一个task（任务）是一系列Activity的集合，这些Activity按照它们被开启的顺序加入到一个stack里，这个stack称为back stack（返回栈）。如果用户点击返回按钮，后加入的Activity就会被移出back stack，当所有Activity都被移除，back stack也不复存在。

Android 7.0 (API level 24)开始支持多窗口，系统为每个Window单独管理task，每个Window可以有多个task。

Task作为一个整体单位，当用户开始了新的task或者回到Home screen，旧的task会移到后台，里面所有的Activity都会stop。用户可以将旧的task再次切换回前台。

可以通过设置<activity>的属性和intent的 flags来改变Activity与task的关联以及Activity在back stack中的表现。

与task和back stack相关的<activity>的属性：

**taskAffinity**

指定该Activity启动后属于哪个task。

默认情况下（不显式指定其值）同一个APP里面所有Activity都属于相同的task，有共同的affinity（其值为应用包名）。

通过指定taskAffinity可以实现同一个APP中的不同Activity具有不同的affinity，不同APP中的Activity可以具有相同的affinity。

指定taskAffinity在两种情况下能发挥作用：

1. 启动Activity的Intent包含FLAG\_ACTIVITY\_NEW\_TASK（清单文件指定或者Intent flag指定）。
2. 设置allowTaskReparenting=”true”。如此，Activity能够从开启它的task移动到taskAffinity对应的task（当该task回到前台）。举个例子，App1中ActivityA指定allowTaskReparenting=”true”，从App2中启动ActivityA，此时ActivityA属于task2，当App1中的task回到前台并且启动ActivityA，ActivityA会从task2转移到前台task。

**launchMode**

启动模式用来定义一个新的Activity的实例怎样与当前task关联。

指定启动模式有两种方式：清单文件或者intent flags。这两种方式不是一一对应的，如果两种同时指定，intent flags生效。

在清单文件的<activity>节点指定launchMode：

<activity

android:name=”...”

android:launchMode=”standard/singleTop/singleTask/singleInstance”

......

/>

standard:默认模式。系统从开启它的task中创建一个新的Activity实例，并且把intent传给它。该Activity能够被实例化多次，每个实例可以属于不同的task，每个task可以有多个实例。

singleTop：如果Activity的实例已经存在于当前task的栈顶，系统不会创建新实例，而是会传递intent给该实例（调用onNewIntent()）。该Activity能够被实例化多次，每个实例可以属于不同的task，每个task可以有多个实例（栈顶存在的话，最多只有2个）。

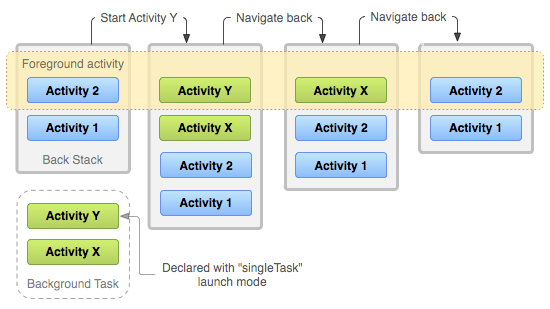
**singleTask**：

系统创建一个新的task，把Activity的实例放到新task的root（前提是指定android:taskAffinity的值为与应用包名不同的值，否则activity的实例属于默认task）。

如果该Activity的实例已经存在于某个单独task中，系统不会创建新实例，而是会移除所在task中该activity之上的所有activity（如果存在），然后传递intent给该实例（调用onNewIntent()）。

ActivityA指定启动模式为singleTask，启动ActivityA后属于task1，那么task1里面只能有一个ActivityA的实例（可以有其它Activity的实例）。例子：Android Browser app

下图展示，如果前台task启动后台task中的singleTask ActivityY，后台task里ActivityY及其下面的Activity都会被带到当前task的栈顶。Activity Y上面的Activity（如果有）被销毁。



singleInstance：与singleTask相似（也需要指定android:taskAffinity的值为与应用包名不同的值），除了一点，系统不会在持有该Activity实例的task中启动任何其它Activity。该Activity实例永远是其所在task中的唯一成员（与singleTask的唯一不同），任何从该task中启动的Activity都是在单独的task中开启。

不论Activity是在当前task还是在新的task中开启，返回键总能将用户导航到前一个Activity。

allowTaskReparenting

见taskAffinity

默认情况下，如果用户离开一个task很长一段时间，系统会清除task中的所有Activity除了root activity。当用户再次回到该task，只有root activity resume。

alwaysRetainTaskState

如果task的root activity的该属性设为true，上面描述的默认行为不回发生，task会保留所有activity即使离开该task很长一段时间。

clearTaskOnLaunch

如果task的root activity的该属性设为true，只要用户离开task，即使一小段时间，再回到该task，task里面除root activity之外都会被清除。跟alwaysRetainTaskState=”true”的作用相反。

finishOnTaskLaunch

与clearTaskOnLaunch类似，不同的是，finishOnTaskLaunch对单个Activity（包括root activity）起作用，而不是整个task。如果设为true，当用户离开后返回到该task，activity不复存在。

与task和back stack相关的intent的 flags：

Intent intent = new Intent();

intent.addFlags(FLAG\_ACTIVITY\_NEW\_TASK|FLAG\_ACTIVITY\_CLEAR\_TOP); // 结合两种flags

startActivity(Intent);

FLAG\_ACTIVITY\_NEW\_TASK

与指定singleTask行为相同。

FLAG\_ACTIVITY\_SINGLE\_TOP

与指定singleTop行为相同。

**FLAG\_ACTIVITY\_CLEAR\_TOP**

如果开启的Activity已经在**当前task**中运行，系统不会创建新该Activity的新实例，而是销毁该实例上面的所有Activity，然后通过onNewIntent()将intent传递给当前栈顶的Activity。

需要注意的是，只有指定launchMode=”standard”并且没有设置FLAG\_ACTIVITY\_SINGLE\_TOP

，Activity将会销毁后重建。其它所有情况，都不会重新创建Activity的新实例，而是将intent传递给其onNewIntent()。

没有相应的launchMode与之等价。通常FLAG\_ACTIVITY\_CLEAR\_TOP和FLAG\_ACTIVITY\_NEW\_TASK结合使用，用于将带开启的Activity置于单独的task中，并能够响应intent。

**开启task：**

<activity ... >

<intent-filter ... >

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

...

</activity>

该Activity的icon和label会显示在App的launcher上，用户点击该launcher能够启动Activity（假设属于task1），并且task1到后台后，任何时候都能点击该launcher使task1回到前台。

不考虑Recents Screen的情况下，应该只为这种launcher activity指定singleTask或singleInstance，因为如果为别的Activity指定这两种启动模式，对应的task切换到后台以后，用户没有途径再让它回到前台。

**Recents Screen**