Android性能专项测试之Heap Viewer工具

> 参考文章:Heap Viewer
> Android 内存监测工具 DDMS –> Heap
> 使用DDMS中的内存监测工具Heap来优化内存

Heap Viewer能做什么？

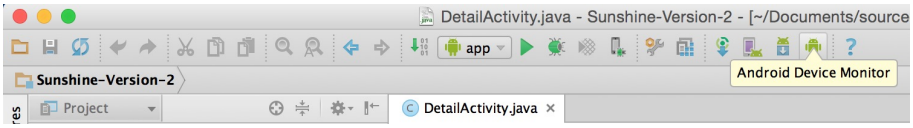
- 实时查看App分配的内存大小和空闲内存大小
- 发现Memory Leaks

Heap Viewer使用条件

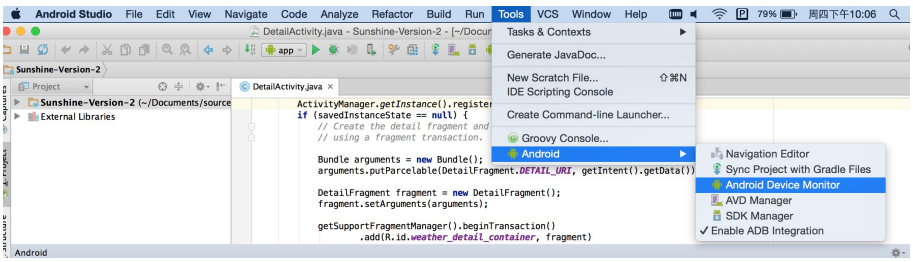
- 5.0以上的系统，包括5.0
- 开发者选项可用

Heap Viewer启动

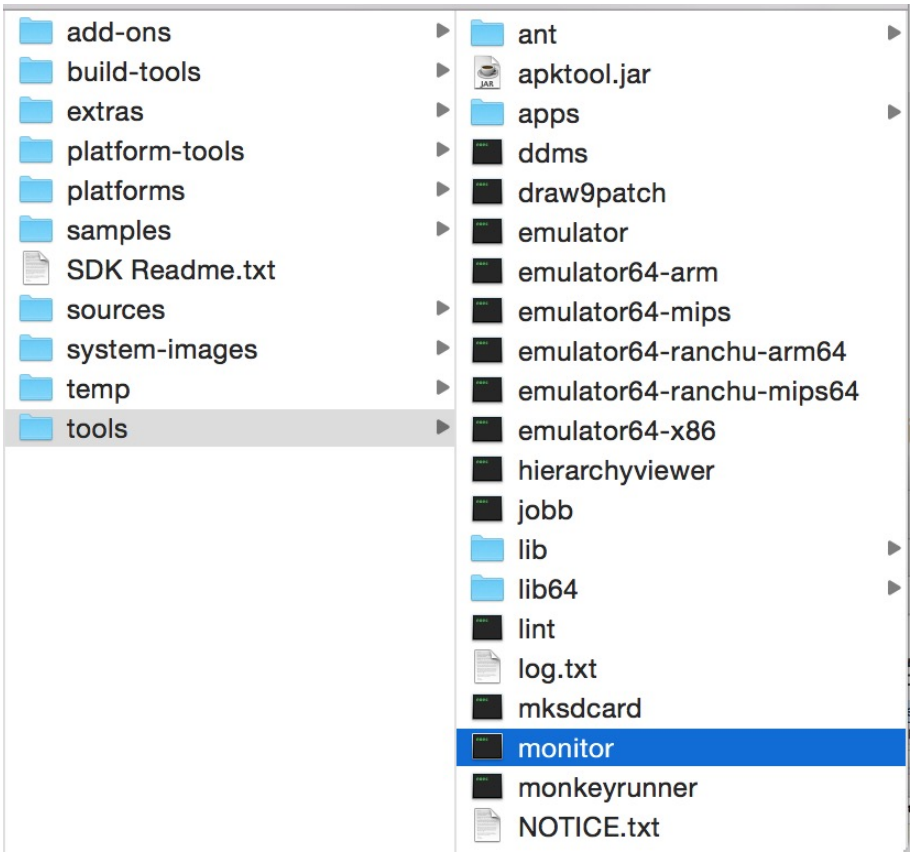
可以直接在Android studio工具栏中直接点击小机器人启动:



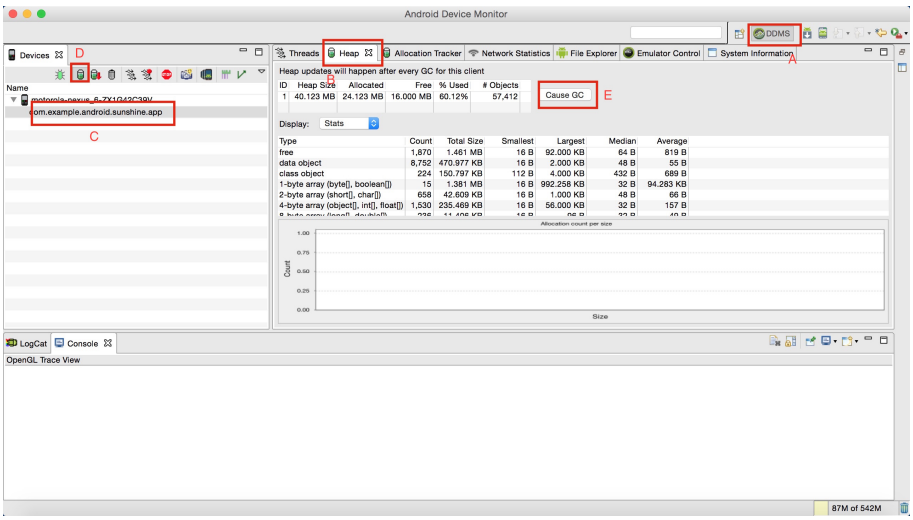
还可以在Android studio的菜单栏中Tools也可以：



如果你不用Android studio，可以在SDK下的tools下的monitor程序打开:



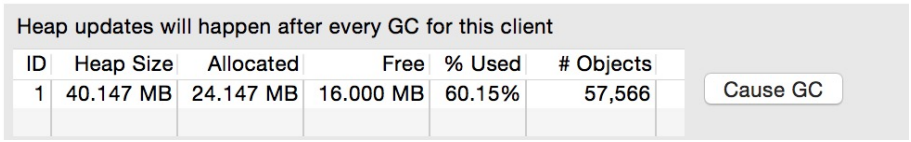
Heap Viewer面板



按上图的标记顺序按下，我们就能看到内存的具体数据，右边面板中数值会在每次GC时发生改变，包括App自动触发或者你来手动触发。

ok，现在来解释下面板中的名词

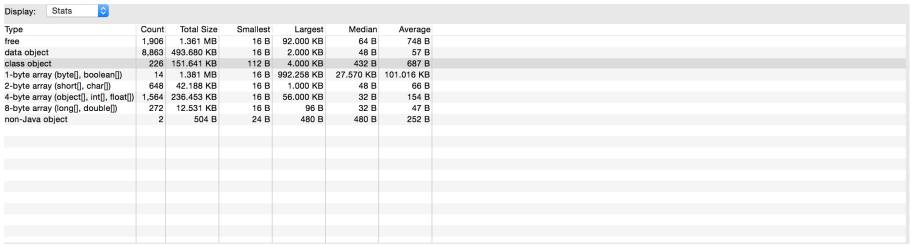
总览



列名	意义
Heap Size	堆栈分配给App的内存大小
Allocated	已分配使用的内存大小

Free	空闲的内存大小
%Used	Allocated/Heap Size,使用率
Objects	对象数量

详情

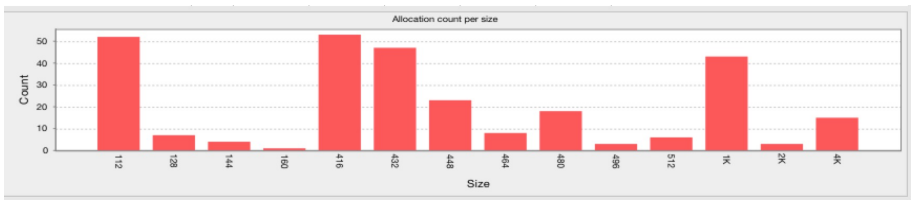


类型	意义
free	空闲的对象
data object	数据对象,类类型对象，最主要的观察对象
class object	类类型的引用对象
1-byte array(byte[],boolean[])	一个字节的数组对象
2-byte array(short[],char[])	两个字节的数组对象
4-byte array(long[],double[])	4个字节的数组对象
non-Java object	非Java对象

下面是每一个对象都有的列名含义：

列名	意义
Count	数量
Total Size	总共占用的内存大小
Smallest	将对象占用内存的大小从小往大排，排在第一个的对象占用内存大小
Largest	将对象占用内存的大小从小往大排，排在最后一个的对象占用的内存大小
Median	将对象占用内存的大小从小往大排，拍在中间的对象占用的内存大小
Average	平均值

当我们点击某一行时，可以看到如下的柱状图：



横坐标是对象的内存大小，这些值随着不同对象是不同的，纵坐标是在某个内存大小上的对象的数量

Heap Viewer的使用

我们说Heap Viewer适合发现内存泄漏的问题，那你知道何为内存泄漏么？

内存泄漏

英文名：Memory Leaks

标准解释：无用的单纯，但是还是没GC ROOT引用的内存

通俗解释：该死不死的内存

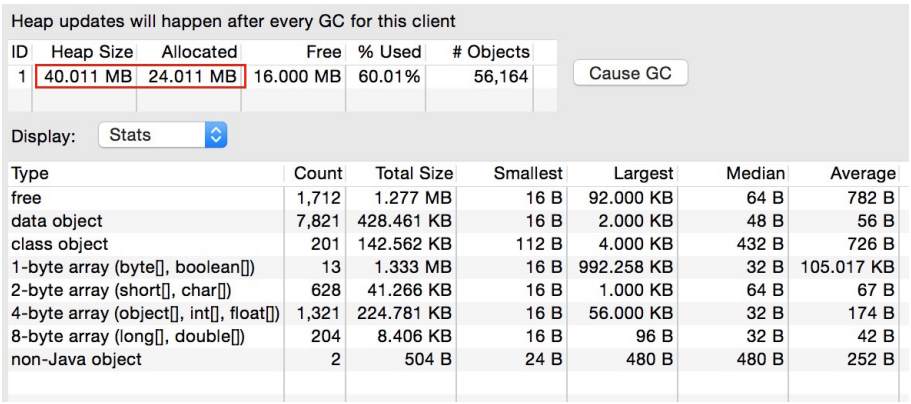
检测

那么如何检测呢？Heap Viewer中的数值会自动在每次发生GC时会自动更新，那么我们是等着他自己GC么？小弟不才，刚开始我就是这么一直等啊等，由于GC的时机是系统把握的，所以很不好把握，既然我们是来看内存泄漏，那么我们在需要检测内存泄漏的用例执行过后，手动GC下，然后观察 `data object` 一栏的 `total size` (也可以观察Heap Size/Allocated内存的情况)，看看内存是不是会回到一个稳定值，多次操作后，只要内存是稳定在某个值，那么说明没有内存溢出的，如果发现内存在每次GC后，都在增长，不管是慢增长还是快速增长，都说明有内存泄漏的可能性。

实例

先来看3个图：

1.刚打开首页,手动GC一下:



Heap updates will happen after every GC for this client

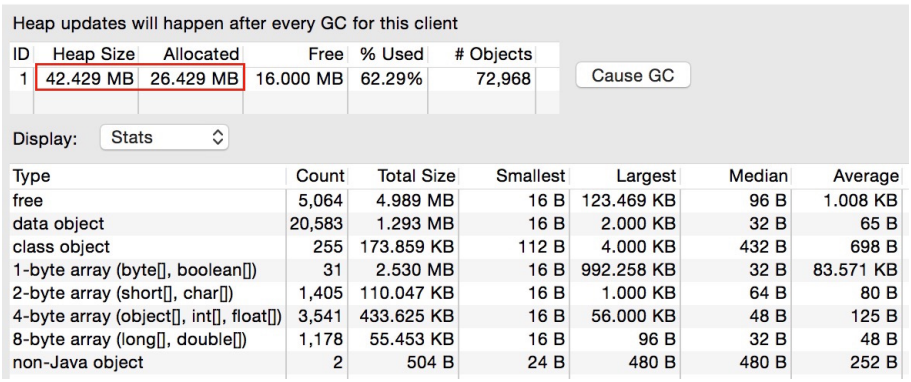
ID	Heap Size	Allocated	Free	% Used	# Objects
1	40.011 MB	24.011 MB	16.000 MB	60.01%	56,164

Cause GC

Display: Stats

Type	Count	Total Size	Smallest	Largest	Median	Average
free	1,712	1.277 MB	16 B	92.000 KB	64 B	782 B
data object	7,821	428.461 KB	16 B	2.000 KB	48 B	56 B
class object	201	142.562 KB	112 B	4.000 KB	432 B	726 B
1-byte array (byte[], boolean[])	13	1.333 MB	16 B	992.258 KB	32 B	105.017 KB
2-byte array (short[], char[])	628	41.266 KB	16 B	1.000 KB	64 B	67 B
4-byte array (object[], int[], float[])	1,321	224.781 KB	16 B	56.000 KB	32 B	174 B
8-byte array (long[], double[])	204	8.406 KB	16 B	96 B	32 B	42 B
non-Java object	2	504 B	24 B	480 B	480 B	252 B

2.首页到详情页10遍，最后回到首页，手动GC一下,直到数值不再变化：



Heap updates will happen after every GC for this client

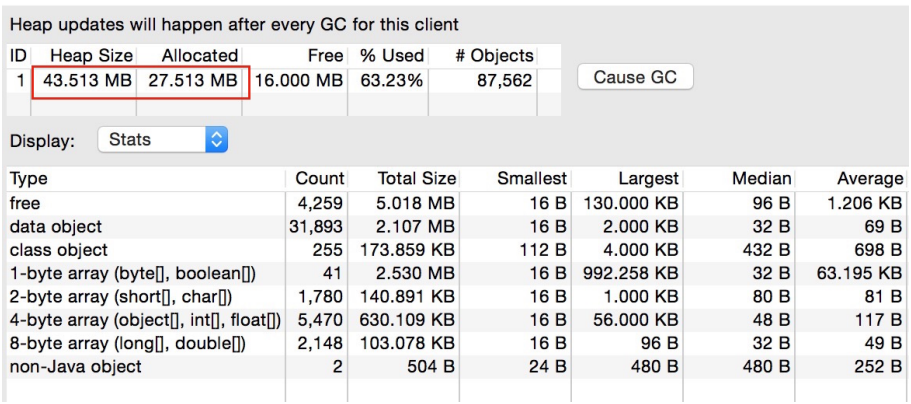
ID	Heap Size	Allocated	Free	% Used	# Objects
1	42.429 MB	26.429 MB	16.000 MB	62.29%	72,968

Cause GC

Display: Stats

Type	Count	Total Size	Smallest	Largest	Median	Average
free	5,064	4.989 MB	16 B	123.469 KB	96 B	1.008 KB
data object	20,583	1.293 MB	16 B	2.000 KB	32 B	65 B
class object	255	173.859 KB	112 B	4.000 KB	432 B	698 B
1-byte array (byte[], boolean[])	31	2.530 MB	16 B	992.258 KB	32 B	83.571 KB
2-byte array (short[], char[])	1,405	110.047 KB	16 B	1.000 KB	64 B	80 B
4-byte array (object[], int[], float[])	3,541	433.625 KB	16 B	56.000 KB	48 B	125 B
8-byte array (long[], double[])	1,178	55.453 KB	16 B	96 B	32 B	48 B
non-Java object	2	504 B	24 B	480 B	480 B	252 B

3.首页到详情页10遍，最后回到首页，手动GC一下：



Heap updates will happen after every GC for this client

ID	Heap Size	Allocated	Free	% Used	# Objects
1	43.513 MB	27.513 MB	16.000 MB	63.23%	87,562

Cause GC

Display: Stats

Type	Count	Total Size	Smallest	Largest	Median	Average
free	4,259	5.018 MB	16 B	130.000 KB	96 B	1.206 KB
data object	31,893	2.107 MB	16 B	2.000 KB	32 B	69 B
class object	255	173.859 KB	112 B	4.000 KB	432 B	698 B
1-byte array (byte[], boolean[])	41	2.530 MB	16 B	992.258 KB	32 B	63.195 KB
2-byte array (short[], char[])	1,780	140.891 KB	16 B	1.000 KB	80 B	81 B
4-byte array (object[], int[], float[])	5,470	630.109 KB	16 B	56.000 KB	48 B	117 B
8-byte array (long[], double[])	2,148	103.078 KB	16 B	96 B	32 B	49 B
non-Java object	2	504 B	24 B	480 B	480 B	252 B

从 `data object` 一栏看到该类型的数值会在不断增长，可能发生了内存泄漏，而我们也可以从上面三个图的标红部分来看，Allocated分别增加

了 2.418M 和 1.084M ，而且你继续这么操作下去，内存依然是增长的趋势

补充

Heap Viewer不光可以用来检测是否有内存泄漏，对于内存抖动，我们也可以用该工具检测，因为内存抖动的时候，会频繁发生GC，这个时候我们只需要开启Heap Viewer,观察数据的变化，如果发生内存抖动，会观察到数据在段时间内频繁更新。