Android Activity的onSaveInstanceState() 和 onRestoreInstanceState()方法：

1. 基本作用：

　　Activity的 onSaveInstanceState() 和 onRestoreInstanceState()并不是生命周期方法，它们不同于 onCreate()、onPause()等生命周期方法，它们并不一定会被触发。当应用遇到意外情况（如：内存不足、用户直接按Home键）由系统销毁一个Activity时，onSaveInstanceState() 会被调用。但是当用户主动去销毁一个Activity时，例如在应用中按返回键，onSaveInstanceState()就不会被调用。因为在这种情况下，用户的行为决定了不需要保存Activity的状态。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。

　　在activity被杀掉之前调用保存每个实例的状态,以保证该状态可以在onCreate(Bundle)或者onRestoreInstanceState(Bundle) (传入的Bundle参数是由onSaveInstanceState封装好的)中恢复。这个方法在一个activity被杀死前调用，当该activity在将来某个时刻回来时可以恢复其先前状态。

　　例如，如果activity B启用后位于activity A的前端，在某个时刻activity A因为系统回收资源的问题要被杀掉，A通过onSaveInstanceState将有机会保存其用户界面状态，使得将来用户返回到activity A时能通过onCreate(Bundle)或者onRestoreInstanceState(Bundle)恢复界面的状态。

　　关于onSaveInstanceState ()，是在函数里面保存一些View有用的数据到一个Parcelable对象并返回。在Activity的onSaveInstanceState(Bundle outState)中调用View的onSaveInstanceState ()，返回Parcelable对象，

　　接着用Bundle的putParcelable方法保存在Bundle  savedInstanceState中。

　　当系统调用Activity的的onRestoreInstanceState(Bundle savedInstanceState)时， 同过Bundle的getParcelable方法得到Parcelable对象，然后把该Parcelable对象传给View的onRestoreInstanceState (Parcelable state)。在的View的onRestoreInstanceState中从Parcelable读取保存的数据以便View使用。

　　这就是onSaveInstanceState() 和 onRestoreInstanceState() 两个函数的基本作用和用法。

2. onSaveInstanceState() 什么时候调用

　　先看Application Fundamentals上的一段话：

　　Android calls onSaveInstanceState() before the activitybecomes vulnerable to being destroyed by the system, but does not bothercalling it when the instance is actually being destroyed by a user action (suchas pressing the BACK key).

　　从这句话可以知道，当某个activity变得"容易"被系统销毁时，该activity的onSaveInstanceState()就会被执行，除非该activity是被用户主动销毁的，例如当用户按BACK键的时候。

　　注意上面的双引号，何为"容易"？意思就是说该activity还没有被销毁，而仅仅是一种可能性。这种可能性有哪些？通过重写一个activity的所有生命周期的onXXX方法，包括onSaveInstanceState()和onRestoreInstanceState() 方法，我们可以清楚地知道当某个activity（假定为activity A）显示在当前task的最上层时，其onSaveInstanceState()方法会在什么时候被执行，有这么几种情况：

　　(1)、当用户按下HOME键时。

　　这是显而易见的，系统不知道你按下HOME后要运行多少其他的程序，自然也不知道activity A是否会被销毁，因此系统会调用onSaveInstanceState()，让用户有机会保存某些非永久性的数据。以下几种情况的分析都遵循该原则

　　(2)、长按HOME键，选择运行其他的程序时。

　　(3)、按下电源按键（关闭屏幕显示）时。

　　(4)、从activity A中启动一个新的activity时。

　　(5)、屏幕方向切换时，例如从竖屏切换到横屏时。

　　在屏幕切换之前，系统会销毁activity A，在屏幕切换之后系统又会自动地创建activity A，所以onSaveInstanceState()一定会被执行，且也一定会执行onRestoreInstanceState()。

　　总而言之，onSaveInstanceState()的调用遵循一个重要原则，即当系统存在“未经你许可”时销毁了我们的activity的可能时，则onSaveInstanceState()会被系统调用，这是系统的责任，因为它必须要提供一个机会让你保存你的数据（当然你不保存那就随便你了）。如果调用，调用将发生在onPause()或onStop()方法之前。（虽然测试时发现多数在onPause()前）

　　3. onRestoreInstanceState()什么时候调用

　　onRestoreInstanceState()被调用的前提是，activity A“确实”被系统销毁了，而如果仅仅是停留在有这种可能性的情况下，则该方法不会被调用，例如，当正在显示activity A的时候，用户按下HOME键回到主界面，然后用户紧接着又返回到activity A，这种情况下activity A一般不会因为内存的原因被系统销毁，故activity A的onRestoreInstanceState方法不会被执行 此也说明上二者，大多数情况下不成对被使用。

　　onRestoreInstanceState()在onStart() 和 onPostCreate(Bundle)之间调用。

4. onSaveInstanceState()方法的默认实

　　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)。

5. 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。

6. 引发activity销毁和重建的其它情况

　　除了系统处于内存不足的原因会摧毁activity之外, 某些系统设置的改变也会导致activity的摧毁和重建. 例如改变屏幕方向(见上例), 改变设备语言设定, 键盘弹出等。