[Android activity的生命周期](http://blog.csdn.net/hpoi/article/details/4629065)

标签： [android](http://www.csdn.net/tag/android)[ui](http://www.csdn.net/tag/ui)[class](http://www.csdn.net/tag/class)[活动](http://www.csdn.net/tag/%e6%b4%bb%e5%8a%a8)[网络](http://www.csdn.net/tag/%e7%bd%91%e7%bb%9c)

2009-10-03 11:38 126853人阅读 [评论](http://blog.csdn.net/hpoi/article/details/4629065#comments)(26) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/hpoi/article/details/4629065#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

Java/Android/Ophone（6） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

 Android activity的生命周期

activity类处于android.app包中，继承体系如下：

1.java.lang.Object

2.android.content.Context

3.android.app.ApplicationContext

4.android.app.Activity

activity是单独的，用于处理用户操作。几乎所有的activity都要和用户打交道，所以activity类创建了一个窗口，开发人员可以通过setContentView(View)接口把UI放到activity创建的窗口上，当 activity指向全屏窗口时，也可以用其他方式实现：作为漂浮窗口（通过windowIsFloating的主题集合），或者嵌入到其他的 activity（使用ActivityGroup）。大部分的Activity子类都需要实现以下两个接口：

* onCreate(Bundle)接口是初始化activity的地方. 在这儿通常可以调用setContentView(int)设置在资源文件中定义的UI, 使用findViewById(int) 可以获得UI中定义的窗口.
* onPause()接口是使用者准备离开activity的地方，在这儿，任何的修改都应该被提交(通常用于ContentProvider保存数据).

为了能够使用Context.startActivity()，所有的activity类都必须在AndroidManifest.xml文件中定义有相关的“activity”项。

activity类是Android 应用生命周期的重要部分。

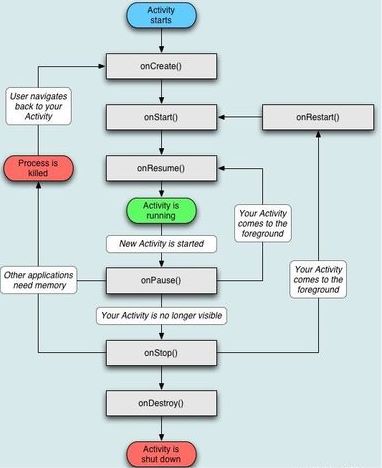
Activity生命周期

在系统中的Activity被一个Activity栈所管理。当一个新的Activity启动时，将被放置到栈顶，成为运行中的Activity，前一个Activity保留在栈中，不再放到前台，直到新的Activity退出为止。

Activity有四种本质区别的状态：

1. 在屏幕的前台（Activity栈顶），叫做活动状态或者运行状态（active or running）
2. 如果一个Activity失去焦点，但是依然可见（一个新的非全屏的Activity 或者一个透明的Activity 被放置在栈顶），叫做暂停状态（Paused）。一个暂停状态的Activity依然保持活力（保持所有的状态，成员信息，和窗口管理器保持连接），但是在系统内存极端低下的时候将被杀掉。
3. 如果一个Activity被另外的Activity完全覆盖掉，叫做停止状态（Stopped）。它依然保持所有状态和成员信息，但是它不再可见，所以它的窗口被隐藏，当系统内存需要被用在其他地方的时候，Stopped的Activity将被杀掉。
4. 如果一个Activity是Paused或者Stopped状态，系统可以将该Activity从内存中删除，Android系统采用两种方式进行删除，要么要求该Activity结束，要么直接杀掉它的进程。当该Activity再次显示给用户时，它必须重新开始和重置前面的状态。

下面的图显示了Activity的重要状态转换，矩形框表明Activity在状态转换之间的回调接口，开发人员可以重载实现以便执行相关代码，带有颜色的椭圆形表明Activity所处的状态。



在上图中，Activity有三个关键的循环：

1. 整个的生命周期，从onCreate(Bundle)开始到onDestroy()结束。Activity在onCreate()设置所有的“全局”状态，在onDestory()释放所有的资源。例如：某个Activity有一个在后台运行的线程，用于从网络下载数据，则该Activity可以在onCreate()中创建线程,在onDestory()中停止线程。
2. 可见的生命周期，从onStart()开始到onStop()结束。在这段时间，可以看到Activity在屏幕上，尽管有可能不在前台，不能和用户交互。在这两个接口之间，需要保持显示给用户的UI数据和资源等，例如：可以在onStart中注册一个IntentReceiver来监听数据变化导致UI的变动，当不再需要显示时候，可以在onStop()中注销它。onStart()，onStop()都可以被多次调用，因为Activity随时可以在可见和隐藏之间转换。
3. 前台的生命周期，从onResume()开始到onPause()结束。在这段时间里，该Activity处于所有 Activity的最前面，和用户进行交互。Activity可以经常性地在resumed和paused状态之间切换，例如：当设备准备休眠时，当一个 Activity处理结果被分发时，当一个新的Intent被分发时。所以在这些接口方法中的代码应该属于非常轻量级的。

Activity的整个生命周期都定义在下面的接口方法中，所有方法都可以被重载。所有的Activity都需要实现 onCreate(Bundle)去初始化设置，大部分Activity需要实现onPause()去提交更改过的数据，当前大部分的Activity也需要实现onFreeze()接口，以便恢复在onCreate(Bundle)里面设置的状态。

public class Activity extends ApplicationContext {

protected void onCreate(Bundle icicle);

protected void onStart();

protected void onRestart();

protected void onResume();

protected void onFreeze(Bundle outIcicle);

protected void onPause();

protected void onStop();

protected void onDestroy();

}