36 | 浅谈测试数据的痛点

2018-09-19 茹炳晟

软件测试52讲 进入课程 >

讲述：茹炳晟
时长 13:22 大小 6.13M

你好，我是茹炳晟。今天我和你分享的主题是：浅谈测试数据的痛点。

在上一篇文章中，我和你分享了创建测试数据的四大类方法，即基于 GUI 操作生成测试数据、通过 API 调用生成测试数据、通过数据库操作生成测试数据，以及综合运用 API 和数据库的方式生成测试数据。

但是，我并没有谈到应该在什么时机创建这些测试数据。比如，是在测试用例中实时创建测试数据，还是在准备测试环境时一下子准备好所有的测试数据呢。

其实，在不同的时机创建测试数据，就是为了解决准备测试数据的不同痛点。那么，准备测试数据的痛点，都体现在哪些方面呢？

在测试用例执行过程中，创建所需的数据往往会耗时较长，从而使得测试用例执行的时间变长；

在测试执行之前，先批量生成所有需要用到的测试数据，就有可能出现在测试用例执行时，这些事先创建好的数据已经被修改而无法正常使用了的情况；

在微服务架构下，测试环境本身的不稳定，也会阻碍测试数据的顺利创建。

那么，今天我们就先来聊聊与测试数据创建时机相关的话题。

从测试数据创建的时机来看，主要分为 On-the-fly（实时创建）和 Out-of-box（事先创建测试数据）两类方法。这两类方法都有各自的优缺点，以及适用的最佳场景。而且在工程实践中，我们往往会综合使用这两种方法。

接下来，我先和你分别介绍一下这两类方法。其实，这两类方法我已经在第 15 篇文章《过不了的坎：聊聊 GUI 自动化过程中的测试数据》中提到过了。但是，当时我只是笼统地和你分享了这两类方法的概念，并没有详细展开讨论。所以，我今天就会通过一些实例，和你更加详细地讨论这两类方法。

On-the-fly

On-the-fly 方法，又称实时创建方法，指的是在测试用例的代码中实时创建要使用到的测试数据。比如，对于用户登录功能的测试，那么在测试用例开始的部分，首先调用我在上一篇文章中介绍的创建新用户的数据准备函数来生成一个新用户，接下来的测试将会直接使用这个新创建的用户。

对于 On-the-fly，测试用例中所有用到的测试数据，都在测试用例开始前实时准备。**采用 On-the-fly 方式创建的数据，都是由测试用例自己维护的，不会依赖于测试用例外的任何数据，从而保证了数据的准确性和可控性，最大程度地避免了出现“脏”数据的可能。**

那到底什么是“脏”数据呢？这里的“脏”数据是指，数据在被实际使用前，已经被进行了非预期的修改。

从理论上来讲，这种由自己创建和维护数据的方式，是最佳的处理方式，很多早期的测试资料都推荐采用这种方式。但是，随着软件架构的发展，以及软件发布频率的快速增长，这种方式的弊端越来越明显，主要体现在以下三方面：

首先，实时创建测试数据比较耗时。在测试用例执行的过程中实时创建测试数据，将直接导致测试用例的整体执行时间变长。

我曾统计过一个大型电商网站的测试用例执行时间，总的测试用例执行时间中，有 30%-40% 的时间花在了测试数据的实时准备上，也就是说测试数据的实时准备花费了差不多一半的测试用例执行时间。

对传统软件企业来说，它们可能并不太在意这多出来的测试执行时间，因为它们的软件发布周期比较长，留给测试的时间也比较长，所以这多出来的时间可以忽略不计。

但是，对于互联网软件企业来说，它们的软件发布频率很高，相应地留给测试执行的时间也都很短，那么缩短测试数据的准备时间的重要性就不言而喻了。

要解决创建测试数据耗时的问题，除了从测试数据准备函数的实现入手外，还可以考虑采用我后面要介绍的事先创建测试数据 Out-of-box 的方式。

其次，测试数据本身存在复杂的关联性。很多时候你为了创建一个你需要使用的业务数据，往往需要先创建一堆其他相关联的数据，越是业务链后期的数据，这个问题就越严重。

比如，创建订单数据这个最典型的案例。由于创建订单的数据准备函数需要提供诸如卖家、买家、商品 ID 等一系列的前置数据，所以你就不得不先创建出这些前置数据。这样做，一方面测试数据准备的复杂性直线上升，另一方面创建测试数据所需要的时间也会变得更长。

为了缓解这个问题，你可以考虑将部分相对稳定的数据事先创建好，而不要采用 On-the-fly 的方式去创建所有的数据。

最后一个问题来自于微服务架构的调整。早期的软件架构都是单体的，只要测试环境部署成功了，那么所有的功能就都可以使用了。而现如今，大量的互联网产品都采用了微服务架构，所以，很多时候测试环境并不是 100% 处于全部可用的状态。也就是说，并不是所有的服务都是可用的，这就给测试数据准备带来了新的挑战。

比如，你为了测试用户登录功能，根据 On-the-fly 的策略，你首先需要创建一个新用户。假设在微服务架构下，注册用户和用户登录隶属于两个不同的微服务，而此时注册用户的微服务恰好因为某种原因处于不可用状态，那么这时你就无法成功创建这个用户，也就是无法创建测试数据。因此，整个测试用例都无法顺利执行，显然这不是我们想要的结果。

为了解决这个问题，你可以采用事先创建数据 Out-of-box 的方式，只要能够保证测试环境在某个时间段没有问题，那么就可以在这个时间段事先创建好测试数据。

为了解决上述三个问题，Out-of-box（即事先创建测试数据）的方式就应运而生了。那么，

接下来我们就一起看看这个方式的原理，以及适用的场景吧。

Out-of-box

Out-of-box 方法，又称开箱即用方法，指的是在准备测试环境时就预先将测试需要用到的数据全部准备好，而不是在测试用例中实时创建。因此，我们可以节省不少测试用例的执行时间，同时也不会存在由于环境问题无法创建测试数据而阻碍测试用例执行的情况。也就是说 Out-of-box 方法可以克服 On-the-fly 方法的缺点，那么这个方式又会引入哪些致命的新问题呢?

Out-of-box 最致命的问题是“脏”数据。

比如，我们在测试用例中使用事先创建好的用户进行登录测试，但这个用户的密码被其他人无意中修改了，导致测试用例执行时登录失败，也就不能顺利完成测试了。那么，此时这个测试用户数据就成为了“脏”数据。

再比如，我们在测试用例中使用事先创建的测试优惠券去完成订单操作，但是由于某种原因这张优惠券已经被使用过了，导致订单操作的失败，也就意味着测试用例执行失败。那么，此时这个测试优惠券数据也是“脏”数据。

由此可见，这些事先创建好的测试数据，在测试用例执行的那个时刻，是否依然可用其实是不一定的，因为这些数据很有可能在被使用前已经发生了非预期的修改。

这些非预期的修改主要来自于以下三个方面：

1. 其他测试用例使用了这些事先创建好的测试数据，并修改了这些数据的状态；
2. 执行手工测试时，因为直接使用了事先创建好的数据，很有可能就会修改了某些测试数据；
3. 自动化测试用例的调试过程，修改了事先创建的测试数据；

为了解决这些“脏”数据，我们只能通过优化流程去控制数据的使用。目前，业内有些公司会将所有事先创建好的测试数据列在一个 Wiki 页面，然后按照不同的测试数据区段来分配使用对象。

比如，假设我们事先创建了 1000 个测试用户，那么用户 ID 在 0001-0200 范围内数据给这个团队使用，而用户 ID 在 0201-0500 范围内的数据则给另一个团队使用。这个分配工作，要靠流程保证，那么前提就是所有人都要遵守这些流程。

但我一直认为，但凡需要靠流程保证的一定不是最靠谱的，因为你无法确保所有人都会遵守流程。也正是因为这个原因，在实际项目中我们还是会经常看到由“脏”数据引发测试用例执行失败的案例。

更糟糕的是，如果自动化测试用例直接采用硬编码的方式，去调用那些只能被一次性使用的测试数据（比如订单数据、优惠券等）的话，你会发现测试用例只能在第一次执行时通过，后面再执行都会因为测试数据的问题而失败。

所以，你还需要在测试用例级别保证测试数据只被调用一次，而这往往会涉及到跨测试用例的测试数据维护问题，往往实现起来非常麻烦。所以说，Out-of-box 方法不适用于只能一次性使用的测试数据场景。

综合运用 On-the-fly 和 Out-of-box

为了充分利用 On-the-fly 和 Out-of-box 这两种方式的各自优点，并且规避各自的缺点，实际的工程实践中，往往是采用综合运用 On-the-fly 和 Out-of-box 的方式来实现测试数据的准备的。

在实际的测试项目中，我们可以根据测试数据的特性，把它们分为两大类，用业内的行话来讲就是“死水数据”和“活水数据”。

“死水数据”是指那些相对稳定，不会在使用过程中改变状态，并且可以被多次使用的数据。比如，商品分类、商品品牌、场馆信息等。这类数据就非常适合采用 Out-of-box 方式来创建。

这里需要特别说明的是，哪些数据属于“死水数据”并不是绝对的，由测试目的决定。

比如，用户数据在大多数的非用户相关的测试用例中基本属于“死水数据”，因为绝大多数的业务测试都会包含用户登录的操作，而且并不会去修改用户本身的数据属性，所以这时我们就可以将用户数据按照“死水数据”处理，也就是采用 Out-of-box 的方式创建。

但是，对于那些专门测试用户账号的测试用例来讲，往往会涉及到用户撤销、激活、修改密码等操作，那么此时的用户数据就不再是“死水数据”了，而应该按照“活水数据”处理。

“活水数据”是指那些只能被一次性使用，或者经常会被修改的测试数据。最典型的数据是优惠券、商品本身、订单等类似的数据。这类数据通常在被一次性使用后状态就发生了变化，不能反复使用。那么这类测试数据，就更适合采用 On-the-fly 自维护的方式。

同时，由于有 Out-of-box 数据的支持，这类数据往往不需要从最源头开始创建，而是可以基于已有的 Out-of-box 数据生成。

比如，在使用 On-the-fly 方式创建订单数据时，你可以直接使用 Out-of-box 的用户数据来作为买家数据。

由此可见，综合运用这两类方法，可以以互补的方式解决测试数据准备的很多痛点，比如测试数据准备比较耗时、测试数据存在“脏”数据的可能，以及测试环境不稳定造成的测试数据无法创建等问题。

总结

今天我从测试数据创建时机的角度，和你分享了 On-the-fly 和 Out-of-box 这两类创建数据的方式。

On-the-fly 方法又称为实时创建方法，指的是在测试用例的代码中实时创建测试用例所要使用到的测试数据，具有数据可靠性高的优点，但是会比较耗时。

而 Out-of-box 方法又称为开箱即用方法，指的是在准备测试环境时就事先准备好测试需要用到的全部数据。这样可以有效缩短测试用例的执行时间，但是存在“脏”数据的问题。

最后，我从“死水数据”和“活水数据”的角度讨论了如何综合运用上述两种方式创建测试数据，其中“死水数据”适合用 Out-of-box 的方式，而“活水数据”适合采用 On-the-fly 的方式。

思考题

你所在的项目中，采用的是什么样的测试数据准备策略，这个策略的优缺点是什么？为什么会选择这样的策略呢？另外，你所在团队会使用线上真实的数据进行测试吗？

感谢你的收听，欢迎你给我留言。

上一篇　35 | 如何准备测试数据?

下一篇　37 | 测试数据的“银弹” - 统一测试数据平台（上）

精选留言 (14)

写留言

better ...

2018-09-20

14

老师我有个小困惑😳，从第一课到现在，我总有一种思路打开，但是无从下手的感觉，于是我思考了一下，有个小小建议，希望老师讲解的时候理论加业务场景/真实代码实现等具体实践相结合，比如讲到ui自动化框架，可以有几个小demo我能跟着去操作的，这样更有

利于我们的吸收和掌握，然后举一反三得去运用到工作中去，不知大家有没有这种感觉。

展开

sylan215
2018-09-19

2

1.《软件测试的艺术》艺术中提出的「软件测试的原则」中的第一条就是「测试用例中一个必需部分是对预期输入或结果的定义」，测试数据就是预期输入了吧。

2.对于服务端这种公用的数据，建议统一提前准备，就是茹老师说的 Out-of-box 方法，但是对于客户端数据，因为每个人执行的环境不一样，有些环境就是实时准备的，所以...

展开

Hunter
2018-10-12

1

还有一点是避免用例运行是引入脏数据，有时候需要注意恢复环境。比如一个用例创建了一个用户，下次再运行创建用户的用例时，就会因为名称重复报错。

展开

口水窝
2019-05-13

一般在做单接口测试时会采用On-the-fly模式，采用压力测试时会采用Out-of-box方式收集数据。进行线上预发布的时候，使用的是线上的真实数据，只是跟生产环境地址不同而已。

展开

年轻人的瞎...
2019-01-09

一般都是两者相结合，由于是分布式架构，都会从各种微服务获取数据，。

展开

作者回复: 嗯嗯，要取决于业务本身是如何设计的

静静张
2018-11-18

我们针对服务端的接口测试需要活水数据，用例执行前构造数据，执行后清除数据。
尽可能保证用例之间互不影响，同时避免脏数据的产生。
前提是我们的规模小，哈哈

小老鼠
2018-11-07

测试环境一定要独立开发环境与运行环境

展开

小老鼠
2018-11-07

可不可以每次执行前先用Out-of-box创建数据，然后再执行测试用例。在Teardown 中消除脏数据。但是在测试过程中发生异常，执行不了teardown方法，产生脏数据。如何办。

展开

胖虫子
2018-11-02

数据这个最麻烦

展开

希涛
2018-10-13

老师，在执行自动化测试的时候，肯定会生成很多测试数据，对于线上环境来讲，这些都是测试数据，一般怎么处理

蓝山
2018-10-09

我有一楼同样的困惑，听了大体思路，但无从下手。希望老师能举具体实例说明。

> 作者回复: 最先可以下手的部分就是先开始封装自己的测试数据准备函数，函数内部可以用api也可以用数据库，或者两者的结合

TJJ

2018-09-19

老师能不能举一些更详细的例子，比如电商模块，哪些适合插数据库，哪些要调接口造

Robert小七

2018-09-19

我们的数据都是手工创建

展开

青木

2018-09-19

老师能不能讲下大数据，机器学习这种效果评测的内容

展开