

107x: 不错, 谢谢!

[Latent Semantic Analysis\(L-SA/L-SI\)算法简介](#)

[Exception性能问题](#)

博客分类:

[问题](#)

[性能](#)

- 1.从Exception往上介绍相关结构、代码

class Exception里面没有什么新鲜东西, 它继承自class Throwable, 接下来我们看一下Throwable的结构, 在它的构造函数中调用了fillInStackTrace这个函数。接下来我们看看这个函数干了些什么。

fillInStackTrace函数的声明为

Java代码

```
1 | public synchronized native Throwable fillInStackTrace();
```

这是个native方法。

然后我们到jdk的代码里去找它的具体实现。

代码

```
01 | Java_java_lang_Throwable_fillInStackTrace(JNIEnv *env, jobject throwable)
02 | {
03 |     JVM_FillInStackTrace(env, throwable);
04 |     return throwable;
05 | }
06 |
07 | JVM_ENTRY(void, JVM_FillInStackTrace(JNIEnv *env, jobject receiver))
08 |     JVMWrapper("JVM_FillInStackTrace");
09 |     Handle exception(thread, JNIHandles::resolve_non_null(receiver));
10 |     java_lang_Throwable::fill_in_stack_trace(exception);
11 | JVM_END
12 |
13 | void java_lang_Throwable::fill_in_stack_trace(Handle throwable, TRAPS) {
14 |     if (!StackTraceInThrowable) return;
15 |     ResourceMark rm(THREAD);
16 |
17 |     -----
18 |     BacktraceBuilder bt(CHECK);
19 |
20 |     for (frame fr = thread->last_frame(); max_depth != total_count;) {
21 |         methodOop method = NULL;
22 |         int bci = 0;
23 |         -----
24 |         bt.push(method, bci, CHECK);
25 |         total_count++;
26 |     }
27 |
28 |     // Put completed stack trace into throwable object
29 |     set_backtrace(throwable(), bt.backtrace());
30 | }
```

上面的代码中, 这一系列调用可以发现, 当你new一个exception的时候, jvm已经在exception里构建好了所有的stacktrace (BacktraceBuilderbt), 这里花费的代价是可观的, 试想一下, 在web项目中, 调用栈的深度可是很大的。因此, 你对stacktrace不感兴趣的时候, 不需要这样的信息时, 最好不要随便的new exception。

这里介绍一个常用的避免这种问题的相应的解决方法, 即不需要stacktrace信息时, 抛自己定义的特殊exception。

自定义XXXException, 覆盖掉native的那个函数, 构造一个空的函数即可, 具体实现如下。

代码

```
1 | XXXException extends Exception {
2 |
3 |     public void    synchronized fillInStackTrace(){}
4 |
5 |     ...
6 |
7 | }
```



然后throw exception的时候, 抛自定义的XXXException就好了, 这样会大大的提高效率, 也节省了空间。

- 2.后记

当然做getStackTrace()的代价是蛮大的。曾经遇到一个案例, 只需要stacktrace中的某个trace, 却要通过getStackTrace()这个函数取到所有的trace, 取其中的第几个, 这样着实有些不划算。后来我们在jdk中给提供了一个接口StackTraceElementXXXUtils::getStackTraceElement(int index, Throwable t)便可以达到这个目的, 节约了不小的时间开销, 也省了内存。

2019软考实战秘籍

历年真题详解, 高效备战2019年软考

分享到:  

[solr基本概念](#) | [spring加载xml去远程获取dtd验证xml的问...](#)

2014-01-24 22:25

浏览 494

[评论\(0\)](#)

分类: [非技术](#)

[查看更多](#)

评论

发表评论



[您还没有登录, 请您登录后再发表评论](#)

相关资源推荐

[java非技术面试题（精心整理）](#)

[关于JavaException的基本知识和性能问题](#)

[Java面试_非技术问答](#)

[java面试技巧（非技术）之一](#)

java开发者顺利通过技术面试后, 也需关注一些常见面试问题及掌握面试应答技巧, 以便更好应对java面试。如何在这些简单、常见的面试问题中与其他面试者拉开差距? 笔者将从稳定性、学习能力、团队合作、应变能力、个人品行特性等方面与大家分享, 今天先与大家分享三个常见问题, 希望能引发大家的共鸣。 一、请做简单的自我介绍。 分析: 问题的本质一方面是想对面试者有个简要的了解; 一方面是HR争取时间快速浏览简历