景，他可能只是部分支付。当我所有的支付一旦满足了我支付单的金额，被完整的支付完成之后，我这里面会涉及一个支付通知的一个行为，支付充值的行为的话，我当时是考虑了一个系统的，一个就是可扩展性，就是它可以支持多种方式去通知，有MQ的，或者是HTTP的，还有那种做反射去做那种就是double直接，直接内部通信大，服务直接那种教育者。

胤禛

07:08

OK，你你暂停一下，你刚才你提到了两个比较关键的两个点哈，一个是那个W的那个远程调用对？一个，是那个MQ的这种节呕恶异步化的一种方式。第一种，我先问第一个问题，就是在double这种微服务的这种环境下，就是远程调用，因为在网络里面会发生抖动，可能你这一次调用可能会失败。比如说你在double比如说a服务，比如说交付，比如说支付这个动作的这个微服，你已经落盘了。说这个这个客户，这个用户已经发起了一个支付，但是在另外一个微夫，比如说是那个预生成或者是预处理的一个微服务，他没有没有收到，因为网络抖动了或者网络断掉了，或者有各种原因，没有收到，没有收到的，毕竟的第一笔的这个动作的数据，已经生成一个记录但是，另外一个服务没有收到，所以说，整个儿两边的那个业务本身就。

小猛

07:20

对对。

胤禛

08:08

不一致，这个怎么去解决，第二个，你说的那个MQ的那个就是截藕，它是因为MQ本身，它实际上是一个消息订阅，消息订阅的一个这样的一个中间件，那你怎么去解决？就是两端，就是一个

procedure一个consumer之间的一致性，因为你的因为你有两个服务，肯定有一个服务是用了procedure另外一个服用consumer对？然后，他怎么保证两端的数据的一致性，因为这些都是你你你，你做这个支付场景里面做，如果是用微服的话。

小猛

08:36

对。

胤禛

08:46

遇到的最长的，就是最常见的这些问题。

小猛

08:51

我首先说一下，就是第一种的那种服务，就是服务直接直接调用的这种的一般来说的话，我们服务这边调用我们这里会设置一个，就是它的一个通知的一个状态。那我在那个服务调用的时候，我不是说是异步调用的，我是同步去掉用的，相当于说我必须要收他的一个应答，结果之后，我才会将他的状态更新为通知，一个成功的一个一个一个标识。那如果说他这个时候通知发生了失败，那我们这边会有一个定时的一个任务，或者去扫描这些，需要支付已完成，但是通知状态为已就位通知的这些数据，我会进行再一次的一个。调用直到他成功为止，保证它的最终一定要是成功。

胤禛

09:36

但是这里面好像会有一个插头，因为你客户端，虽然认定这次调用失败了，但是只是因为网络抖动，只是服务器并没有给你回执，但是服务器确实把这个数据，就是比如说服务端是另外一个服务，另外服务，确实也收到了这个信息，然后把它落盘了，对，只是说网络回来的时候网络抖动，你没收到而已。

小猛

10:01

这个这个的话没有关系，因为首先第一点，我们服务之间的一个通信，它有一个唯一，唯一的一个值的，比如说我通知过去的话，我肯定是通知的他这一个整个一个号，那对于我来说的话，他通知他收到了我一个请求，他这边已经处理完了，他第二次再给我重复发一个，就是这个请求通知过去对他来说，他只是说。

胤禛

10:02

对?

小猛

10:25

会检测到说这笔订单已经是一个支付完成状态了。

胤禛

10:28

你在业务上其实你在，你在另一端实现的幂等对。

小猛

10:32

对对对对对这一块，他在在在服务交。

胤禛

10:34

行OK，那再再问下一个问题，就刚才那个MQRMQ怎么去保证两端的这个这个消这个数据一致性。

小猛

10:45

MQ。

小猛

10:49

MQ的话，我这边理解他。和那个。你是类似的，我只要保证我的消息。是一定是发到MQ里面发到就是说发成功了，我就我就我就默认，我这边其实已经是完成了，我就无需再去关注他了，他只需要订单那边他自己去消费即可。我保证的只是说我的消息一定要发到MQ里面去就就即可了我这边可以。

胤禛

11:17

你了解过事务性，你去了解过事务消息吗？

小猛

11:21

事务消息我了解过，但是我们，没有去使用这一块。

胤禛

11:27

那你刚才说的这个方案能保证两端一定数据是一致？

小猛

11:32

首先我这边要保证的是，我的消息一定会投放进去。这边的话我会，我不是说就是说我一定要收到她的消息，确认告诉我已经写入到那个MQ的队列里面去了，这个时候我才去更新我的状态，为那个一通知，那我这边相当于说我能保证的是第一点是他的消息是，已经被投放的。

小猛

11:53

对于对于订单来言的话，他这边。

胤禛

11:56

也就是说，你得也就是说，你得选择一个能够去给你回执的，能确认你这个消息已经落盘的，这样的一个消息中间。

小猛

12:05

对对对对对，我们用的是MQ，我们是是说一定要他这边就是消息给我返回，就是说已经投递成功了。对对对。

胤禛

12:12

会会给你一个AK。

胤禛

12:15

会给你个CK对。

小猛

12:16

是的是的，我这边的话，因为这边的话就是没有去追求她的那个，就是消息投递的一个性能的问题了。

胤禛

12:23

那那个consumer那一段。

小猛

12:25

反正那一端的话，他这边的话也是同理的。消费端的话，他他，他也要保证他的消息一定就是说会去做这个，就是说处理，首先它这里也可以通过几种机制，第一个是我们这里会通过消息投递的一个形式去做处理，第二个是他还可以同一个一些补偿自己的一些补偿机制，比如说收到了我们的一些处理的一些异常结果的时候，他是不是他自己其实也可以主动来查一下我们这里的一些不会。

胤禛

12:54

这里有个关键问题，就是consumer怎么知道procedure发了消息。

胤禛

12:59

因为它算是一个分离的分离的系统，对，分布式系统。

小猛

12:59

怎么知道?

胤禛

13:03

那consumer怎么知道那个另外的服务已经有有所动作了，发送了一个消息，这边需要去订阅，需要去查询。

胤禛

13:14

怎么知道?

小猛

13:18

这边的话是。

胤禛

13:18

就是保证说白，说白了就是背后的逻辑，怎么保证这个消息不丢，就是消息不丢的话，就consumer一定会收到，就确保这与一抹语义的，就是至少一次可达的这个语义。

小猛

13:32

这个的话。这个我我这边设计上面的话，可能功能它就一定能可达。

胤禛

13:40

这个不能做这样的假设，网络网络上也会出现各种情况，OK。

小猛

13:47

对，可能还没是没有遇到这种场景。

胤禛

13:51

现在已经过了将近20多分钟，然后我基本了解你的情况了，刚才我听起来，其实你这个方法还是蛮复杂的，你你有没有更简单的就是优化过的，就是你思考过，也许你现在项目这么做，但是你想没想过他会有更好的一些解决方案。

小猛

14:11

就是你是说的那个刚刚那个支付的整个完整的链路通知这一块。

胤禛

14:15

对。

胤禛

14:17

对对对。

小猛

14:20

我我因为我这边，其实我对支付的一个业务参考斯深度思考，我是是这样子想的，他这边的思考是分为两块，一块是所谓的业务上面的一个沉淀，所谓业务上的沉淀的话，它可以是这么理解，就是我们可以刚刚可以想到，他里面其实会有很多的一些，比如说异常那些都得犯一些，支付能力的一些提供这一块的话，是一个业务上的一个方案，一细节。

小猛

14:47

其实第二个是一个底层的那个通用模块，我们因为我们其实做支付，我做那种第三方支付的时候，他们不是那种纯正的说是，来一个渠道接一个渠道，这样子它的接入方式多了之后，成本会非常高，我们会考虑将这一块沉淀出来，就是做成通用型的，也就是个配置化，将他的一些通用的功能，比如说密钥阿阿通讯协议，然后。

胤禛

14:54

明白。

小猛

15:15

他的一些什么参数，组装，解析的一些，抱红音，打码等等这一块儿全都做成配置化，也就未来的话，我增加一种新的，只要它的基础功能支持的场景下，我觉得说把他们从这些基础功能里面给他关联起来，我就可以完成一个新的第三方的一个业务功能，支持这一块，其实跟外部的那个业务功能。

胤禛

15:24

明白。

胤禛

15:34

特工。

小猛

15:39

提供是就是两块是独立开来的。

胤禛

15:43

就相当于业务流程在上层做业务流程编排，对，那你那你上层业务流程编排这一块儿，如果交给程序员去编排的话，一个可能会比较复杂，因为里面两种情况有没有你们上面有没有建类似于这种业务流程编排引擎之类的东西。

小猛

15:47

对。

小猛

16:06

目前是没有，但是我想去。

胤禛

16:09

那就还还还是还是靠程序员自己去手工去写。

胤禛

16:13

去组装。

小猛

16:16

对，目前我们试试你最上层的这一块是暂时还是没有的事，是自己去业务，就是通过自己程序员去去去去组装的，但是最下层的那块。调用第三方这一块的话，就是通过配置方配置的一个方式，就是他自己映射到具体的一个底层的原始方法去去阻断调用的。

胤禛

16:39

那个配置是由那个业务方的开发人员自己去写。

小猛

16:43

对有业务方他自己去去去去学。

胤禛

16:45

你你，你这边你这边其实只是提供一个平台对吗。

小猛

16:51

我这边提供一个平台，但是。

胤禛

16:54

明白。

小猛

16:54

这一块的话，其实因为我们现在所在的公司，并没有分到就是说那么大，就是有些东西可能就是全都全都需要自己一个人来做的。

胤禛

17:05

好的，那我大概明白了，OK，我再问一下你，你现在这个整个这套系统现在用户量是多少。

小猛

17:15

我们的用户量的话，主要是分为两块，一块是线上的APP的一个用户，这个的话用户量的话大概有几百万，但是活跃用户的话，可能是几十十万多十多万的样子，其次的话是。

胤禛

17:29

那你的那你的那个QPS是多少，PPS是多少。

小猛

17:34

这个是看高峰，还是说看平时。

胤禛

17:38

你就看高峰是多少。

小猛

17:41

高分的话，TPS，我们这边大概是2000的样子。

胤禛

17:48

你有多少台机器?

小猛

17:53

我主要说下我们这边的机器，其他的，不是太关注，然后我们这边的话大概是六台的样子。

胤禛

18:01

OK。

胤禛

18:03

好的，我技术这块儿我问完了之后，我这块有一些几个小的开放式问题，一个是说你觉得。对你来说哈，就是能影响你，比如说你在一个公司里面，比如说那个假设哈，比如说跟你的上级冲突了，你怎么去解决这个问题。

小猛

18:27

上级冲突了。首先跟上级冲突肯定是有一定的原因的那。首先要去思考这个原因到底是工作带来的，还是说只是纯粹的个人的一个情绪上面带来的，工作上面带，假如说是工作上面带来的话那。必然就需要思考，就是说你到底这个事情是属于谁，对了谁错了，然后你跟他冲突，他引来的一个成本会怎么样，因为因为要考虑一个很现实的问题，就是你是属于下级，他是属于上级有一些问题的话，不可能是两个人一直这样子去去去去僵硬的，肯定要采取一个比较。就是折中的一个方式，去某一个方面去妥协，去去去解决这些问题。就是就假如说就是上司并不认可你这个方案，然后你一定说要用这个方案。然后两个人吵起来了那。反过来先，上次为什么不认可你的方案，肯定是有原因的，那我。选出自己让他们满意的方案，我就可以解决他这个事情，这这是假如说工作上的一些事情，其次是情绪上的，情绪上的话，一般来说就是，可能是因为公司的一些环境，或者什么一些问题，就是压抑，对你的一个心理压力会比较大，这一块的话，我个人觉得就是。主要看自己的一个。快速调节能力就说白了，就是你平时下班以后对自己的一个情绪。

小猛

19:58

要进行一个放松，而不是说将自己在上班时候带着一些个人情绪带到自己下班，然后第二天又重复这样子，造成一个就是死循环这样子，而是他自己在平时不工作的时候，尽量去把自己的一个。

胤禛

20:12

我明白了，就是咱注意一下时间，我们只只有45分钟OK，我第二个问题是说你你的优缺点是什么。

小猛

20:16

好的。

小猛

20:25

我的优点的话，我个人觉得我的优点就是说，第一个我做一些功能设计，或者一些东西方方式去去去考虑的话，我更加偏向于他的一个原子通用能力这一块，就是我不是像对头像对对对，因为我个人现在目前在工作中遇到的很多就是同事开发，他们都会有一个很本质的问题是他们。

胤禛

20:40

抽象抽象能力对吗。

小猛

20:51

做一些东西，他们只是为了完成这个功能，他们不去考虑这个东西怎么去抽象化，去怎么去做那个底层复用，还有蛮来进行第二次开发的时候，可以减少我们的一个工作量，这是我目前觉得我可能这块优点还是有的。

胤禛

21:07

OK，那我问一下你，你刚才提到一个抽象，你知道抽象的最关键点是什么？

小猛

21:13

首先第一个地方的话就是它。有一些细节点不能跟当前的一些业务业务，业务就是太过于强大的关联，而是我我我也觉得他是一个无状态的一些一些基础功能就是打个比方，我这里面可能最底层那些基础原则方法，都可能是跟当前的业务可能是完全毫无关系的。

小猛

21:37

好，但是但是经过当前的业务进入它一个组装之后，他可以就是说组装出一个具有独立的一个业务形态的一个一个方法。

胤禛

21:37

OK。

胤禛

21:48

OK，那你觉得你的缺点是啥？

小猛

21:51

缺点的话，我个人觉得。相对来说，自己可能所所属的一个业务线接触的不是不是特别多，可能做的就是支付或者是交易这块，对于全部的一个业务线掌控的不够全。

胤禛

22:11

你指的不够权是指哪方面？除了技术上，或者说是业务上，还有还有其他方面业务上。

小猛

22:16

以业务上就是就是业务上面，就是，比如说你对于这整个一个公司的一个业务线，你可能所所能掌握的知识度，它可能就局限于你。现在目前工作所接触到的一个业务线，而其他没有接触到这个业务线的话，相对来说，因为接触的少，所以他对她于它的一个深度理解不是很足。

胤禛

22:39

你觉得你应该怎么去弥补这个这个缺点。

小猛

22:45

第一个是勇于去主动去。尝试去做一些不属于自己就是擅长领域的一些。新的任务去挑战自己，就是说比如说我说我对商品不了解，但是我愿意主动去尝试，去做这一块，只要有这个机会，我就愿意主动去尝试，去做这个东西，只要自己愿意去做的东西越多，慢慢的自己所能掌控的一些业务线也会，资质的深度也会越多。其次的话可以通过一，你说。

胤禛

23:13

OK，那你，你继续说，你可以继续说。

小猛

23:16

其次的话是第二个，就是可以通过一些，就是朋友之间的一些分享，以及一些，有合适的一些教材学习，通过这个东西去补足自己的一些业务知识知识面。

胤禛

23:32

OK，你能简单说一下你是怎么学习的？

小猛

23:37

首先的话，我是我这个人学习他比较偏向于那个方案设计这一块儿，因为我认为所有的方案设计，它不是说一定趋向于当前的。你所看到的这个业务系统，它并不一定是的，他只要是一个优秀的设计，它就可以用到你的一个就是项目，时间里面来。

小猛

24:00

这是第一个点，我就是我第一我喜欢就是把他们的一些设计应用到我们的项目里面来，这是第一个，第二个是我。

胤禛

24:00

OK。

小猛

24:09

比较喜欢去研究一下人家那些源码的编码风格，因为我现在就是感觉就是代码风格之间有些代码分割好的非常易懂，可为护墙维修非常强，有些代码，他就是为了功能堆直接堆出来的代码，都不知道他到底写的什么东西那种。更加偏向，于是说去学习人家一个好的代码风格，主要学习的话是以这两个方面为主。

胤禛

24:38

OK。可以我觉得，我再问一个问题，就是，你觉得这个七七，因为里面的是PC，你做一个PC应该具备哪些能力。

小猛

24:54

首先第一个点，他肯定要对某一块业务线就非常的熟悉，它能够可以去对某一个业务线的，这个整体的功能模块就能够从头到底，这是第一个点。

小猛

25:07

第二个点的话。就我刚刚说的，业务限制是你的专长，但是其他的一些归你所上传的业务线也要有一定的知识面，否则你跟人家去沟通，你你作为一个主，你跟人家去推动这些业务，你不了解人家业务，你没办法跟人家去沟通这一块东西，所以你要对其他的业务线也有一定的一个掌控力，这是第二个点。

小猛

25:29

第三个点，就刚刚说的你项目里面可能会面临一些。比如说高并发，各种异常场景，那你也要有一些实际的一些解决方案，通过做成方案能够弥补，公司在这个上面尽量不会出什么错误，我觉得起码这几个点是要达到的。

胤禛

25:49

OK，行，我基本了解你的情况了，你有什么问题问我吗。

小猛

25:57

我想请问一下就是。

小猛

25:59

这这一块刚刚说的，我刚刚就是聊的时候聊到这个业务里面，说是业务设计的会相对比较复杂，我想了解一下就是如何将真正复杂的业务做成，看上去会很简单的，因为我之前也看过一些一些系统设计，比如说什么流程编排，但是我个人觉得流程编排，虽然说是加一些业务，完全解耦开来了，但是他好像又又又带来了一个隐患，就是你对业务系统不熟，你对这个系统不熟之前，你会让后来的维护的人，他似乎有有一点点麻烦。不知道我的理解对不对。

胤禛

26:38

首先你提到的一个通用性的一个问题，其实这个也是系统设计的一个问题，就是因为的差异性和共性的这样一个共同的问题，因为也像你说的差异性不可能，就是说我一对一的来个需求就对应我肯定是要做一个这样的框架集的东西对？所以首先，框架及东西框架集的东西，首先要把那个分布式的问题，包括业务的问题全部封装里面到里面去。当然这是一个里面要进行抽象，那抽象的最基本的就是说，假设，我在下单的时候，我不需要去关心，这个就是如果我是开发的话，我去调这个接口，我不需要关心后面的怎么去，比如说他的数据一致性，首先数据一致性，首先分两盘哈，就刚才的那个，你用什么2PC去调用，或者是这个调用，其实有本身的问题，因为网络会抖动，包括下游的网络，可能不稳定，服务也不稳定，所以首先要解决一致性问题，蹦依赖于数据库，因为如果依赖于数据库的话，还会引入一些分布式事务的问题，你的性能会下降，对？这是第一个问题。所以你要把这个分布式事物进行解耦，结耦之后，就是第一个，你写的时候，你首先要把你写的这个状态就是a服务，比如说a服务调B服务，你把a的服务所产生的这个后果，业务的结果先写不到一个分布式缓存，因为分布式缓存是立即可见的，也就是B。不过马上就能看得到对？然后，你把a的那个结果儿，存到你本地的是物理去就本地的数据库，就是本地的数据库，已经给你回执了，说已经已经存存OK了，这时候你缓存的也同步了，就是BB服务马上就能看到，所以说它整个的业务应该是一致的对？然后，下面下面，他在通过一个比如说通过condo的一个中间件来观察你的来watch你的这个整个儿就是a服务的整个的服务状态，然后把这个状态再同步给B的数据库，也就是异步的，因为这里面就住了分离，首先是把分布式事务降解成本地事务，就是两边都是本地事务。第二个，就是说我不关心数据是不是同步的，我只关心我们看到他，我是通过分布式缓存就看到他立马是同步的底层的数据，可以异步的同步，最终一致性就OK了，这样的话你就保证了一个是数据最终是可以一致的，业务本身，就是锵锵湿湿的。

胤禛

29:06

不会有任何问题，这是第一个层次的问题。第二个层次的问题，你要保证多个服务，因为我刚才刚才提到了这个工作流编排，他是12345678，这么多个动作，对，怎么保证这个动作，因为这个动作里面隐含一个信息，这些动作是只读向前的，他不能回退。

小猛

29:18

对。

胤禛

29:24

对，所以说你这里面也要做一个状态机，这个状态机就是你每做一个状态，没每执行一步的时候，他把这个状态，它是个分布式，它有分布式锁，那么他会把这个状态的准确的就是同步给数据库，因为他有分布式锁，可以保证这个并发的情况下，只有一个客户能拿到这个说对，然后再把这个状态内置了以后。状态机的状态也改变了，那么第二个客户，比如说用户，因为这个病发的环境，在读取这个接口的时候，因为状态变化了，就把它rejection。所以说你你整个的这个过程，整个过程是受这一个强一致的这个状态继续去管控的，所以说这时候这时候，你能达到你在外部访问的时候，你不需要关心这些事情，你你可以保证他返回的状态，成功或者失败一定是成功或者失败。

胤禛

30:17

对，所以第二个，就是说在业务上要抽象的，就是里面要要分解出来，分解出来业务的哪些认为是拓展点。比如说你支付的时候，支付的时候场景不同，你把场景全列出来，每个场景我认为比如说支付的时候，我是用美金还是用日元，还是什么支付方式比较，那这个地方我可以做个拓展点对？然后，还有支付的时候，可能就说要清关，因为。

小猛

30:40

对。

胤禛

30:45

我购买的是一个海外的产品，或者是国内国内是不需要清关的，那清关这一块儿的环节，我可以做成一个拓展点，对？那么这些拓展点，我们可以通过接口拓展出来，然后，比如说业务方，他有自己特殊的一些流程，但是流程你可以通过这个业务流程编排，但是这个拓展点他可以拓展出来，里面加上自己的逻辑，加上自己的逻辑以后，就可以去应对它不同的业务场景，但是这个地方要注意，你的主干肯定是没有变的。这都是通过接口拓展出去的，他只是在运行泰的时候给你做了动态的替换，动态的拓展而已，对？所以这个地方就能保证你这个平台是整个是抽象的。然后，是那个就是所有的动作都是原子化的，就是不会不会因为不一致性产生两边的就是钱和货不一致了，对，然后你的所有的流程都是固化的，固定的，你只需要去拓展他就OK了。

小猛

31:43

明白。听听听老师这。所以这一番。