# android activity相关

# Activity启动模式

## Activit任务堆栈

Activity中的任务是与用户交互的一组Activity的集合，Activity会被按打开顺序安排在一个堆栈里。

当应用运行起来后就会开启一条线程，线程中会运行一个任务栈，当Activity实例创建后就会放入任务栈中。Activity启动模式的设置在AndroidManifest.xml文件中，通过配置Activity的属性android:launchMode=""设置。

launchMode在多个Activity跳转的过程中扮演着重要的角色，它可以决定是否生成新的Activity实例，是否重用已存在的Activity实例，是否和其他Activity实例公用一个task里。这里简单介绍一下task的概念，task是一个具有栈结构的对象，一个task可以管理多个Activity，启动一个应用，也就创建一个与之对应的task。

## 1.1.5.2. standard模式（默认）

系统默认创建Activity是standard模式的。这种模式表示，系统每次创建Activity时，都会创建新实例，并放在task栈顶，返回时会逐个activity退出。

## singleTop模式

这种模式系统会检查当前要启动的Activity实例在任务栈中是否正处于栈顶，如果处于栈顶则无需重新创建新的实例，会重用已存在的实例，否则会建新的实例并放在任务栈顶中。

## singleTask模式

当activity的模式为singleTask的时候，会在task把activity以上的实例全部清除，调用该实例的newInstance()方法重用该Activity，使该activity成为栈顶实例，显示在屏幕前。如果task中不存在该activity的实例，则重新创建。

## singleInstance模式

此模式一般用于加载较慢的，比较耗性能且不需要每次都重新创建的Activity。

singleInstance会启用一个新的栈结构，将Acitvity放置于这个新的栈结构中，并保证不再有其他Activity实例进入。当该模式Activity实例在任务栈中创建后，只要该实例还在任务栈中，即只要激活的是该类型的Activity，都会通过调用实例的newInstance()方法重用该Activity，此时使用的都是同一个Activity实例，它都会处于任务栈的栈顶。当调用到onNewIntent(intent)的时候，需要在onNewIntent() 中使用setIntent(intent)赋值给Activity的Intent.否则，后续的getIntent()都是得到老的Intent。

## 总结

singleTop：如果启动activity位于task栈顶，则直接使用，否则创建新实例放入task。

singleTask：如果activity存在task中就使用，并删除重用activity前面的activity，重用的activity置顶任务栈，否则创建新实例。

singleInstance：在一个新栈中创建activity实例，并让多个应用共享该栈中的该activity实例。（调用Activity和重用Activity不在一个栈中）

## 参考文档

Android:SingleInstance填坑——优雅地返回到上一个Activity

http://blog.sina.com.cn/s/blog\_48964b120102yw8y.html

Android SingleInstance的Activity与悬浮窗切换逻辑(startActivity无反应)

https://www.jianshu.com/p/f410e77931a6

Android 关于SingleInstance Activity的注意事项

https://juejin.cn/post/6901479273283911688

# ViewPager2 基础使用

## 参考文档

Androidx 中的 ViewPager 与 ViewPager2

<https://www.jianshu.com/p/924046eae137>

学不动也要学！深入了解ViewPager2

<https://blog.csdn.net/qq_20521573/article/details/103484026>

实现ViewPager和ViewPager2的Fragment懒加载

<https://www.cnblogs.com/swalka/articles/16839096.html>

# Fragment 生命周期

式样

微软雅黑 Light 5号

段落 固定值 23磅

# 目录1

## 目录2

### 目录3

#### 目录4

##### 目录5