## jconsole

### 通过Java/JMX得到full GC次数？ - Script Ahead, Code Behind - ITeye技术网站

http://rednaxelafx.iteye.com/blog/790015

今天有个同事问如何能通过[JMX](http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement/)获取到某个Java进程的full GC次数：

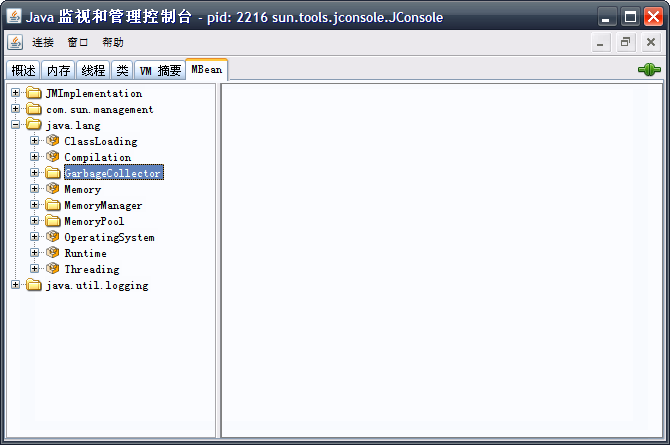
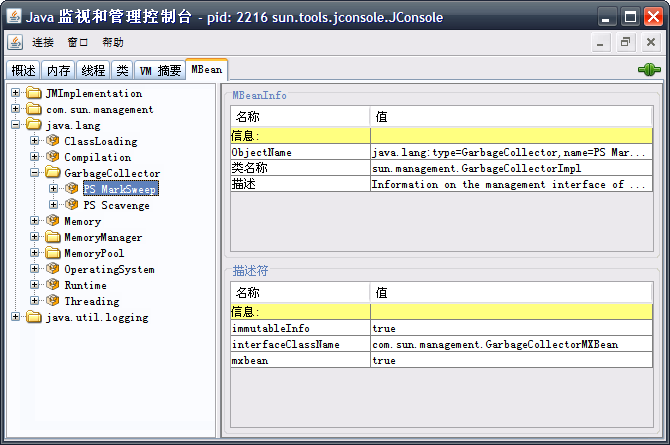
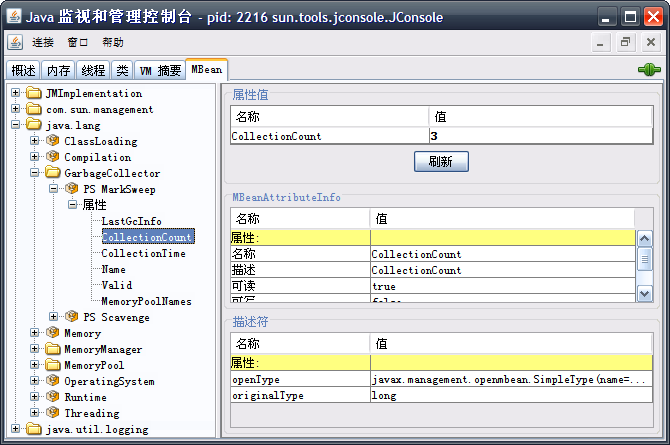
**引用**

hi,问个问题，怎们在java中获取到full gc的次数呢？   
我现在用jmx的那个得到了gc次数，不过不能细化出来full gc的次数

**Java代码  [收藏代码](javascript:void())**

1. **for** (**final** GarbageCollectorMXBean garbageCollector
2. : ManagementFactory.getGarbageCollectorMXBeans()) {
3. gcCounts += garbageCollector.getCollectionCount();
4. }

你比如我现在是这样拿次数的

我回答说因为full GC概念只有在分代式GC的上下文中才存在，而JVM并不强制要求GC使用分代式实现，所以JMX提供的标准[MXBean](http://download-llnw.oracle.com/javase/6/docs/technotes/guides/management/mxbeans.html) API里不提供“full GC次数”这样的方法也正常。   
既然“full GC”本来就是非常平台相关的概念，那就hack一点，用平台相关的代码来解决问题好了。这些GC的MXBean都是有名字的，而主流的JVM的GC名字相对稳定，非要通过JMX得到full GC次数的话，用名字来判断一下就好了。   
  
举个例子来看看。通过JDK 6自带的[JConsole](http://download-llnw.oracle.com/javase/6/docs/technotes/guides/management/jconsole.html)工具来查看相关的MXBean的话，可以看到，   
  
GC的MXBean在这个位置：   
   
  
这个例子是用server模式启动JConsole的，使用的是ParallelScavenge GC，它的年老代对应的收集器在这里：   
   
  
该收集器的总收集次数在此，这也就是full GC的次数：   
   
  
于是只要知道我们用的JVM提供的GC MXBean的名字与分代的关系，就可以知道full GC的次数了。   
Java代码写起来冗长，这帖就不用Java来写例子了，反正API是一样的，意思能表达清楚就OK。   
用一个[Groovy](http://groovy.codehaus.org/)脚本简单演示一下适用于Oracle (Sun) HotSpot与Oracle (BEA) JRockit的GC统计程序：

**Groovy代码  [收藏代码](javascript:void())**

1. import java.lang.management.ManagementFactory
3. printGCStats = {
4. def youngGenCollectorNames = [
5. // Oracle (Sun) HotSpot
6. // -XX:+UseSerialGC
7. 'Copy',
8. // -XX:+UseParNewGC
9. 'ParNew',
10. // -XX:+UseParallelGC
11. 'PS Scavenge',
13. // Oracle (BEA) JRockit
14. // -XgcPrio:pausetime
15. 'Garbage collection optimized for short pausetimes Young Collector',
16. // -XgcPrio:throughput
17. 'Garbage collection optimized for throughput Young Collector',
18. // -XgcPrio:deterministic
19. 'Garbage collection optimized for deterministic pausetimes Young Collector'
20. ]
22. def oldGenCollectorNames = [
23. // Oracle (Sun) HotSpot
24. // -XX:+UseSerialGC
25. 'MarkSweepCompact',
26. // -XX:+UseParallelGC and (-XX:+UseParallelOldGC or -XX:+UseParallelOldGCCompacting)
27. 'PS MarkSweep',
28. // -XX:+UseConcMarkSweepGC
29. 'ConcurrentMarkSweep',
31. // Oracle (BEA) JRockit
32. // -XgcPrio:pausetime
33. 'Garbage collection optimized for short pausetimes Old Collector',
34. // -XgcPrio:throughput
35. 'Garbage collection optimized for throughput Old Collector',
36. // -XgcPrio:deterministic
37. 'Garbage collection optimized for deterministic pausetimes Old Collector'
38. ]
40. R: {
41. ManagementFactory.garbageCollectorMXBeans.each {
42. def name  = it.name
43. def count = it.collectionCount
44. def gcType;
45. switch (name) {
46. case youngGenCollectorNames:
47. gcType = 'Minor Collection'
48. break
49. case oldGenCollectorNames:
50. gcType = 'Major Collection'
51. break
52. default:
53. gcType = 'Unknown Collection Type'
54. break
55. }
56. println "$count <- $gcType: $name"
57. }
58. }
59. }
61. printGCStats()

执行可以看到类似这样的输出：

**Command prompt代码  [收藏代码](javascript:void())**

1. 5 <- Minor Collection: Copy
2. 0 <- Major Collection: MarkSweepCompact

↑这是用client模式的HotSpot执行得到的；

**Command prompt代码  [收藏代码](javascript:void())**

1. 0 <- Minor Collection: Garbage collection optimized for throughput Young Collector
2. 0 <- Major Collection: Garbage collection optimized for throughput Old Collector

↑这是用JRockit R28在32位Windows上的默认模式得到的。   
  
通过上述方法，要包装起来方便以后使用的话也很简单，例如下面Groovy程序：

**Groovy代码  [收藏代码](javascript:void())**

1. import java.lang.management.ManagementFactory
3. class GCStats {
4. static final List<String> YoungGenCollectorNames = [
5. // Oracle (Sun) HotSpot
6. // -XX:+UseSerialGC
7. 'Copy',
8. // -XX:+UseParNewGC
9. 'ParNew',
10. // -XX:+UseParallelGC
11. 'PS Scavenge',
13. // Oracle (BEA) JRockit
14. // -XgcPrio:pausetime
15. 'Garbage collection optimized for short pausetimes Young Collector',
16. // -XgcPrio:throughput
17. 'Garbage collection optimized for throughput Young Collector',
18. // -XgcPrio:deterministic
19. 'Garbage collection optimized for deterministic pausetimes Young Collector'
20. ]
22. static final List<String> OldGenCollectorNames = [
23. // Oracle (Sun) HotSpot
24. // -XX:+UseSerialGC
25. 'MarkSweepCompact',
26. // -XX:+UseParallelGC and (-XX:+UseParallelOldGC or -XX:+UseParallelOldGCCompacting)
27. 'PS MarkSweep',
28. // -XX:+UseConcMarkSweepGC
29. 'ConcurrentMarkSweep',
31. // Oracle (BEA) JRockit
32. // -XgcPrio:pausetime
33. 'Garbage collection optimized for short pausetimes Old Collector',
34. // -XgcPrio:throughput
35. 'Garbage collection optimized for throughput Old Collector',
36. // -XgcPrio:deterministic
37. 'Garbage collection optimized for deterministic pausetimes Old Collector'
38. ]
40. static int getYoungGCCount() {
41. ManagementFactory.garbageCollectorMXBeans.inject(0) { youngGCCount, gc ->
42. if (YoungGenCollectorNames.contains(gc.name))
43. youngGCCount + gc.collectionCount
44. else
45. youngGCCount
46. }
47. }
49. static int getFullGCCount() {
50. ManagementFactory.garbageCollectorMXBeans.inject(0) { fullGCCount, gc ->
51. if (OldGenCollectorNames.contains(gc.name))
52. fullGCCount + gc.collectionCount
53. else
54. fullGCCount
55. }
56. }
57. }

用的时候：

**Groovysh代码  [收藏代码](javascript:void())**

1. D:\>\sdk\groovy-1.7.2\bin\groovysh
2. Groovy Shell (1.7.2, JVM: 1.6.0\_20)
3. Type 'help' or '\h' for help.
4. --------------------------------------------------
5. groovy:000> GCStats.fullGCCount
6. ===> 0
7. groovy:000> System.gc()
8. ===> null
9. groovy:000> GCStats.fullGCCount
10. ===> 1
11. groovy:000> System.gc()
12. ===> null
13. groovy:000> System.gc()
14. ===> null
15. groovy:000> GCStats.fullGCCount
16. ===> 3
17. groovy:000> GCStats.youngGCCount
18. ===> 9
19. groovy:000> GCStats.youngGCCount
20. ===> 9
21. groovy:000> GCStats.youngGCCount
22. ===> 9
23. groovy:000> System.gc()
24. ===> null
25. groovy:000> GCStats.youngGCCount
26. ===> 9
27. groovy:000> GCStats.fullGCCount
28. ===> 4
29. groovy:000> quit

这是在Sun JDK 6 update 20上跑的。顺带一提，如果这是跑在JRockit上的话，那full GC的次数就不会增加——因为JRockit里System.gc()默认是触发young GC的；请不要因为Sun HotSpot的默认行为而认为System.gc()总是会触发full GC的。   
  
Poonam Bajaj以前也写过一篇blog提到HotSpot VM里的GC MBean的名字的：   
[Collector names for GarbageCollectorMXBean MXBean](https://blogs.oracle.com/poonam/entry/collector_names_for_garbagecollectormxbean_mxbean)   
  
关于JMX的MXBean的使用，也可以参考下面两篇文档：   
[Groovy and JMX](http://docs.codehaus.org/display/GROOVY/Groovy+and+JMX)   
[Monitoring the JVM Heap with JRuby](http://www.engineyard.com/blog/2010/monitoring-the-jvm-heap-with-jruby/)