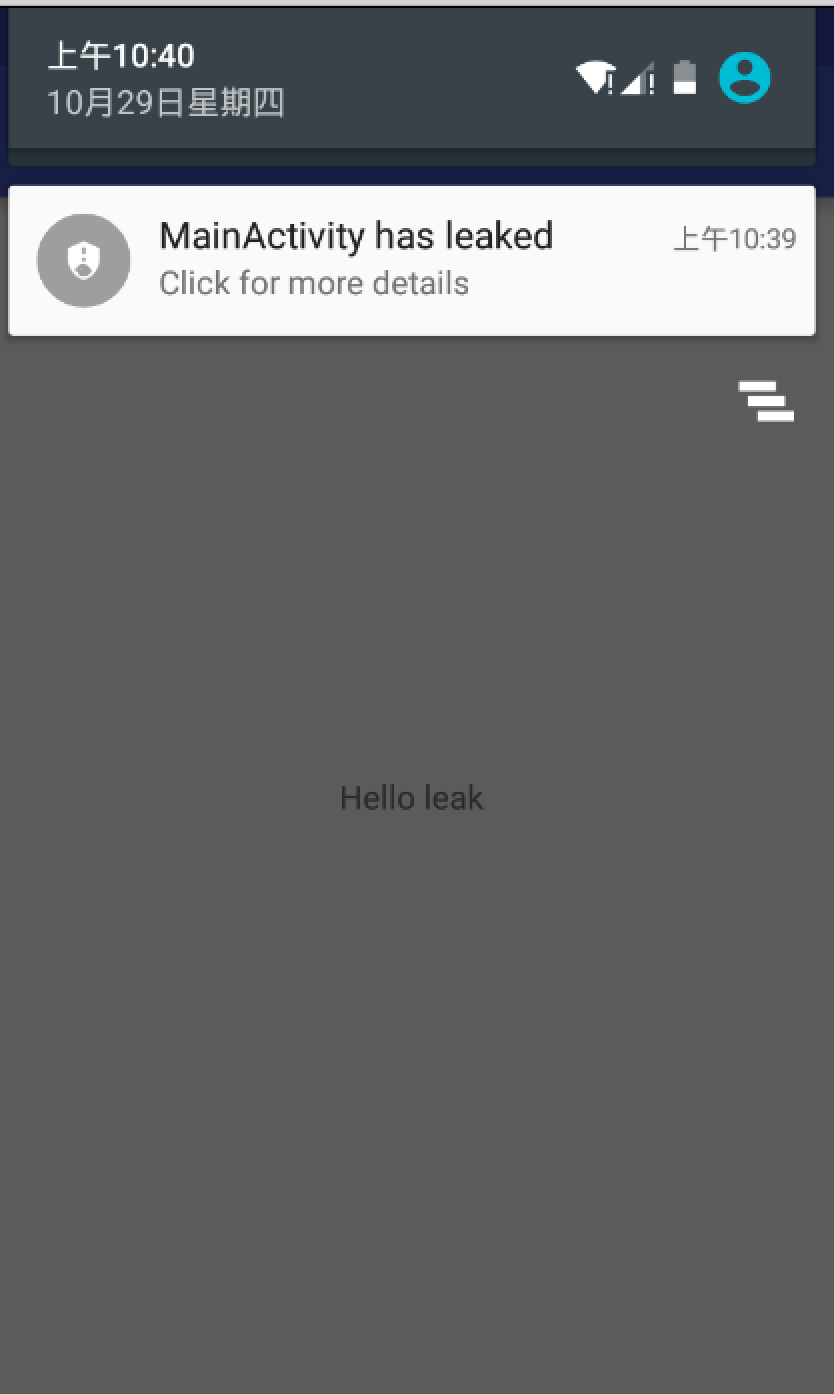
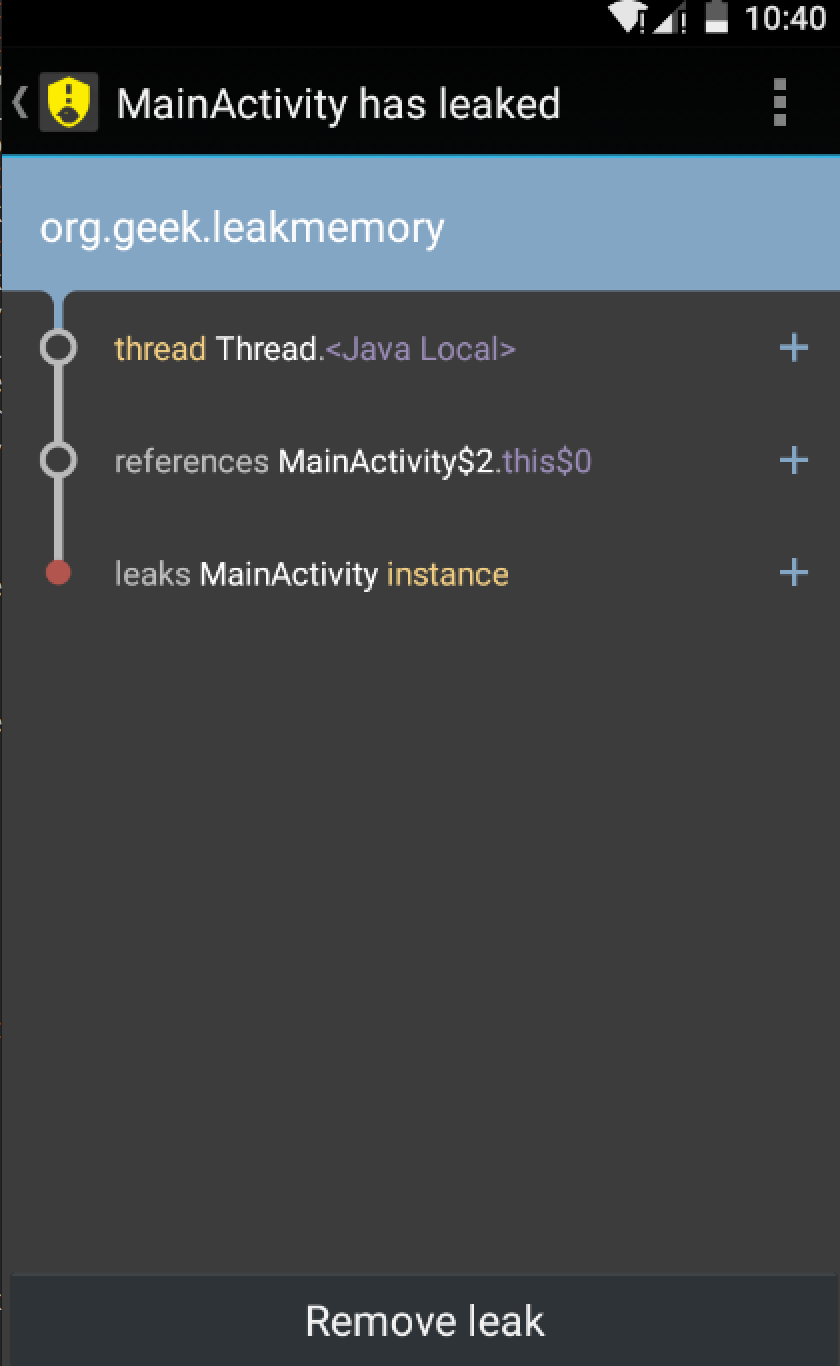
http://blog.csdn.net/watermusicyes/article/details/46333925

之前碰到的OOM问题，终于很直白的呈现在我的眼前：我尝试了MAT，但是发现不怎么会用。直到今天终于发现了这个新工具：

当我们的App中存在内存泄露时会在通知栏弹出通知：



当点击该通知时，会跳转到具体的页面，展示出Leak的引用路径，如下图所示：



LeakCanary 可以用更加直白的方式将内存泄露展现在我们的面前。

以下是我找到的学习资料，写的非常棒：   
1、[LeakCanary: 让内存泄露无所遁形](http://www.liaohuqiu.net/cn/posts/leak-canary/)   
2、[LeakCanary 中文使用说明](http://www.liaohuqiu.net/cn/posts/leak-canary-read-me/)

**AndroidStudio** (官方)上使用LeakCanary 请移步：   
<https://github.com/square/leakcanary>

**Eclipse** 上使用LeakCanary 请移步我的：   
<https://github.com/SOFTPOWER1991/LeakcanarySample-Eclipse>

[**Android**](http://lib.csdn.net/base/android)**studio** （自己弄的）上使用LeakCanary也可以看这个：

[leakcanarySample\_androidStudio](https://github.com/SOFTPOWER1991/leakcanarySample_androidStudio)

工程包括：

1. LeakCanary库代码
2. LeakCanaryDemo示例代码

使用步骤：

1. 将LeakCanary import 入自己的工程
2. 添加依赖：

compile project(':leakcanary')

1. 在Application中进行配置
2. public class ExampleApplication extends Application {
3. ......
4. //在自己的Application中添加如下代码
5. public static RefWatcher getRefWatcher(Context context) {
6. ExampleApplication application = (ExampleApplication) context
7. .getApplicationContext();
8. return application.refWatcher;
9. }
10. //在自己的Application中添加如下代码
11. private RefWatcher refWatcher;
12. @Override
13. public void onCreate() {
14. super.onCreate();
15. ......
16. //在自己的Application中添加如下代码
17. refWatcher = LeakCanary.install(this);
18. ......
19. }
20. .....
21. }
    * 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
    * 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
22. 在Activity中进行配置

public class MainActivity extends AppCompatActivity {

......

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

//在自己的应用初始Activity中加入如下两行代码

RefWatcher refWatcher = ExampleApplication.getRefWatcher(this);

refWatcher.watch(this);

textView = (TextView) findViewById(R.id.tv);

textView.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

startAsyncTask();

}

});

}

private void async() {

startAsyncTask();

}

private void startAsyncTask() {

// This async task is an anonymous class and therefore has a hidden reference to the outer

// class MainActivity. If the activity gets destroyed before the task finishes (e.g. rotation),

// the activity instance will leak.

new AsyncTask<Void, Void, Void>() {

@Override

protected Void doInBackground(Void... params) {

// Do some slow work in background

SystemClock.sleep(20000);

return null;

}

}.execute();

}

}

* 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
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 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
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43

1. 在AndroidMainfest.xml 中进行配置,添加如下代码

<service

android:name="com.squareup.leakcanary.internal.HeapAnalyzerService"

android:enabled="false"

android:process=":leakcanary" />

<service

android:name="com.squareup.leakcanary.DisplayLeakService"

android:enabled="false" />

<activity

android:name="com.squareup.leakcanary.internal.DisplayLeakActivity"

android:enabled="false"

android:icon="@drawable/\_\_leak\_canary\_icon"

android:label="@string/\_\_leak\_canary\_display\_activity\_label"

android:taskAffinity="com.squareup.leakcanary"

android:theme="@style/\_\_LeakCanary.Base" >

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>