Activity的生命周期

Activity从创建到销毁的过程，叫Activity的生命周期

Activity何时创建：打开图标后，任务栈创建，Activity创建，压入栈底部

Activity何时销毁：按Back后，Activity被销毁，当任务栈中无Activity时，任务栈销毁Activity=任务栈

Activity应用通常：启动一个应用，linux系统会为这个应用分配唯一一个标识符，一个进程，一个UI主线程及任务栈

进程：一个进程至少会有一个主线程，进程有独立的空间和数据，其它子线程都是由主线程创建分配的

线程： 主线程 即UI线程，属于进程 线程无独立的空间和数据，共享进程的空间和数据

onCreate() 当Activity第一次创建的时候调用【重点】

onDestory() 当Activity销毁的时候调用【重点】

onStart() 当Activity变成可见的时候调用

onStop() 当Activity不可见的时候调用

onResume() 当Activity可以和用户交互的时候调用，即获取焦点

onPause() 当Activity不可以和用户交互的时候调用，即失获取焦点

onRestart() 当Activity停止了，但是没有销毁，从停止到启动时调用

生命周期方法	调用时间
onCreate()	当Activity第一次创建的时候调用
onDestory()	当Activity销毁的时候调用
onStart()	当Activity变成可见的时候调用
onStop()	当Activity不可见的时候调用
onResume()	当Activity可以和用户交互的时候调用
onPause()	当Activity不可和用户交互的时候调用
onRestart()	当Activity停止了，但是没有销毁，从停止到启动时调用

完整生命周期：从onCreate(Bundle)开始到onDestory()结束。Activity在onCreate()设置的所有“全局“状态，onDestory()释放所有的资源。

可见生命周期：从onStart()开始到onStop()结束。在这段时间，可以看到Activity在屏幕上，尽管可能不在前台，不能和用户交互。在两个接口之间，需要保持显示给用户UI的数据和资源等。

前台生命周期：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的整个生命周期都定义在对应的接口方法中，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。

横竖屏切换时Activity生命周期的变化

默认情况下，先销毁原来的Activity，再创建新的Activity，

从而让横竖屏切换时不会销毁原来的Acticvity

在清单文件中配置固定的横竖屏

```
<activity android:screenOrientation="portrait"> </activity>
```

android:screenOrientation=”portrait” 永远竖屏

android:screenOrientation=”landspace” 永远横屏

若不指定该属性的值，默认情况下该值为“sensor”，即根据传感器来自动设置屏幕的方向。

设置系统的环境，使其不再敏感横竖屏的切换。让系统不再敏感横竖屏切换可以在activity中设

android:configChanges=”orientation|keyboardHidden|screenSize”

参数	含义
configChanges=”orientation”	屏幕方向改变：不让屏幕在切换时重新创建activity。
sreensize	

	屏幕大小
keyboardHidden	软键盘，如果切换屏幕，软键盘会去判断屏幕大小是否合适显示软键盘，在判断过程中会重启activity

任务栈 ------>界面Activity，每一个应用每次打开都会有一个唯一的任务栈用来保存应用的每一个界面

栈作为一种数据结构，是一种只能在一端进行插入和删除操作的特殊线性表。它按照先进后出的原则存储数据，先进入的数据被压入栈底，最后的数据在栈顶，需要读数据的时候从栈顶开始弹出数据。栈具有记忆作用，对栈的插入与删除操作中，不需要改变栈底指针。

任务栈：任务栈是用来提升用户体验而设计的，记录打开界面和关闭界面的信息。每开启一个应用程序，android操作系统就会给这个应用程序分配一个任务栈。

一般情况下每个应用程序一开启就会创建一个任务栈，任务栈的id是自动增长的。

最小化的时候，应用程序实际上是后台运行，任务栈是保留的，当Activity退出了实际上就是任务栈清空了。

通过调用getTaskId()方法，可以获取当前任务栈的Id。

```
//任务栈中装的是：Activity
//何时创建任务栈:点击应用程序图标时，创建任务栈，且将第一个界面装入任务栈
//何时销毁任务栈:当任务栈中无界面时，销毁任务栈

//每个界面都可以获取界面所在任务栈的编号，该编号唯一
int taskId = this.getTaskId();
System.out.println("taskId=" + taskId);
```

队列 ------>消息机制Message

队列也是一种特殊的线性表，特殊之处在于它只允许在表的前端（front）进行删除操作，而在表的后端（rear）进行插入操作，和栈一样，队列是一种操作受限制的线性表。通过对栈和队列的比较，我们知道：栈是先进后出，后进先出；队列是先进先出，后进后出。

Activity的启动模式

a，说出**standard**模式的特点

总创建新的**Activity**

b，说出**singletop**模式的特点

栈顶相同**Activity**重用

c，说出**singletask**模式的特点

整个栈相同**Activity**重用，且强行让其上的**Activity**出栈，自己位于栈定位置

d，说出**singleInstance**模式的特点

整个手机中，所有栈中，相同**Activity**只有一个，且位于新的栈中

- singletop 单一顶部模式

如果任务栈的栈顶存在这个要开启的activity，不会重新创建新的activity，而是复用已存在的activity。保证栈顶如果存在，则不会重复创建，但如果不在栈顶，那么还是会创建新的实例。

应用场景：书签界面

- singletask 单一任务模式

是一个比较严格的模式，在当前任务栈里面只能有一个实例存在，当开启activity的时候，就去检查在任务栈里面是否有实例已经存在，如果有实例存在就复用这个已经存在的activity，并且把这个activity上面的所有的别的activity都清空，复用这个已经存在的activity。

应用场景：浏览器界面，播放器界面

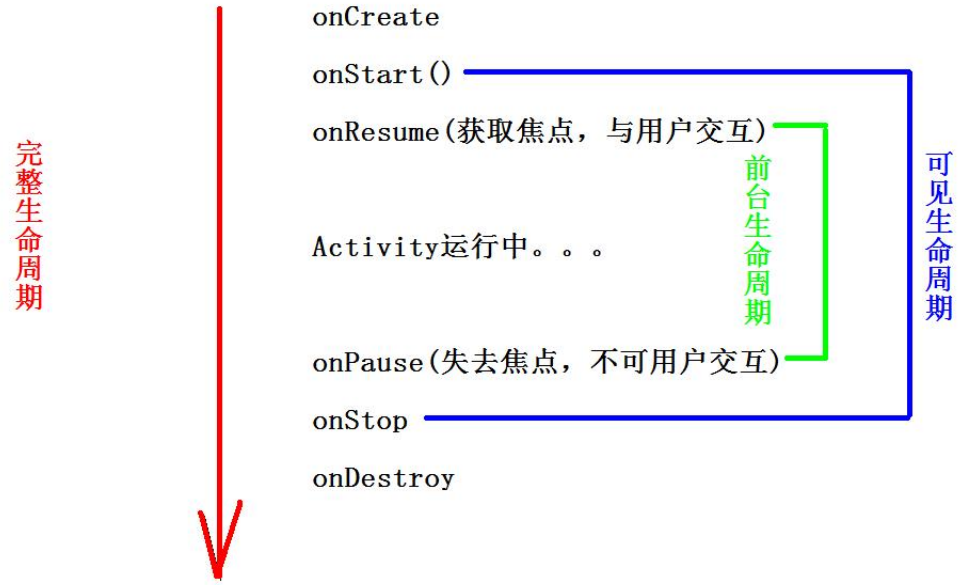
如果一个activity的创建需要占用大量的系统资源（cpu，内存）一般配置这个activity为singletask的启动模式。webkit内核(c) 初始化需要大量内存如js解析引擎、html渲染引擎、http解析、下载...如果使用singletask模式，可以减少内存开销，cpu占用。

- singleInstance

这种启动模式比较特殊，它会启用一个新的任务栈，activity会运行在自己的任务栈里，这个任务栈里面只有一个实例存在并且保证不再

有其他Activity实例进入。在整个手机操作系统里面只有一个实例存在。

应用场景：来电界面。




```
//任务栈中装的是：Activity
//何时创建任务栈:点击应用程序图标时，创建任务栈，且将第一个界面装入任务栈
//何时销毁任务栈:当任务栈中无界面时，销毁任务栈

//每个界面都可以获取界面所在任务栈的编号，该编号唯一
int taskId = this.getTaskId();
System.out.println("taskId=" + taskId);
```