# AndroidActivity的onSaveInstanceState() 和 onRestoreInstanceState()方法

1调用时机：

* 1. 由系统销毁一个Activity时，onSaveInstanceState() 会被调用。

系统销毁的时机一般有：

**>>内存不足，系统资源回收**

**>>用户直接按home键**。   
**>>长按HOME键，选择运行其他的程序时。  
>>按下电源按键（关闭屏幕显示）时。**

**>>设备配置的变更**

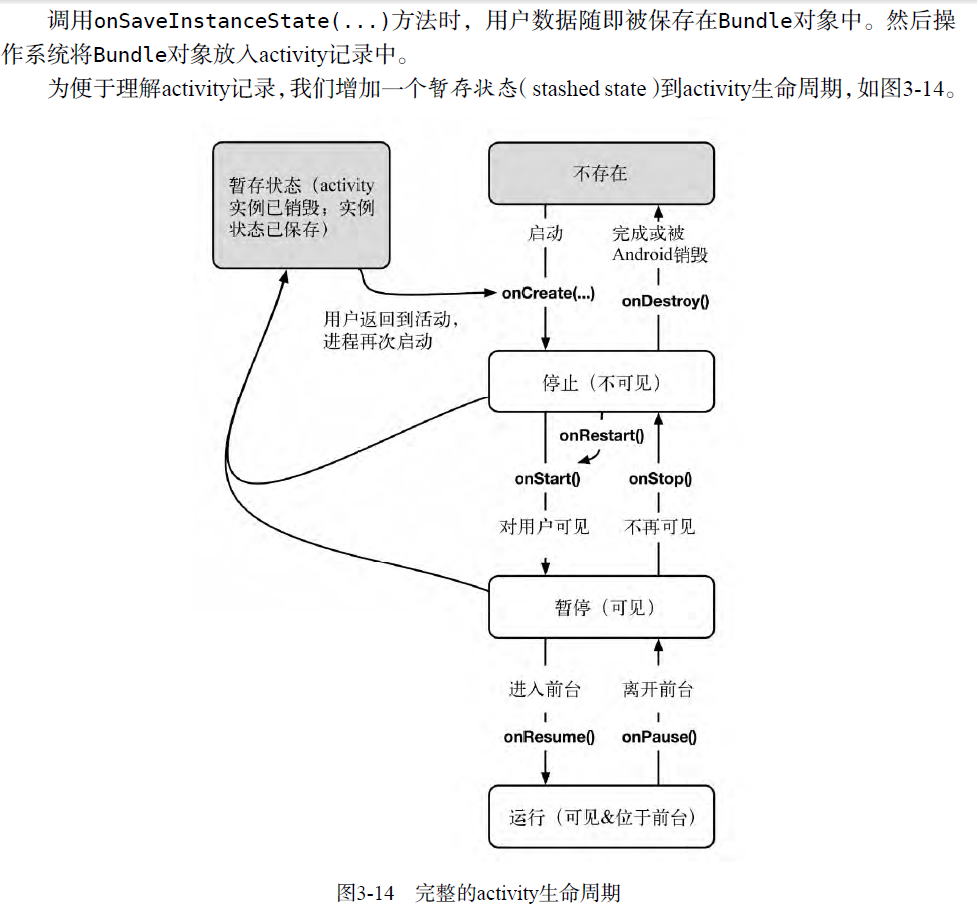
**>>从activity A中启动一个新的activity时。**

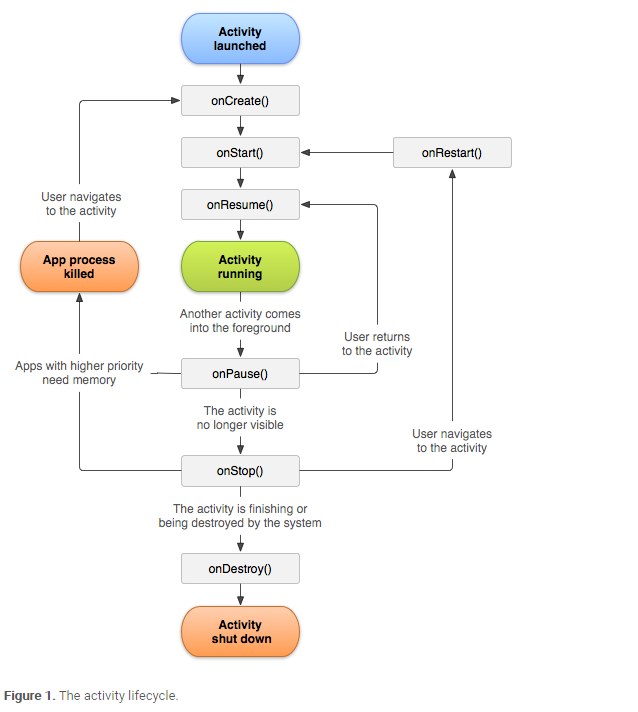
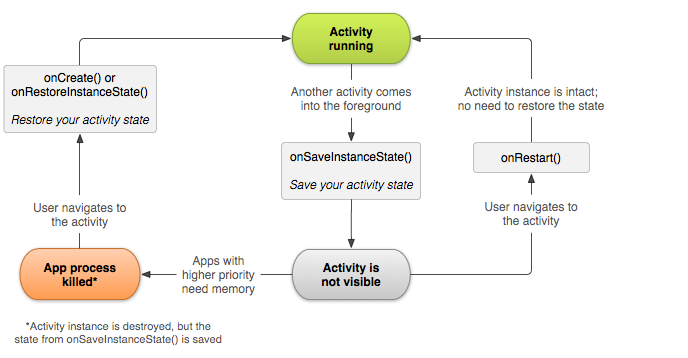
设备配置与备选资源

旋转设备会改变设备配置（device configuration）。设备配置是用来描述设备当前状态的一系

列特征。这些特征包括：**屏幕的方向、屏幕的密度、屏幕的尺寸、键盘类型、底座模式以及语言**，

等等。通常，为匹配不同的设备配置，应用会提供不同的备选资源。为适应不同分辨率的屏幕，向

项目里添加多套箭头图标就是这样一个使用案例。  
  


activity暂存后，Activity对象不再存在，但操作系统会将activity记录对象保存起来。这样，在需要恢复activity时，操作系统可以使用暂存的activity记录重新激活activity。

注意，activity进入暂存状态并不一定需要调用onDestroy()方法。不过，onPause()和onSave-

InstanceState(...)通常是我们需要调用的两个方法。常见的做法是，覆盖onSaveInstanceState(...)方法，将数据暂存到Bundle对象中，覆盖onPause()方法处理其他需要处理的事情。

有时，Android不仅会销毁activity，还会彻底停止当前应用的进程。不过，只有在用户离开当前应用时才会发生这种情况。即使这种情况真的发生了，暂存的activity记录依然被系统保留着，以便于用户返回应用时activity的快速恢复。

那么暂存的activity记录到底可以保留多久？前面说过，**用户按了后退键后，系统会彻底销毁当前的activity**。此时，暂存的activity记录同时被清除。此外，**系统重启或长时间不使用activity时，暂存的activity记录通常也会被清除**。

1.2 onSaveInstanceState()通常用于保存什么数据和状态？

通常onSaveInstanceState()只适合用于保存一些临时性的状态，

而onPause()适合用于数据的持久化保存。

* 1. 用户back键返回不会调用 onSaveInstanceState()
  2. Bundle中存储和恢复的数据类型:

**>>基本数据类型，   
>>实现Serializable接口的对象，**

**>>实现Parcelable的对象。   
>>实现CharSequence的对象。  
>>或者这些类型的ArrayList<>形式对象。  
>>或许Bundle类型或Bundle子类。**

注意，我们在Bundle中存储和恢复的数据类型只能是基本数据类型（primitive type）以及可以实现Serializable接口的对象。创建自己的定制类时，如需在onSaveInstanceState(...)方法中保存类对象，记得实现Serializable接口。

测试onSaveInstanceState(...)的实现是个好习惯，尤其在需要存储和恢复对象时。设备

旋转很容易测试，但测试低内存状态就困难多了。本章末尾会深入学习这部分内容，继而学习如何模拟Android为回收内存而销毁activity的场景。

1.5 **onSaveInstanceState()方法的默认实现。**

　　如果我们没有覆写onSaveInstanceState()方法, 此方法的默认实现会自动保存activity中的某些状态数据, 比如activity中各种UI控件的状态.。android应用框架中定义的几乎所有UI控件都恰当的实现了onSaveInstanceState()方法,因此当activity被摧毁和重建时, 这些UI控件会自动保存和恢复状态数据. 比如EditText控件会自动保存和恢复输入的数据,而CheckBox控件会自动保存和恢复选中状态.开发者只需要为这些控件指定一个唯一的ID(通过设置android:id属性即可), 剩余的事情就可以自动完成了.如果没有为控件指定ID, 则这个控件就不会进行自动的数据保存和恢复操作。

由上所述, 如果我们需要覆写onSaveInstanceState()方法, 一般会在第一行代码中调用该方法的默认实现:super.onSaveInstanceState(outState)。

1.6 是否需要重写onSaveInstanceState()方法

　　既然该方法的默认实现可以自动的保存UI控件的状态数据, 那什么时候需要覆写该方法呢?

　　如果需要保存额外的数据时, 就需要覆写onSaveInstanceState()方法。大家需要注意的是：onSaveInstanceState()方法只适合保存瞬态数据, 比如UI控件的状态, 成员变量的值等，而不应该用来保存持久化数据，持久化数据应该当用户离开当前的 activity时，在 onPause() 中保存（比如将数据保存到数据库或文件中）。说到这里，还要说一点的就是在onPause()中不适合用来保存比较费时的数据，所以这点要理解。

由于onSaveInstanceState()方法方法不一定会被调用, 因此不适合在该方法中保存持久化数据, 例如向数据库中插入记录等. 保存持久化数据的操作应该放在onPause()中。若是永久性值，则在onPause()中保存；若大量，则另开线程吧，别阻塞UI线程。