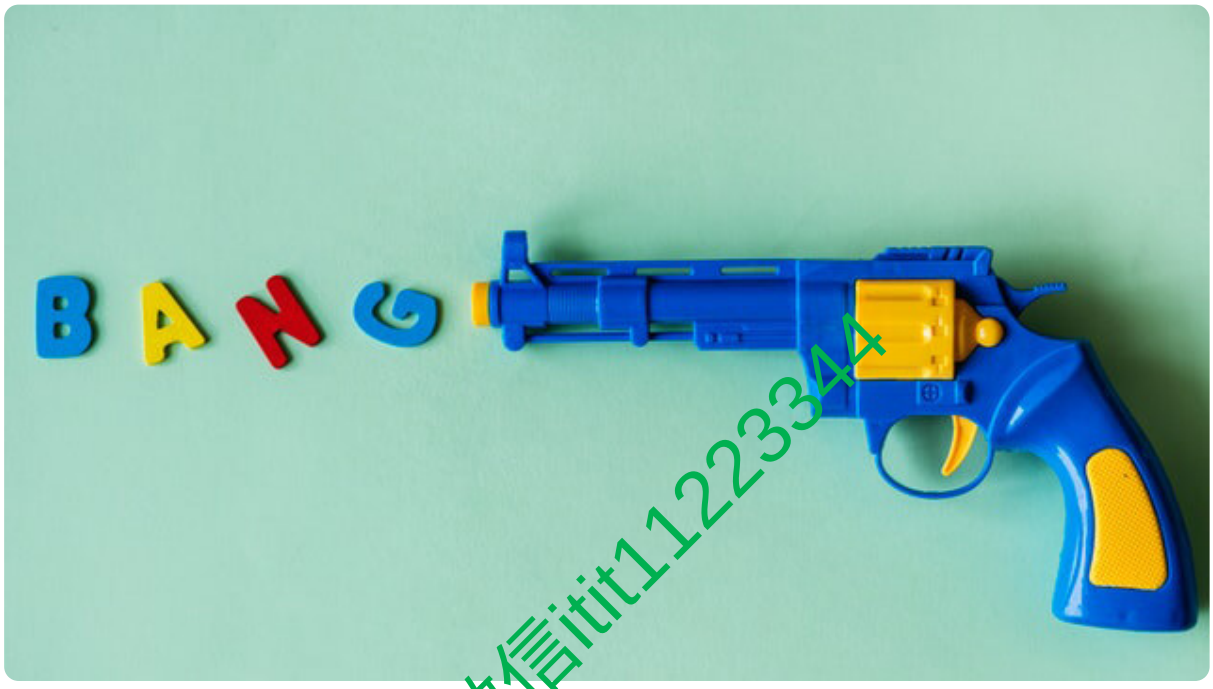
05 你不是一个人在战斗 - 遇到问题的正确姿势

更新时间：2020-05-29 18:25:54



> 勤能补拙是良训，一分辛劳一分才。——华罗庚

在我们实际测试学习的过程中，不管你是在进行功能测试、环境部署、代码编程、工具使用、框架搭建等等，都不可避免的遇到各种各样的问题。遇到问题，一有不顺利了就去找别人请教，这样是不可取的，非但透支了与他人的关系，更是对个人的学习和记忆不利。就像我们前边聊过的，懒惰的拿来主义并不能带来效率上的提高。

然而实际面对问题的时候，很多人都会突然不知所措，找不到任何头绪。那么应该怎么办呢？我在这里简单总结一下测试遇到问题的一些常见的排查方法。

确认是否环境问题

在大家实际遇到的问题中，我大致总结了一下，大约有 50% 的问题是由于环境引发的。比如，

能用，启动报错。其实仔细排查一下就会发现，由于本地已经存在一个运行中的 MySQL 服务，占用了 3306 端口导致启动失败。

再例如，我们使用 LoadRunner 录制脚本时，无法弹出 IE 浏览器，这时候需要考虑的环境因素就比较多了，包括操作系统版本、LR 版本、浏览器版本以及浏览器设置。当然，我了解，大家很多对这些环境问题不甚了解，自己很难排查。我这里推荐大家一个很简单的判断方法，就是当遇到工具安装和基础使用的问题，而明明网上以及老师都可用只有你不可用时候，那么八成就是环境问题，先去查询排查一下。

错误日志的作用

不知道是不是因为我的要求过高导致的，我在去很多企业做内训、以及线上线下学员沟通都发现一个通病，80% 以上的测试人员不去看日志。很多人回答我说：测试为什么还要看日志，找开发不就好了？这是一种很浅显的理解，觉得测试人员最重要的就是发现问题，至于找问题，NoNo，那就交给开发人员搞定吧。

我们从两个角度来说日志的作用。先说我们在测试的过程中发现了 Bug，既然测试人员不需要修正 Bug，那么到底还需不需要能看懂日志，需不需要找到问题发生的点和原因呢？我个人觉得是十分必要的。在中小型公司里，之所以测试不受重视主要在于测试只去做 “测” 的工作，而实际上在一线互联网公司里，测试人员需要承担的职责更多，不仅仅是发现问题，还包括定位问题、指明解决问题的方法。

通过对日志的分析，也能更进一步了解发现问题到底是由于环境、数据、上下游系统还是系统本身的 Bug，也避免了提出更多的无效问题。举个例子来说，当我们系统异常难以排查时候，我们发现日志中包含如下问题：

```
java.lang.Exception: java.lang.RuntimeException: code:EC_COMMON_NSTD_0112,message:用于计算分库分表位信息异常
        at com.hundsun.jrescloud.logger.aop.LogHandlerAspect.arround(LogHandlerAspect.java:75)
        at sun.reflect.GeneratedMethodAccessor166.invoke(Unknown Source)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.springframework.aop.aspectj.AbstractAspectJAdvice.invokeAdviceMethodWithGivenArgs(AbstractAspectJAdvice.java:627)
        at org.springframework.aop.aspectj.AbstractAspectJAdvice.invokeAdviceMethod(AbstractAspectJAdvice.java:616)
        at org.springframework.aop.aspectj.AspectJAroundAdvice.invoke(AspectJAroundAdvice.java:70)
        at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:168)
        at org.springframework.aop.aspectj.AspectJAfterThrowingAdvice.invoke(AspectJAfterThrowingAdvice.java:62)
        at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:168)
        at org.springframework.aop.interceptor.ExposeInvocationInterceptor.invoke(ExposeInvocationInterceptor.java:92)
        at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:179)
        at org.springframework.aop.framework.JdkDynamicAopProxy.invoke(JdkDynamicAopProxy.java:213)
        at com.sun.proxy.$Proxy105.selectList(Unknown Source)
        at sun.reflect.GeneratedMethodAccessor168.invoke(Unknown Source)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at com.alipay.sofa.rpc.filter.ProviderInvoker.invoke(ProviderInvoker.java:77)
        at com.alipay.sofa.rpc.filter.IpTransmitFilter.invoke(IpTransmitFilter.java:82)
        at com.alipay.sofa.rpc.filter.FilterInvoker.invoke(FilterInvoker.java:96)
        at com.alipay.sofa.rpc.filter.ProviderProfileFilter.invoke(ProviderProfileFilter.java:67)
        at com.alipay.sofa.rpc.filter.FilterInvoker.invoke(FilterInvoker.java:96)
        at com.alipay.sofa.rpc.filter.sofatracer.ProviderTracerFilter.invoke(ProviderTracerFilter.java:74)
        at com.alipay.sofa.rpc.filter.FilterInvoker.invoke(FilterInvoker.java:96)
        at com.alipay.sofa.rpc.filter.ProviderBaggageFilter.invoke(ProviderBaggageFilter.java:45)
        at com.alipay.sofa.rpc.filter.FilterInvoker.invoke(FilterInvoker.java:96)
        at com.alipay.sofa.rpc.filter.RpcServiceContextFilter.invoke(RpcServiceContextFilter.java:62)
```

Ok，这是我们在前端看不到的错误信息，通过这里，我们确定，这个错误是由于后端分库分表算法计算异常引起的。那我们可以更深入定位一下算法的问题，也可以直接将该算法存在问题提给开发，这样呢，就减少了很多开发排查问题的工作量，也更容易赢得开发的尊重。

上面的第一个角度，换个角度来说，我们测试人员自己也会在工作过程中碰到自己的问题，有的是代码端的，有的是工具端的。比如 Selenium 执行报错，比如 LoadRunner 录制报错，这个时候请求开发明显是更加不靠谱的，还是需要自己首先解决问题。Selenium 脚本执行时候有时候会遇到错误如下：

```
File "Q:\software\Python\Lib\site-packages\selenium\webdriver\remote\errorhandler.py", line 242, in check_response
  raise exception_class(message, screen, stacktrace)
selenium.common.exceptions.ElementNotVisibleException: Message: element not visible
  (Session info: chrome=67.0.3396.62)
```

我们找到核心关键字，有经验的同学这时候就知道了 elment not visible 可能出现的情况有两种，一是定位的标识不唯一，二是元素之间有所遮挡，所以我们要开始分别排查两种情况的可能性，这就是我们分析问题的过程，处处离不开对日志的分析。Jmeter、LoadRunner 这样的商用工具同样有自己的错误提示，同样可以帮助我们去分析问题。

善用搜索引擎

其实这一点算是前边内容的延伸，既然要独立解决问题，就少不了搜索相关问题和解决办法。曾经跟测试圈里的某资深专家聊天的时候提到过：其实现在绝大多数的人不会用搜索引擎，现在回想一下，深以为然。我猜可能有很多同学会觉得不服气，但是最近至少几年我遇到的很多学员的问题都是由于 “不会搜索” 造成的，比如说：最近有个同学问我，他想找一段随机产生手机号的代码，可是怎么搜都搜不到。我觉得比较好奇，就追问他自己是怎么搜的，他说搜索 “随机手机号”，出来都是什么生成器啊之类的，没有可用的代码。我告诉他，试着在前边加个 Java？果然，很快找到了。

这个例子很小，但是却反映了大家使用搜索引擎的最大的问题：**不会找关键字。**

不够精简

有很多人很喜欢直接在搜索引擎里边搜 “如何...”、“怎么办...”、“怎么才能... 呢” 这样的关键字，尽管一般的搜索引擎都会过滤一些语气词，也能够搜到一些结果，但是精简检索式无疑是提高检索效率最好的方法。

至于方法可以比较简单，把你想提问的问题写下来，把其中的形容词、疑问词都筛掉，只留下大多数的名词就可以了，可以填充一些必要的修饰。比如你要搜索 “哪个自动化测试工具最好用”，可以直接筛选成为 “自动化测试 好用” 就可以了。

缺少信息

和不够精简相对的，就是缺少信息。其实最前边的 “随机手机号” 就是很好的例子。在搜索过程中一定要标识自己的需求。如果我们要搜索某一个算法，一定要加上语言或者 “算法” 等字样标识，类似 “Java XXX”、“XXX 算法” 或者 “XXX 代码”，这样会准确很多。

筛除杂质

很多同学找到日志中或者是控制台的报错，看不懂直接就扔到了搜索引擎里边去搜，发现找不到任何结果。这是为什么呢？搜索前一定要筛选掉你报错信息里的杂质，拿最纯正的错误去搜索。什么是杂质？我们特有的类名、变量名、参数值、报错信息中的中文都会是一些杂质，需要筛除掉。我们简单来看一个报错：

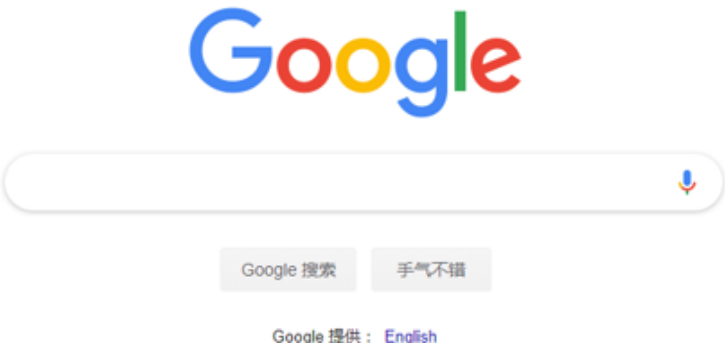
```
ERROR .System.ServiceModel.EndpointNotFoundException: The connection attempt lasted for a time span of 00:00:00.9375060. TCP error code 10061: No connection could be made because the target machine actively refused it 10.18.3.1:9000. --->
System.Net.Sockets.SocketException: No connection could be made because the target machine actively refused it 10.18.3.1:9000
```


在这个报错里，需要筛选掉的信息包括各种 ip 等等，实际更好的搜索内容为：No connection could be made because the target machine actively refused it。这样才能获得最适合的结果。

专业精准

当我们寻找某个领域的内容时候，越是用大而泛的词语，搜索出来的结果多而浅，而越是用专业而精准的关键词，得到的结果少而精。比如我们现在要测试的是一个大数据模型，可是现在不知道如何去测，当你搜索 “大数据测试” 的时候，你会得到很多宽泛的结果，而当你了解了你的模型，转而去搜 “Hadoop 测试”、“MapReduce 验证” 等关键字时候往往能够得到更精确的信息。

除了不会找关键字的问题以外，其实我们在搜索问题时可以尝试不同的搜索引擎。“兼听则明偏信则暗”，兼容 Google 和百度才是技术人的大成，因为 Google 更容易搜索到国外一些论坛的解决方案，而百度中国内的论坛 — 知乎、CSDN、简书等的权重更高。



当然，即便是搜索的结果，也不可尽信。老话讲 “择其善者而从之，其不善者而改之” 是十分有道理的，所以搜索到解决方案了，最重要的是分析是否适用和尝试是否可以解决。在网上找东西是一个愉快的过程，有一点像在 “淘宝”，不但需要火眼金睛，还需要与浩瀚的信息斗智斗勇。

我在这里简单列一下我自己觉得搜索结果相对可靠的一些网站，仅供参考：

1） 国外论坛

Stack Overflow、Reddit、CodeProject、Bytes

2） 国内网站

CSDN、简书、知乎

排名不分先后，如果大家在查询问题时候发现上述网站的结果，可以优先点开看看。其实我们工作中测试中 80% 的问题，都已经被别人遭遇过了，我们会搜索就可以解决其中的大半，剩下的 20% 才是研究，世上无难事，只要会搜索。

寻求帮助

如果经过我们的分析、查找、尝试仍然没有解决的话，那么就有必要向他人求助了，求助也就是

“提问的艺术”。

← 04 努力≠高效-测试高效学习浅谈 06 提问的艺术 →

精选留言 3

欢迎在这里发表留言，作者筛选后可公开显示

ZuanMoc

留言开始少了，哈哈

0 回复 2020-09-28

Python工程师

勤能补拙是良训,一分辛苦一分才.

1 回复 2020-01-31

小酱酱

风落老师，好崇拜你呀，思维果然不在一个层面，感觉一定会有很多收货

4 回复 2019-09-11

风落几番 回复 **小酱酱**

谢谢谢谢，像我们这么容易骄傲的少年，面对夸奖还是难以抑制内心的喜悦啊！！

回复 2019-09-16 11:00:10

千学不如一看，千看不如一练