[LeakCanary](http://blog.csdn.net/itfootball/article/details/49361401)

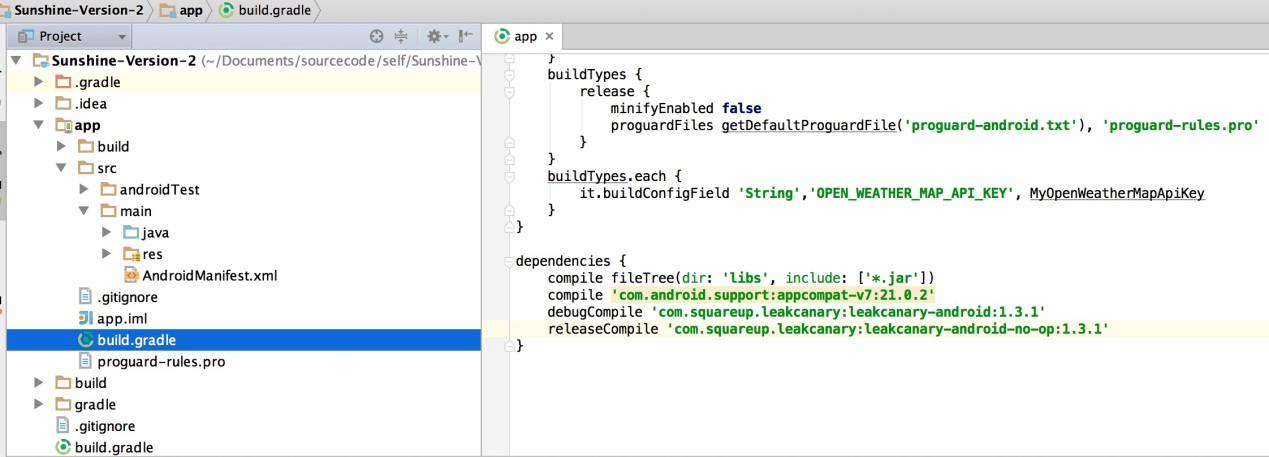
添加LeakCanary依赖包

https://github.com/square/leakcanary

在主模块app下的build.gradle下添加如下依赖:

debugCompile 'com.squareup.leakcanary:leakcanary-android:1.3.1'

releaseCompile 'com.squareup.leakcanary:leakcanary-android-no-op:1.3.1'

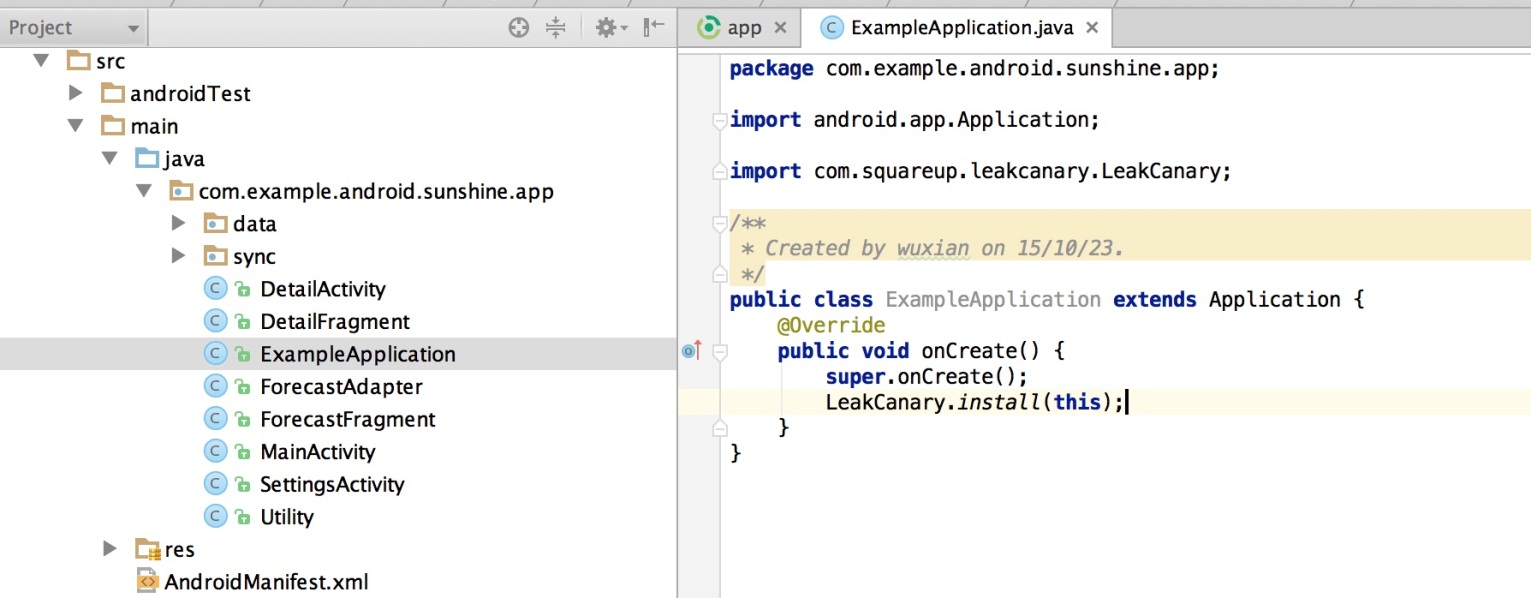


# 开启LeakCanary

## 添加Application子类

首先创建一个ExampleApplication，该类继承于Application,在该类的onCreate方法中添加如下代码开启LeakCanary监控:

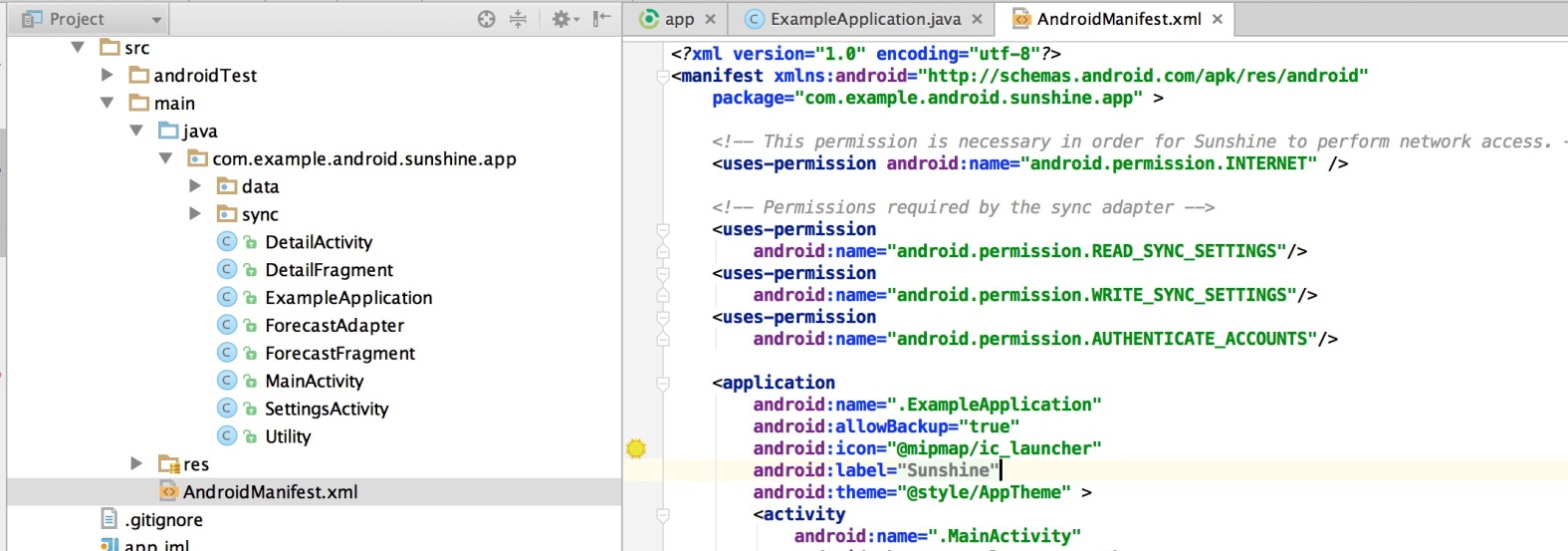
LeakCanary.install(this);



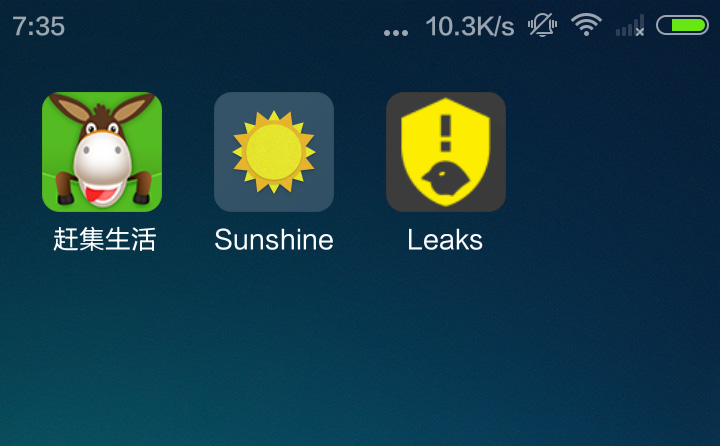
## 在配置文件中注册ExampleApplication

在AndroidManifest.xml中的application标签中添加如下信息:

android:name=".ExampleApplication"



这个时候安装应用到手机，会自动安装一个Leaks应用，如下图: 



# 制造一个内存泄漏的点

建立一个ActivityManager类，单例模式，里面有一个数组用来保存Activity：

package com.example.android.sunshine.app;

import android.app.Activity;

import android.util.SparseArray;

import android.view.animation.AccelerateInterpolator;

import java.util.List;

public class ActivityManager {

private SparseArray<Activity> container = new SparseArray<Activity>();

private int key = 0;

private static ActivityManager mInstance;

private ActivityManager(){}

public static ActivityManager getInstance(){

if(mInstance == null){

mInstance = new ActivityManager();

}

return mInstance;

}

public void addActivity(Activity activity){

container.put(key++,activity);

}

}

然后在DetailActivity中的onCreate方法中将当前activity添加到ActivityManager的数组中:

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_detail);

ActivityManager.getInstance().addActivity(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)

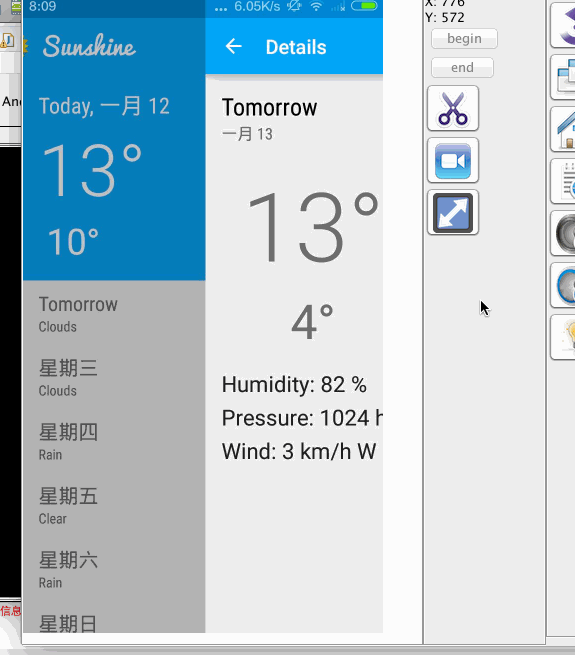
.commit();

}

}

我们从首页跳转到详情页的时候会进入DetailActivity的onCreate的方法，然后就将当前activity添加到了数组中，当返回时，我们没把他从数组中删除。再次进入的时候，会创建新的activity 并添加到数组中，但是之前的activity仍然被引用，无法释放，但是这个activity不会再被使用，这个时候就造成了内存泄漏。我们来看看LeakCanary是如何报出这个问题的。

# 演示



解析的过程有点耗时，所以需要等待一会才会在Leaks应用中，当我们点开某一个信息时，会看到详细的泄漏信息:   
