sun.misc.Cleaner

原创

lolichan 于 2017-04-28 09:39:47 发布 3392 收藏 1
分类专栏： java基础 文章标签： java 开发工具 移动开发
版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/lolichan/article/details/84883783
版权


java基础 专栏收录该内容
93 篇文章 1 订阅
订阅专栏

java NIO包是通过sun.misc.Cleaner和PhantomReference来实现堆外内存的自动释放的。现在来学习下Cleaner的使用

sun.misc.Cleaner是JDK内部提供的用来释放堆外内存的API

```
package com.cq.myproject.cleaner;

import java.lang.reflect.Field;

import sun.misc.Unsafe;

public class FreeMemoryTask implements Runnable {

 private long address;

 public FreeMemoryTask(long address)
 {
  this.address = address;
 }

 public void run() {
  System.out.println("running freeMemoryTask");
  if(address == 0)
  {
   System.out.println("already released");
  }
  else {
   getUnsafeInstance().freeMemory(address);
   System.out.println("released");
  }
 }

 public static Unsafe getUnsafeInstance() {
  try {
   Field theUnsafeInstance = Unsafe.class.getDeclaredField("theUnsafe");
   theUnsafeInstance.setAccessible(true);
   return (Unsafe) theUnsafeInstance.get(Unsafe.class);
  } catch (Exception e) {
   return null;
  }
 }
}
```

这个实现Runnable接口的类，功能就是释放堆外内存。这是我们必须要做的是，JVM没办法帮我们做（这里的Unsafe对象要手动import）

```
package com.cq.myproject.cleaner;
import sun.misc.Cleaner;
public class ObjectInHeapUseCleaner {
 private static long address = 0;

 public ObjectInHeapUseCleaner(){
  address = FreeMemoryTask.getUnsafeInstance().allocateMemory(2*1024*1024);
 }

 public static void main(String[] args) {
  while(true)
  {
   System.gc();
   ObjectInHeapUseCleaner heap = new ObjectInHeapUseCleaner();
   Cleaner.create(heap,new FreeMemoryTask(address));
  }
 }
}
```

运行这段代码，程序正常运行不会出现OOM（看任务管理器内存使用率不动，cpu使用率跳高）

Cleaner.create()需要两个参数：需要监控的内存对象和程序释放资源的回调。当JVM进行GC时，如果发现我们监控的对象，不存在强引用（Cleaner对象对他的引用是幽灵引用），就会调用第二个参数的run()方法逻辑，执行完run()方法时（此时已经释放了堆外内存），JVM会自动释放堆内存中我们监控的对象