Android性能专项测试之Allocation Tracker(Device Monitor)

2015-09-26 12:25 1387人阅读 评论(2) 收藏 举报

分类： Android性能（13）

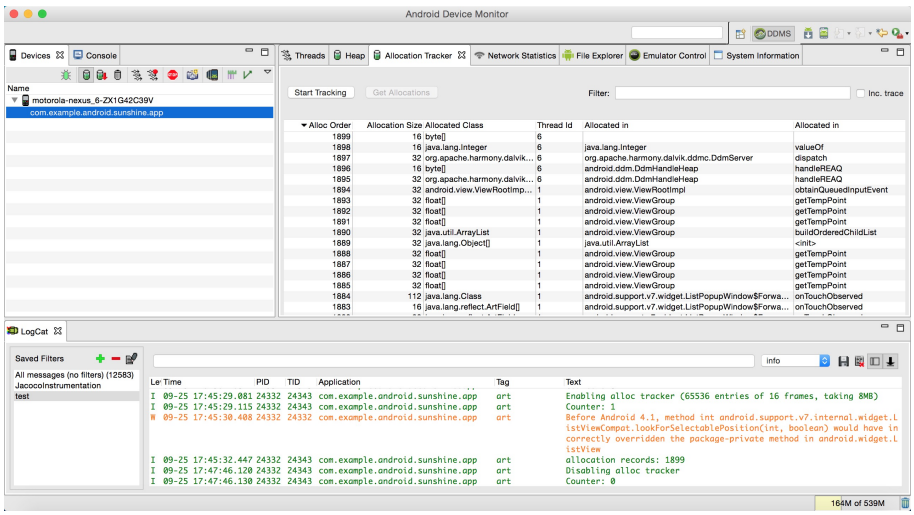
Allocation Tracker 能做什么？

追踪内存分配信息，按顺序排列，这样我们就能清晰看出来某一个操作的内存是如何一步一步分配出来的。比如在有内存抖动的可疑点，我们可以通过查看其内存分配轨迹来看短时间内有多少相同或相似的对象被创建，进一步找出发生问题的代码。

Allocation Tracker使用条件

- Root手机
- 开发者选项可用

Allocation Tracker面板



各名称的含义如下:

名称	意义
Alloc Order	分配序列
Allocation Size	分配的大小
Allocated Class	被分配的对象
Thread Id	线程id号
Allocated in	在哪个类分配的
第二个Allocated in	在哪个方法分配的

Allocation Tracker操作

1.首先进入你要追踪的界面

2.点击 `Start Tracking` 按钮，开始跟踪内存分配轨迹

3.操作你的界面，尽量时间短点

4.点击 `Get Allocations` 按钮，抓去内存分配轨迹信息，显示在右边的面板中，默认以内存大小排序，你可以以分配顺序排序或者仍以列排序。

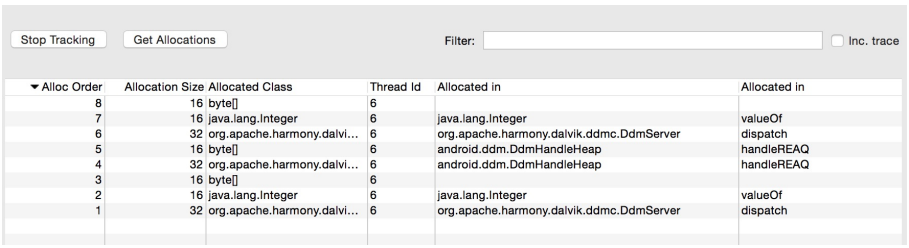
5.logcat中会显示出这次的轨迹共抓到内存分配轨迹记录数，可以简单的理解分配了多少次内存，这个数值和Alloc order的最大值是相等的

6.如果你不想看那么多乱七八糟的，你可以使用Filter来过滤，输入包名就可以了。

实例

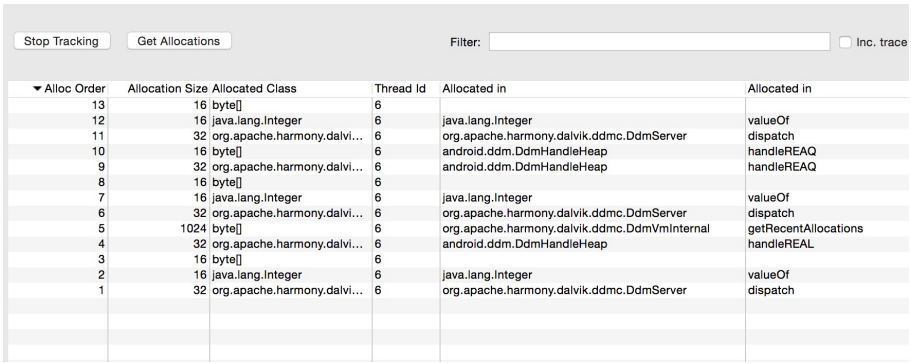
无任何操作时内存轨迹

打开首页，点击 `Stop tracking` ，然后点击 `Get Allocations` ，会看到下面1～8的内存分配序列:


Stop Tracking Get Allocations Filter: Inc. trace

Alloc Order	Allocation Size	Allocated Class	Thread Id	Allocated in	Allocated in
8	16	byte[]	6		
7	16	java.lang.Integer	6	java.lang.Integer	valueOf
6	32	org.apache.harmony.dalvi...	6	org.apache.harmony.dalvik.ddmc.DdmServer	dispatch
5	16	byte[]	6	android.ddm.DdmHandleHeap	handleREAQ
4	32	org.apache.harmony.dalvi...	6	android.ddm.DdmHandleHeap	handleREAQ
3	16	byte[]	6		
2	16	java.lang.Integer	6	java.lang.Integer	valueOf
1	32	org.apache.harmony.dalvi...	6	org.apache.harmony.dalvik.ddmc.DdmServer	dispatch

再按一次 `Get Allocations` 会出现如下状态:

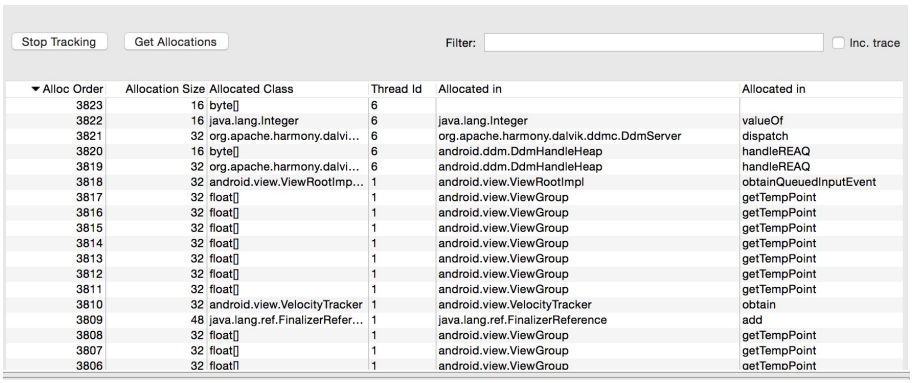

Stop Tracking Get Allocations Filter: Inc. trace

Alloc Order	Allocation Size	Allocated Class	Thread Id	Allocated in	Allocated in
13	16	byte[]	6		
12	16	java.lang.Integer	6	java.lang.Integer	valueOf
11	32	org.apache.harmony.dalvi...	6	org.apache.harmony.dalvik.ddmc.DdmServer	dispatch
10	16	byte[]	6	android.ddm.DdmHandleHeap	handleREAQ
9	32	org.apache.harmony.dalvi...	6	android.ddm.DdmHandleHeap	handleREAQ
8	16	byte[]	6		
7	16	java.lang.Integer	6	java.lang.Integer	valueOf
6	32	org.apache.harmony.dalvi...	6	org.apache.harmony.dalvik.ddmc.DdmServer	dispatch
5	1024	byte[]	6	org.apache.harmony.dalvik.ddmc.DdmVmInternal	getRecentAllocations
4	32	org.apache.harmony.dalvi...	6	android.ddm.DdmHandleHeap	handleREAL
3	16	byte[]	6		
2	16	java.lang.Integer	6	java.lang.Integer	valueOf
1	32	org.apache.harmony.dalvi...	6	org.apache.harmony.dalvik.ddmc.DdmServer	dispatch

这些信息估计都是DDMS和app交互产生的内存，我们可以忽略

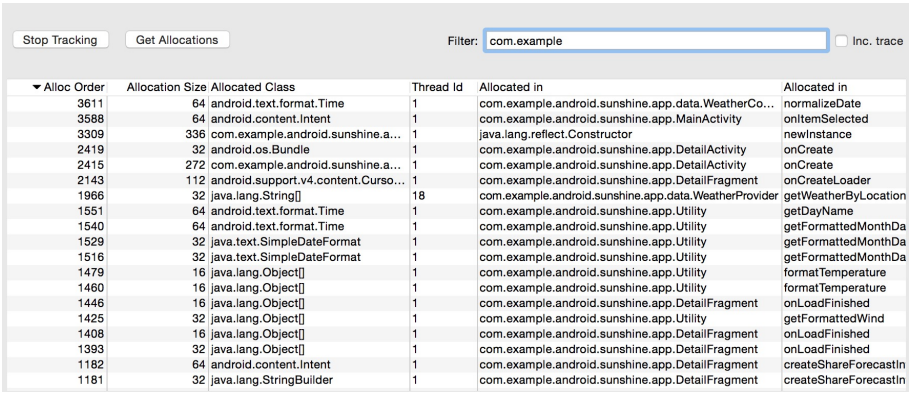
正常操作的内存轨迹

如果这个时候我们想单独获取某次操作的内存轨迹，首先一定要记得 `Stop Tracking` 再 `Start Tracking` 一下，让追踪点初始化一下，这个时候我们从首页进入一个详情页,看一下我们的内存分配轨迹：


Stop Tracking Get Allocations Filter: Inc. trace

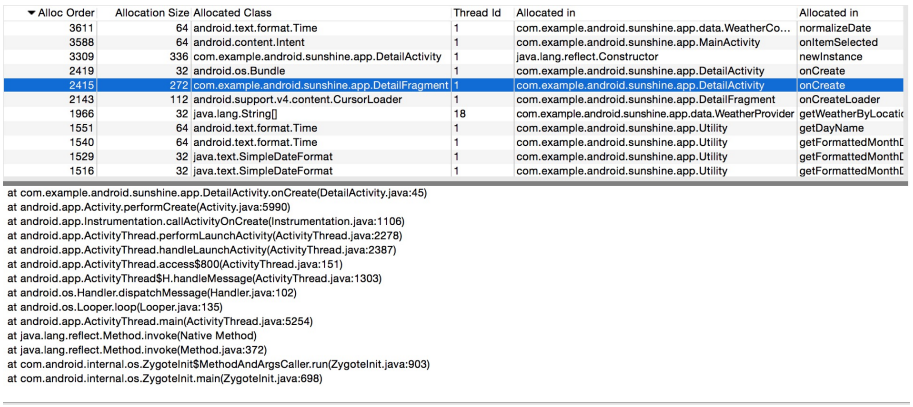
Alloc Order	Allocation Size	Allocated Class	Thread Id	Allocated in	Allocated in
3823	16	byte[]	6		
3822	16	java.lang.Integer	6	java.lang.Integer	valueOf
3821	32	org.apache.harmony.dalvi...	6	org.apache.harmony.dalvik.ddmc.DdmServer	dispatch
3820	16	byte[]	6	android.ddm.DdmHandleHeap	handleREAQ
3819	32	org.apache.harmony.dalvi...	6	android.ddm.DdmHandleHeap	handleREAQ
3818	32	android.view.ViewRootImp...	1	android.view.ViewRootImpl	obtainQueuedInputEvent
3817	32	float[]	1	android.view.ViewGroup	getTempPoint
3816	32	float[]	1	android.view.ViewGroup	getTempPoint
3815	32	float[]	1	android.view.ViewGroup	getTempPoint
3814	32	float[]	1	android.view.ViewGroup	getTempPoint
3813	32	float[]	1	android.view.ViewGroup	getTempPoint
3812	32	float[]	1	android.view.ViewGroup	getTempPoint
3811	32	float[]	1	android.view.ViewGroup	getTempPoint
3810	32	android.view.VelocityTracker	1	android.view.VelocityTracker	obtain
3809	48	java.lang.ref.FinalizerRefer...	1	java.lang.ref.FinalizerReference	add
3808	32	float[]	1	android.view.ViewGroup	getTempPoint
3807	32	float[]	1	android.view.ViewGroup	getTempPoint
3806	32	float[]	1	android.view.ViewGroup	getTempPoint

追踪到的内存分配3823次，看着是不是有点无从下手，没关系，用Filter过滤下:



Alloc Order	Allocation Size	Allocated Class	Thread Id	Allocated in	Allocated in
3611	64	android.text.format.Time	1	com.example.android.sunshine.app.data.WeatherCo…	normalizeDate
3588	64	android.content.Intent	1	com.example.android.sunshine.app.MainActivity	onItemSelected
3309	336	com.example.android.sunshine.a…	1	java.lang.reflect.Constructor	newInstance
2419	32	android.os.Bundle	1	com.example.android.sunshine.app.DetailActivity	onCreate
2415	272	com.example.android.sunshine.a…	1	com.example.android.sunshine.app.DetailActivity	onCreate
2143	112	android.support.v4.content.Curso…	1	com.example.android.sunshine.app.DetailFragment	onCreateLoader
1966	32	java.lang.String[]	18	com.example.android.sunshine.app.data.WeatherProvider	getWeatherByLocation
1551	64	android.text.format.Time	1	com.example.android.sunshine.app.Utility	getDayName
1540	64	android.text.format.Time	1	com.example.android.sunshine.app.Utility	getFormattedMonthDa
1529	32	java.text.SimpleDateFormat	1	com.example.android.sunshine.app.Utility	getFormattedMonthDa
1516	32	java.text.SimpleDateFormat	1	com.example.android.sunshine.app.Utility	getFormattedMonthDa
1479	16	java.lang.Object[]	1	com.example.android.sunshine.app.Utility	formatTemperature
1460	16	java.lang.Object[]	1	com.example.android.sunshine.app.Utility	formatTemperature
1446	16	java.lang.Object[]	1	com.example.android.sunshine.app.DetailFragment	onLoadFinished
1425	32	java.lang.Object[]	1	com.example.android.sunshine.app.Utility	getFormattedWind
1408	16	java.lang.Object[]	1	com.example.android.sunshine.app.DetailFragment	onLoadFinished
1393	32	java.lang.Object[]	1	com.example.android.sunshine.app.DetailFragment	onLoadFinished
1182	64	android.content.Intent	1	com.example.android.sunshine.app.DetailFragment	createShareForecastIn
1181	32	java.lang.StringBuilder	1	com.example.android.sunshine.app.DetailFragment	createShareForecastIn

过滤后，就剩下了跟我们App源码有关系的分配轨迹，我们随便选择一栏，可以看到其trace信息:



Alloc Order	Allocation Size	Allocated Class	Thread Id	Allocated in	Allocated in
3611	64	android.text.format.Time	1	com.example.android.sunshine.app.data.WeatherCo…	normalizeDate
3588	64	android.content.Intent	1	com.example.android.sunshine.app.MainActivity	onItemSelected
3309	336	com.example.android.sunshine.app.DetailActivity	1	java.lang.reflect.Constructor	newInstance
2419	32	android.os.Bundle	1	com.example.android.sunshine.app.DetailActivity	onCreate
2415	272	com.example.android.sunshine.app.DetailFragment	1	com.example.android.sunshine.app.DetailActivity	onCreate
2143	112	android.support.v4.content.CursorLoader	1	com.example.android.sunshine.app.DetailFragment	onCreateLoader
1966	32	java.lang.String[]	18	com.example.android.sunshine.app.data.WeatherProvider	getWeatherByLocati
1551	64	android.text.format.Time	1	com.example.android.sunshine.app.Utility	getDayName
1540	64	android.text.format.Time	1	com.example.android.sunshine.app.Utility	getFormattedMonthD
1529	32	java.text.SimpleDateFormat	1	com.example.android.sunshine.app.Utility	getFormattedMonthD
1516	32	java.text.SimpleDateFormat	1	com.example.android.sunshine.app.Utility	getFormattedMonthD

```
at com.example.android.sunshine.app.DetailActivity.onCreate(DetailActivity.java:45)
at android.app.Activity.performCreate(Activity.java:5990)
at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1106)
at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2278)
at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:2387)
at android.app.ActivityThread.access$800(ActivityThread.java:151)
at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1303)
at android.os.Handler.dispatchMessage(Handler.java:102)
at android.os.Looper.loop(Looper.java:135)
at android.app.ActivityThread.main(ActivityThread.java:5254)
at java.lang.reflect.Method.invoke(Native Method)
at java.lang.reflect.Method.invoke(Method.java:372)
at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:903)
at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:698)
```

上图中，我们可以看出来，在第2415次内存分配中，分配的是 `DetailFragment` 对象，占用内存272字节，处理线程Id为1，

在 `com.example.android.sunshine.app.DetailActivity` 的 `onCreate` 方法中分配的。从trace信息可以看出来该方法一步一步被调用的信息。

然后我们回源码中确认下，以下代码就是我们上面选择的内存分配的地方：

```
    private final String LOG_TAG = DetailActivity.class.getSimpleName();

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_detail);
            Log.d(LOG_TAG, "onCreate");
            ActivityManager.getInstance().registerActivity(this);
            if (savedInstanceState == null) {
                // Create the detail fragment and add it to the activity
                // using a fragment transaction.

                Bundle arguments = new Bundle();
                arguments.putParcelable(DetailFragment.DETAIL_URI, getIntent().getData());

                DetailFragment fragment = new DetailFragment();
                fragment.setArguments(arguments);

                getSupportFragmentManager().beginTransaction()
                        .add(R.id.weather_detail_container, fragment)
                        .commit();
            }
        }
```