

平凡希

博客园

首页

新随笔

联系

订阅

管理

随笔 – 127

文章 – 0

评论 – 385

阅读 – 301万

昵称： 平凡希
园龄： 11年
粉丝： 1865
关注： 1
[+加关注](#)

< 2022年1月 >						
日	一	二	三	四	五	六
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

搜索

找找看

谷歌搜索

常用链接

- 我的随笔
- 我的评论
- 我的参与
- 最新评论
- 我的标签

随笔分类

- Java IO(5)
- Java NIO(7)
- java8(3)
- Java多线程(20)
- java基础(11)
- Java集合(8)
- Java虚拟机(9)
- Mybatis(9)
- mysql(15)
- redis(9)

内存溢出与内存泄漏

一、基本概念

内存溢出：简单地说内存溢出就是指**程序运行过程中申请的内存大于系统能够提供的内存，导致无法申请到足够的内存**，于是就发生了内存溢出。

内存泄漏：内存泄漏指程序运行过程中**分配内存给临时变量，用完之后却没有被GC回收，始终占用着内存**，既不能被使用也不能分配给其他程序，于是就发生了内存泄漏。

内存溢出 out of memory，是指**程序在申请内存时，没有足够的内存空间供其使用**，出现out of memory；

内存泄露 memory leak，是指**程序在申请内存后，无法释放已申请的内存空间**，一次内存泄露危害可以忽略，但内存泄露堆积后果很严重，无论多少内存，迟早会被占光。

memory leak会最终会导致out of memory！

内存泄露是指**无用对象（不再使用的对象）持续占有内存或无用对象的内存得不到及时释放，从而造成的内存空间的浪费称为内存泄露**。内存泄露有时不严重且不易察觉，这样开发者就不知道存在内存泄露，但有时也会很严重，会提示你Out of memory。

二、内存溢出的常见情况

内存溢出有以下几种常见的情况：

1、java.lang.OutOfMemoryError: PermGen space (持久带溢出)

我们知道jvm通过持久带实现了java虚拟机规范中的方法区，而运行时常量池就是保存在方法区中的，因此发生这种溢出可能是运行时常量池溢出，或是由于程序中使用了大量的jar或class，使得方法区中保存的class对象没有被及时回收或者class信息占用的内存超过了配置的大小。

2、java.lang.OutOfMemoryError: Java heap space (堆溢出)

发生这种溢出的原因一般是创建的对象太多，在进行垃圾回收之前对象数量达到了最大堆的容量限制。

解决这个区域异常的方法一般是通过内存映像分析工具对Dump出来的堆转储快照进行分析，看到到底是内存溢出还是内存泄漏。如果是内存泄漏，可进一步通过工具查看泄漏对象到GC Roots的引用链，定位出泄漏代码的位置，修改程序或算法；如果不存在泄漏，就是说内存中的对象确实都还必须存活，那就应该检查虚拟机的堆参数-Xmx(最大堆大小)和-Xms(初始堆大小)，与机器物理内存对比看是否可以调大。

3、虚拟机栈和本地方法栈溢出

如果线程请求的栈深度大于虚拟机所允许的最大深度，将抛出StackOverflowError。

如果虚拟机在扩展栈时无法申请到足够的内存空间，则抛出OutOfMemoryError。

三、内存泄漏

内存泄漏的根本原因是长生命周期的对象持有短生命周期对象的引用，尽管短生命周期的对象已经不再需要，但由于长生命周期对象持有它的引用而导致不能被回收。

以发生的方式来分类，内存泄漏可以分为4类：

- 1、**常发性内存泄漏**。发生内存泄漏的代码会被多次执行到，每次被执行的时候都会导致一块内存泄漏。
- 2、**偶发性内存泄漏**。发生内存泄漏的代码只有在某些特定环境或操作过程下才会发生。常发性和偶发性是相对的。对于特定的环境，偶发性的也许就变成了常发性的。所以测试环境和测试方法对检测内存泄漏至关重要。
- 3、**一次性内存泄漏**。发生内存泄漏的代码只会被执行一次，或者由于算法上的缺陷，导致总会有一块仅且一块内存发生泄漏。比如，在类的构造函数中分配内存，在析构函数中却没有释放该内存，所以内存泄漏只会发生一次。

spring(8)
SpringMVC(8)
设计模式(5)

随笔档案

2018年6月(1)
2018年4月(1)
2018年1月(3)
2017年12月(2)
2017年11月(10)
2017年10月(3)
2017年9月(6)
2017年8月(9)
2017年7月(6)
2017年6月(12)
2017年5月(3)
2017年4月(3)
2017年3月(24)
2017年2月(14)
2017年1月(2)
更多

阅读排行榜

1. SpringMVC工作原理(454517)
2. Spring系列之Spring常用注解总结(250627)
3. mysql中find_in_set()函数的使用(193067)
4. Java集合之LinkedHashMap(171883)
5. springmvc请求参数获取的几种方法(155871)

评论排行榜

1. Spring系列之Spring常用注解总结(41)
2. SpringMVC工作原理(38)
3. 深入理解Java中的String(26)


4、隐式内存泄漏。程序在运行过程中不停的分配内存，但是直到结束的时候才释放内存。严格的说这里并没有发生内存泄漏，因为最终程序释放了所有申请的内存。但是对于一个服务器程序，需要运行几天，几周甚至几个月，不及时释放内存也可能导致最终耗尽系统的所有内存。所以，我们称这类内存泄漏为隐式内存泄漏。

从用户使用程序的角度来看，内存泄漏本身不会产生什么危害，作为一般的用户，根本感觉不到内存泄漏的存在。真正有危害的是内存泄漏的堆积，这会最终消耗尽系统所有的内存。从这个角度来说，一次性内存泄漏并没有什么危害，因为它不会堆积，而隐式内存泄漏危害性则非常大，因为较之于常发性和偶发性内存泄漏它更难被检测到。


下面总结几种常见的内存泄漏：

1、静态集合类引起的内存泄漏：

像HashMap、Vector等的使用最容易出现内存泄露，这些静态变量的生命周期和应用程序一致，他们所引用的所有的对象Object也不能被释放，从而造成内存泄漏，因为他们也将一直被Vector等引用着。



```
Vector<Object> v=new Vector<Object>(100);
for (int i = 1; i<100; i++)
{
    Object o = new Object();
    v.add(o);
    o = null;
}
```



在这个例子中，循环申请Object 对象，并将所申请的对象放入一个Vector 中，如果仅仅释放引用本身（o=null），那么Vector 仍然引用该对象，所以这个对象对GC 来说是不可回收的。因此，如果对象加入到Vector 后，还必须从Vector 中删除，最简单的方法就是将Vector对象设置为null。

2、修改HashSet中对象的参数值，且参数是计算哈希值的字段

当一个对象被存储到HashSet集合中以后，修改了这个对象中那些参与计算哈希值的字段后，这个对象的哈希值与最初存储在集合中的就不同了，这种情况下，用contains方法在集合中检索对象是找不到的，这将会导致无法从HashSet中删除当前对象，造成内存泄漏，举例如下：



```
public static void main(String[] args){

    Set<Person> set = new HashSet<Person>();
    Person p1 = new Person("张三","1",25);
    Person p2 = new Person("李四","2",26);
    Person p3 = new Person("王五","3",27);
    set.add(p1);
    set.add(p2);
    set.add(p3);
    System.out.println("总共有:"+set.size()+" 个元素!"); //结果: 总共有:3 个元素!
    p3.setAge(2); //修改p3的年龄,此时p3元素对应的hashCode值发生改变
    set.remove(p3); //此时remove不掉,造成内存泄漏
    set.add(p3); //重新添加,可以添加成功
    System.out.println("总共有:"+set.size()+" 个元素!"); //结果: 总共有:4 个元素!

    for (Person person : set){
        System.out.println(person);
    }
}
```



3、监听器

在java 编程中，我们都需要和监听器打交道，通常一个应用当中会用到很多监听器，我们会调用一个控件的诸如addXXXListener()等方法来增加监听器，但往往在释放对象的时候却没有记住去删除这些监听器，从而增加了内存泄漏的机会。

4、各种连接

比如数据库连接（dataSource.getConnection()），网络连接(socket)和io连接，除非其显式的调用了其close() 方法将其连接关闭，否则是不会自动被GC 回收的。对于ResultSet 和Statement 对象可以不进行显式回收，但Connection 一定要显式回收，因为Connection 在任何时候都无法自动回收，而Connection一旦回收，ResultSet 和Statement 对象就会立即为NULL。但是如果使用连接池，情况就不一样了，除了要显式地关闭连接，还必须显式地关闭ResultSet Statement 对象（关闭

4. JDK1.8 HashMap源码分析(15)
5. java集合框架综述(15)

推荐排行榜
1. Spring系列之Spring常用注解总结(135)
2. SpringMVC工作原理(129)
3. java集合框架综述(58)
4. 深入理解Java中的String(57)
5. Java垃圾回收（GC）机制详解(38)

最新评论
1. Re:深入理解Java的接口和抽象类
讲得太好了，感谢大佬！
--Yansoul
2. Re:Java8之方法引用
Arrays.sort(pArr, (a ,b) → Person.c ompareByAge(a, b)); 为什么可以这样 写？实现接口不需要重写里边的compa re方法吗？
--初晨~
3. Re:Java多线程之ReentrantLock与Con dition
详细
--天蝎输出全靠吼
4. Re:Spring系列之beanFactory与Applic ationContext
nice
--小孟随笔
5. Re:java 内部类详解
默默收藏，白嫖怪留
--zttt帅哥

其中一个，另外一个也会关闭），否则就会造成大量的Statement 对象无法释放，从而引起内存泄
漏。这种情况下一般都会在try里面去连接，在finally里面释放连接。

5、单例模式

如果单例对象持有外部对象的引用，那么这个外部对象将不能被jvm正常回收，导致内存泄露。

不正确使用单例模式是引起内存泄露的一个常见问题，单例对象在被初始化后将在JVM的整个生
命周期中存在（以静态变量的方式），如果单例对象持有外部对象的引用，那么这个外部对象将不能
被jvm正常回收，导致内存泄露，考虑下面的例子：

```
class A{
    public A () {
        B.getInstance().setA(this);
    }
    ....
}
//B类采用单例模式
class B{
    private A a;
    private static B instance=new B();
    public B () {}

    public static B getInstance(){
        return instance;
    }

    public void setA(A a){
        this.a=a;
    }
    //getter...
}
```

显然B采用singleton模式，它持有一个A对象的引用，而这个A类的对象将不能被回收。想象下如果A
是个比较复杂的对象或者集合类型会发生什么情况。

避免内存泄漏的几点建议：

- 1、尽早释放无用对象的引用。
- 2、避免在循环中创建对象。
- 3、使用字符串处理时避免使用String，应使用StringBuffer。
- 4、尽量少使用静态变量，因为静态变量存放在永久代，基本不参与垃圾回收。

分类: [Java虚拟机](#)

好文要顶 关注我 收藏该文



平凡希
关注 - 1
粉丝 - 1865

+加关注

70

« 上一篇: [java 内部类详解](#)
» 下一篇: [mysql索引原理](#)

posted @ 2017-08-13 21:18 平凡希 阅读(6876) 评论(1) 编辑 收藏 举报

刷新评论 刷新页面 返回顶部

登录后才能查看或发表评论，立即 [登录](#) 或者 [逛逛](#) 博客园首页

- 【推荐】发布 vscode 插件 Cnblogs Client For VSCode 预览版
- 【推荐】华为开发者专区，与开发者一起构建万物互联的智能世界
- 【推荐】参与华为 HarmonyOS 开发者创新大赛，一起创造无限可能

编辑推荐：

- [Three.js 实现虎年春节3D创意页面](#)
- [关于代码质量退化的思考](#)
- [asp.net core 中优雅的进行响应包装](#)
- [全链路压测\(4\)：全链路压测的价值是什么？](#)
- [神奇的滤镜！巧妙实现内凹的平滑圆角](#)

最新新闻：

- [2021年全球收入超10亿美元的8款手游：《王者荣耀》吸金超28亿美元位列第二](#)
 - [22 岁 ICPC 世界冠军加入华为](#)
 - [参与开源，让程序员都成为发明家而不是“码农”](#)
 - [为了让你转发视频，又一个社交媒体巨头开始向抖音取经](#)
 - [一加 10 Pro 首发测评：第二代哈苏影像挺强，但它还点了新技能树？](#)
- » [更多新闻...](#)