



这些BUG，防不胜防

软件测试杂谈 2024-03-25 24 阅读8分钟 专栏：python和测试

智能总结

复制 重新生成

文章主要总结了常见的六大类时间 case 及防护手法，包括时间转换、夏令时、时间精度、时间比较、时间存储、权限类。每类都通过模拟场景揭示问题原因，并分别从开发设计、测试等阶段提出防护思路。

关联问题: 时间转换怎样测试 夏令时计算注意啥 时间精度咋校验

基于该文章内容继续向AI提问

常见时间case与防护分析

话不多说，上干货！笔者经过长年累月的积累，针对常见的时间case已在本文中总结出六大类。

第一类

时间转换类

模拟场景：张三在北京时间5月20日零点使用A国跨境电商APP，预备参加520情人节零点当天开启的限时优惠活动，下单礼物送给自己的女朋友，一顿操作下单成功预备获取返利时，发现订单截图中，下单时间并不是5月20日零点而是5月19日，男子难道穿越时空了？

原因：他当然没有穿越，这是一个典型的、比较容易发生在前端的计算显示错误。该跨境电商app的服务端接口计算、存储用的是A国地区时区的时间，前端展示未转换成男子所在地区时区时间展示导致。

防护手法：

- 开发设计阶段常见防护思路：时间戳转换成年月日时间格式展示的时候前后端时区要保持一致，我们常用的时区是东八区（UTC+8），根据需要转换对应时区；年月日时分秒格式字符串转换成time格式用的方法如果是time.Parse默认用的是UTC，跟北京时间相差八小时，所以一般使用ParseInLocation指定时区转换。
- 测试阶段固防思路：客户端/WEB 需要切换时区进行测试，判断最终时间实现逻辑是否符合产品需求。

第二类

夏令时类

模拟场景：张三在某网站上输入自己出生年月想测算自己年龄星座运势等信息，发现输入1986年7月8日后计算的年龄永远大一小时，星座运势无法准确计算，张三兴奋不已，难道他是经历了未知时间虫洞的天选之子？

原因：张三真的想多了。这是计算错误，中国曾在1986~1991这6年间实施过夏令时，具体做法是每年从四月中旬第一个星期日的凌晨2时整（北京时间），将时钟拨快一小时，即将表针由2时拨至3时，夏令时开始；到九月中旬第一个星期日的凌晨2时整（北京夏令时间），再将时钟拨慢一小时，即将表针由2时拨至1时，夏令时结束。这样会导致夏令时开始日实际只有23 小时，而夏令时结束日有25小时。而1986年5月04日至9月14日又属于夏令时的日期范围，解析日期的时候出现时间偏差值。

防护手法：

- 开发设计阶段防护思路：夏令时日期范围有：1986年5月4日至9月14日（1986年因是实行夏令时的第一年，从5月4日开始到9月14日结束）；1987年4月12日至9月13日；1988年4月10日至9月11日；1989年4月16日至9月17日；1990年4月15日至9月16日；1991年4月14日至9月15日。

当遇到这些特殊日期计算的时候需要注意计算时间段内有夏令时的时候，在夏令时日期的0点0分，采用夏令时，时间会调整为1点；在夏令时期间的时间应该为当时的时间减去一小时，将调快的一小时减回来。

- 测试阶段防护思路：校验点需要查看涉及夏令时日期的场景，格外注意下特殊场景（一天会出现23小时、25小时场景）。

第三类

时间精度类

模拟场景：张三在某视频APP上观看1分钟后点开观看记录发现刚才自己竟观看了1小时视频，时间怎么过的这么快？

原因：张三又一次想多了，服务端接口返回值里的时间数值精度为秒级，前端获取以后展示出来的接口的秒级时间数值，但是前端展示单位却固定写死的时间精度为小时，导致用户误解。

防护手法：

- 开发设计阶段防护思路：在设计代码阶段注意时间精度的转换公式：

Nanosecond Duration = 1 //纳秒

Microsecond = 1000 * Nanosecond //微秒

Millisecond = 1000 * Microsecond //毫秒

Second = 1000 * Millisecond //秒



软件测试杂谈

LV.4

160

文章

20k

阅读

36

粉丝

关注

私信

相关推荐

修改他人代码：怎么才能减少发布Bug...

1.3k阅读 · 4点赞

和AI网聊10分钟被骗430万，真实诈骗...

3.4k阅读 · 21点赞

拒绝代码搞怪 帮程序员拨“bug”反正

49阅读 · 0点赞

记录一次坑爹的问题

281阅读 · 1点赞

系统故障防不胜防？不存在的，让大佬...

112阅读 · 0点赞

精选内容

总结：Helm 命令详解

夕颜111 · 14阅读 · 0点赞

JUC并发—5.AQS源码分析一

东阳马生架构 · 14阅读 · 0点赞

MybatisPlus----构造器wrapper的使用...

Aska_Lv · 40阅读 · 4点赞

MybatisPlus----SQL注入器提升批量插...

Aska_Lv · 37阅读 · 5点赞

鸿蒙轻内核M核源码分析系列二一 03 文...

别说我什么都不会 · 14阅读 · 1点赞

找对属于你的技术圈子

回复「进群」加入官方微信群



Minute = 60 * Second //分

Hour = 60 * Minute //小时

2.测试阶段防护思路：校验点1： 时间单位是否统一（s、ms、min、h、天）；校验点2： 跨年、跨月、跨日的场景。

第四类
时间比较类

模拟场景： 老板打开OA系统查看昨日15点到今日15点的会议安排信息，发现展示为空，遂未做会议行程安排，导致错失百万合同，是秘书老张失职了吗？

原因： 这个责任还是得“程序错误”来扛，在时间进行对比的时候需要统一时间基准点，如果没有基准点（比如例子中昨日15点是用北京时区时间，而近日15点是用美国时区时间）会导致无法进行正确的计算导致展示为空。

防护手法：

1.开发设计阶段防护思路：跟时区无关，使用的比较函数中会统一时间基准点（compareTime.Before、compareTime.After），注意24小时内、跨年、跨月、跨日、跨天等等的时间。

2.测试阶段防护思路：校验点需要注意24小时内选择不同时间进行比较测试、跨年选择去年跟今年的时间进行比较测试、跨月选择上个月底跟本月初的时间进行比较测试、跨日选择昨日跟今日的时间进行比较测试的。

第五类
时间存储类

模拟场景： 2000年跨晚期间众多平台发生“千年虫”事件，强如微软也遭遇了千年虫2022版：在1月1日0点0分期间的不少Exchange服务器的用户发现自己的祝福邮件没发出去，被留在了2021年。

原因： 古早的千年虫问题产生的原因是由于在计算机软、硬件以及数字化程序控制芯片的各种设备和业务处理系统中，只使用了两位十进制数来表示年份，因此，当日期从1999年12月31日进入2000年1月1日后，系统将无法正常识别由“00”表示的2000年（计算机可能将这个年分识别为1900年）这一具体年份，从而带来进行跨世纪的年份、日期处理时的计算错误，引发各种各样的计算机业务处理系统和控制系统的功能紊乱[1]，之所以千年虫有2022版本是因为部分公司采用了一种名叫“yymddHHMM”的有符号变量（Int32）来存储日期。但是，因为该变量最多只能存储-2,147,483,647到+2,147,483,647的数据，而2022年1月1日午夜的新日期值为2,201,010,001，超过了这个范围。

防护手法：

1.开发设计阶段防护思路：以mysql数据库存储为例，mysql存储时间通常用类型：a、timestamp实际存储的是带时区的时间 b、datetime存储的就是格式化后的字符串不携带时区信息 c、TIME 用于表示时分秒，如果实际应用值需要保存时分秒 就可以使用 TIME。有正负数，负数存储两个日期时间相减。

2.测试阶段防护思路：用例设计时根据需求选择合适的时间存储方式。

第六类
权限类

模拟场景： 张三在23年第一天进入公司打卡上班时发现，工牌无法被识别提示权限不足，该不会被“原地被离职”了吧？

原因： 张三遇到了经典的时间权限类问题，整个系统的有效期被设置在2022年12月31日晚23:59:59秒结束。在打卡权限功能需要针对指定时间段授权，当设计时没有考虑授权时间有效期过期的情况时，就会出现该授权失败问题。

防护手法：

1.开发设计阶段防护思路：设计时间权限功能需要注意考虑紧急case，针对功能或用户个体开设黑白名单来应对。

2.测试阶段防护思路：注意时间跨度范围内跟范围外的用例设计。

3.运营阶段防护思路：增加权限监控和提醒机制，便于提早发现问题。



总结

针对常见时间case的防护分析告一段落，更多内容分享还待笔者后续持续探索跟踩坑后再与大家分享讨论。

更多内容可以学习《测试人的 Python 工具书》书籍、《性能测试 JMeter 实战》书籍

标签：

测试

话题：

每天一个知识点

本文收录于以下专栏



python和测试

专栏目录

分享python和测试

16 订阅 · 155 篇文章

订阅


上一篇

测试工作的价值体现

下一篇

yarl, 一个神奇的 Python 库!

评论 0



登录 / 注册

即可发布评论!



暂无评论数据

为你推荐

关于时间国际化的方案

语石 | 3年前 | 👁 1.4k | 👍 6 | 💬 2

Java

前端的计时是否准确，技术方案如何做

余杭子曰 | 4年前 | 👁 7.7k | 👍 40 | 💬 5

前端

关于移动端js倒计时

一只放羊的前端 | 2年前 | 👁 1.1k | 👍 2 | 💬 评论

前端

项目中的时间转化实战

卢卡多多 | 3年前 | 👁 405 | 👍 1 | 💬 评论

后端

面试

记录一次生产问题排查

苏凉 | 3年前 | 👁 676 | 👍 2 | 💬 评论

后端

前端倒计时二三事

Ambber | 4年前 | 👁 974 | 👍 6 | 💬 4

JavaScript

监控和告警 | 网站被攻击了？

腾讯云云开发 | 3年前 | 👁 1.4k | 👍 3 | 💬 评论

后端

领导：谁再用定时任务实现关闭订单，立马滚蛋！

东边有耳 | 3年前 | 👁 79k | 👍 536 | 💬 256

架构

后端

java web-正确处理时间和时区问题

TOPEN | 3年前 | 👁 1.6k | 👍 15 | 💬 评论

后端

解决iOS系统时间被修改的问题

iOS_大书 | 3年前 | 👁 3.1k | 👍 22 | 💬 4

iOS

从时区数据库的角度理解时区和夏令时

VeSync技术 | 1月前 | 👁 71 | 👍 点赞 | 💬 评论

后端

超时时间应该设置多长？长尾请求和重试风暴又该如何解决？打造高效稳定的网络请求！

牛肉烧烤屋 | 6天前 | 👁 80 | 👍 点赞 | 💬 评论

后端

微服务

Go每日一库之68：dateparse

luckzack | 1年前 | 👁 480 | 👍 1 | 💬 评论

Go

Go 每日一库之 dateparse

darjun | 3年前 | 👁 860 | 👍 3 | 💬 评论

Go

后端

H5嵌套到小程序、APP，还能埋点统计？看完这篇便知晓

哈罗哈皮 | 3年前 | 👁 1.6k | 👍 18 | 💬 2

JavaSc...

微信小...

