[一，Activity 2](#_Toc484074756)

[1.Activity的概念和作用 2](#_Toc484074757)

[2.创建Activity 2](#_Toc484074758)

[3.打开一个页面 3](#_Toc484074759)

[①Intent的初始化 3](#_Toc484074760)

[②Intent标记 4](#_Toc484074761)

[③不同的activity可以设置不同的launchMode 4](#_Toc484074762)

[④android:configChanges属性 5](#_Toc484074763)

[⑤android:windowSoftInputMode属性 5](#_Toc484074764)

[4.生命周期 5](#_Toc484074765)

[①onCreate 6](#_Toc484074766)

[②onResume 7](#_Toc484074767)

[③onNewIntent 7](#_Toc484074768)

[④onActivityResult 7](#_Toc484074769)

[5.横竖屏切换和生命周期的联系 7](#_Toc484074770)

[①在主配置文件中，不设置android:configChanges 8](#_Toc484074771)

[②在主配置文件中，设置android:configChanges 9](#_Toc484074772)

[二，Service 10](#_Toc484074773)

[1.Service的概念和作用 10](#_Toc484074774)

[2.Service的启动 11](#_Toc484074775)

[①通过startService启动 11](#_Toc484074776)

[②通过bindService启动 11](#_Toc484074777)

[3.Service的生命周期 12](#_Toc484074778)

[4.生命周期和横竖屏切换的联系 12](#_Toc484074779)

[三，附代码 13](#_Toc484074780)

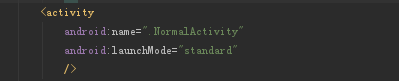
## 一，Activity

### 1.Activity的概念和作用

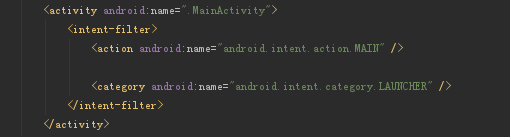
Activity是 Android 程序的基本组成单元，是Android 应用层开发的四大组件之一，是Android应用程序的一个用户接口，用户和应用程序直接进行交互的接口。同时Activity也是一个各种控件的一个容器，我们把控件摆放进去。  
 从[设计](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=8&fv=0&jk=2bc1de51e707a202&k=%C9%E8%BC%C6&k0=%C9%E8%BC%C6&kdi0=0&luki=1&n=10&p=baidu&q=67090150_cpr&rb=0&rs=1&seller_id=1&sid=2a207e751dec12b&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1978320&u=http%3A%2F%2Fwww%2Etry8%2Ecn%2Fthread%2D513%2D1%2D1%2Ehtml&urlid=0" \t "_blank)层面上来讲：功能类似于Mvc设计模式中的Controller控制层，在Android中，通过Activity选择要显示的View,从View中获取数据然后传给Model层进行处理，最后显示出来。

### 2.创建Activity

1，创建一个新Activity，我们需要让我们的类继承Activity（继承AppCompatActivity最终也是继承Activity），然后在主配置文件中，进行配置，类似于这样：



如果我们写的这个Activity是我们的启动页面，那么我们就需要在配置中额外加一些：



2，接下来，我们需要在activity里面增加控件，否则就是一片空白。一般来说，我们会在onCreate中通过setContentView来设置一个View，做为内容。这个方法，即可以传递一个View实例进去，也可以传递一个res.layout.xml文件进去（写法类似于：R.layout.activity\_main）。

3，而后，我们需要对这样的layout文件进行编写。如下图，就是一个简单的页面。其中根容器就是LinearLayout（线性布局，顾名思义，这种布局就是可以横向，也可以纵向，这个布局里面的元素最终都是一条线展示，可以通过android:orientation设置，如果设置成vertical，就是纵向，如果设置成horizontal，就是横向，默认横向），除了这种布局，还有RelativeLayout（相对布局，相对的意思就是里面的元素，谁在谁左边，谁在谁中间），FrameLayout（帧布局，里面的元素，每个元素都是浮在上一个元素的表层）等。

4，xml中有一些比较重要的属性，如

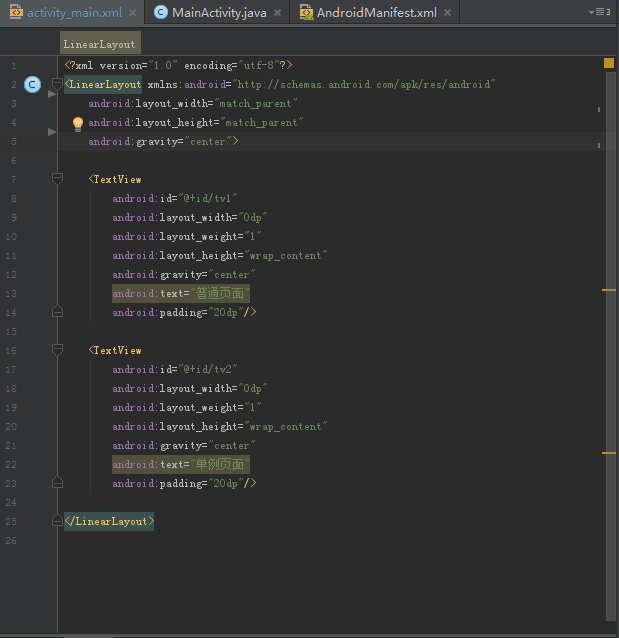
android:layout\_width该元素的宽度（match\_parent，wrap\_content，具体数值）

android:layout\_height该元素的高度（match\_parent，wrap\_content，具体数值）

android:layout\_weight该元素在父布局中的权重（影响高度或者宽度）

android:gravity该布局中的子元素如何排版（居中，上下居中，左右居中。。。）

android:scaleType这个属性是ImageView的，设置图片的剪切方式（http://blog.csdn.net/larryl2003/article/details/6919513）



### 3.打开一个页面

#### ①Intent的初始化

通过以上的设置，我们可以在运行项目的时候，看到我们写的第一个页面，然后，我们需要实现跳转到其他页面，简单的写，可以这样：



其中startActivity是Activity的一个方法，可以开启一个新的页面。传递的参数是一个Intent（意图），表示，你将要去什么页面。可以使用以下几种初始化方法：

1. Intent的构造方法，直接把需要打开的页面的class传递进去
2. Intent的构造方法，new Intent(Intent.ACTION\_CALL, Uri.parse("tel:13811111111"))，直接把Action和URI传递进去
3. 设置package和className来跳转到指定页面intent.setClassName("packageName", "className");。

#### ②Intent标记

Intent打开的时候，会附加很多标记（flags），常用的如下：

1. **FLAG\_ACTIVITY\_NEW\_TASK**我们启动的每个Activity都是会保存在Activity栈中，使用这个标记后，表示保存在一个新栈中(如果没有该页面对应的栈Task就是新建)，一般在打开其他app页面的时候，使用。同时，如果是非Activity上下文调用startActivity，也需要附加这个标记，否则，会报错。
2. **FLAG\_ACTIVITY\_CLEAR\_TOP**当Intent对象包含这个标记时，如果在栈中发现存在Activity实例，则清空这个实例之上的Activity，使其处于栈顶。
3. **FLAG\_ACTIVITY\_SINGLE\_TOP**当task中存在目标Activity实例并且位于栈的顶端时，不再创建一个新的，直接利用这个实例。
4. **FLAG\_ACTIVITY\_CLEAR\_WHEN\_TASK\_RESET**如果一个Intent中包含此属性，则它转向的那个Activity以及在那个Activity其上的所有Activity都会在task重置时被清除出task。当我们将一个后台的task重新回到前台时，系统会在特定情况下为这个动作附带一个FLAG\_ACTIVITY\_RESET\_TASK\_IF\_NEEDED标记，意味着必要时重置task，这时FLAG\_ACTIVITY\_CLEAR\_WHEN\_TASK\_RESET就会生效。
5. **FLAG\_ACTIVITY\_RESET\_TASK\_IF\_NEEDED**这个标记在以下情况下会生效：1.启动Activity时创建新的task来放置Activity实例；2.已存在的task被放置于前台。系统会根据affinity对指定的task进行重置操作，task会压入某些Activity实例或移除某些Activity实例。
6. 网上会有更多的事例，<http://blog.csdn.net/chaoyue0071/article/details/46744787>

#### ③不同的activity可以设置不同的launchMode

1>**standard**：Activity的默认加载方法，该方法会通过跳转到一个新的activity，同时将该实例压入到栈中（不管该activity是否已经存在在Task栈中，都是采用new操作）。例如： 栈中顺序是A B C D ，此时D通过Intent跳转到A，那么栈中结构就变成 A B C D A ，点击返回按钮的 显示顺序是 D C B A，依次摧毁。

2>**singleTop**：singleTop模式下，当前Activity D位于栈顶的时候，如果通过Intent跳转到它本身的Activity （即D），那么不会重新创建一个新的D实例，所以栈中的结构依旧为A B C D，如果跳转到B，那么由于B不处于栈顶，所以会新建一个B实例并压入到栈中，结构就变成了A B C D B。和**FLAG\_ACTIVITY\_SINGLE\_TOP**相似

3>**singleTask**：singleTask模式下，Task栈中只能有一个对应Activity的实例。例如：现在栈的结构为：A B C D。此时D通过Intent跳转到B，则栈的结构变成了：A B。其中的C和D被栈弹出销毁了，也就是说位于B之上的实例都被销毁了。和**FLAG\_ACTIVITY\_CLEAR\_TOP**相似

4>**singleInstance**：singleInstance模式下，会将打开的Activity压入一个新建的任务栈中。例如：Task栈1中结构为：A B C ，C通过Intent跳转到了D（D的模式为singleInstance），那么则会新建一个Task 栈2，栈1中结构依旧为A B C，栈2中结构为D，此时屏幕中显示D，之后D通过Intent跳转到D，栈2中不会压入新的D，所以2个栈中的情况没发生改变。如果D跳转到了C，那么就会根据C对应的launchMode的在栈1中进行对应的操作，C如果为standard，那么D跳转到C，栈1的结构为A B C C ，此时点击返回按钮，还是在C，栈1的结构变为A B C，而不会回到D。

#### ④android:configChanges属性

1. 横竖屏相关

在4.0以前，需要设置orientation|keyboardHidden或者orientation|screenSize就可以监听到onConfigurationChanged；

在4.0以后，必须配置orientation|screenSize才能监听到onConfigurationChanged

1. 键盘相关

keyboardHidden和keyboard目前还没发现在4.0以后起着什么样的作用。

#### ⑤android:windowSoftInputMode属性

1. 在没有滚动控件的页面中，

adjustPan可以让最下方的输入框被推到输入法上方，这样，使得输入框可见。adjustResize就不能把输入框推到上方，结果就是，无法看到输入框。

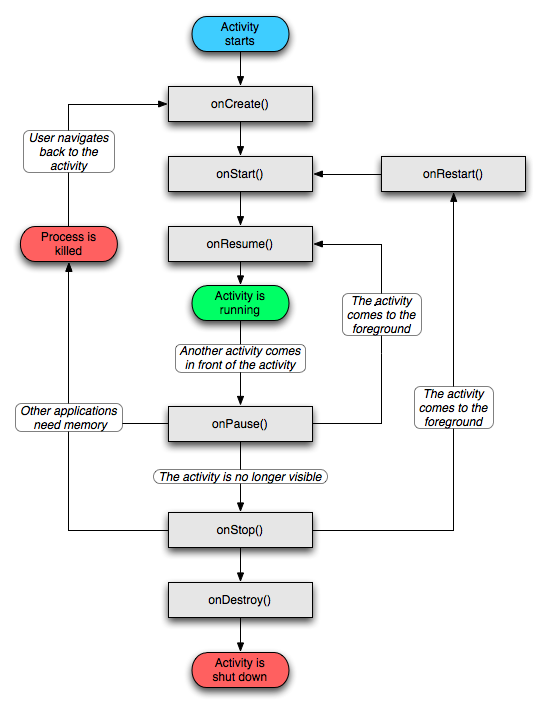
1. 在有滚动控件的页面中，

adjustPan可以让最下方的输入框被推到输入法上方，这样，使得输入框可见，但是上面的页面，无法滚动了。adjustResize同样可以实现改变区域高度，来使得输入法可见，并且，上面的区域可以滚动。

3> 想要了解更多，1可以查看demo（首页的【键盘模式】），2可以看这个地址http://m.jb51.net/article/56043.htm

### 4.生命周期

借用一下网络上面的生命周期图片:



生命周期中有几个比较重要的，如下

#### ①onCreate

每次创建的时候执行一次。我们一般在这个方法中，执行，初始化View或者加载xml资源的任务；以及缓存数据处理，或者发送网络请求。同时，如果我们想要设置Activity全屏，也可以在这里面执行：（需要在setContentView之前）

requestWindowFeature(Window.FEATURE\_NO\_TITLE);

getWindow().setFlags(WindowManager.LayoutParams.FLAG\_FULLSCREEN, WindowManager.LayoutParams.FLAG\_FULLSCREEN);

#### ②onResume

每次页面得到展示，就会执行这个方法，一般，我们不做耗时的操作，更多的是一些逻辑操作（页面一展示，我们就需要执行的一些操作）

#### ③onNewIntent

当我们的Activity设置成非standard模式，然后我们在AActivity上面再次打开AActivity的时候，就会进入onNewIntent方法；这个一般是我们的首页，重复打开首页，我们会根据传递进来的参数，来判断，最终首页的展示方式。

#### ④onActivityResult

当我们从A页面，使用startActivityForResult的方法打开B页面，然后关闭B页面，我们就能在A页面中接收到onActivityResult事件，B可以把一些处理结果返回回来。然后A可以根据B返回的数据，进行处理。例如A页面打开一个系统相册，选择一个图片（会自动关闭相册），会在A 页面的onActivityResult中，返回给我们图片的content地址。

关于生命周期，需要注意以下几点：

1>只有当另外一个Activity要覆盖在当前Activity的时候，当前Activity才会执行onPause方法，以AlertDialog为例，当Dialog弹出的时候，我们的Activity并不会执行onPause(5.0;6.0;2.3测试通过)。如果我们需要监听这样的事件，那我们可以使用onWindowFocusChanged方法。

2>一般情况下，AActivity打开BActivity的时候，AActivity会先进入onPause，然后进入onStop方法；除了，BActivity页面是透明的情况，这种情况，AActivity页面，不会进入onStop。

3>当我们的AActivity页面设置了launchMode（standard，singleTop，singleTask，singleInstance）并且属性不是standard，那么，当我们在AActivity上面再次打开AActivity的时候，就会进入onNewIntent方法。

### 5.横竖屏切换和生命周期的联系

在我们了解横竖屏切换的之前，我们需要了解下，三个生命周期内方法onSaveInstanceState，onRestoreInstanceState，onConfigurationChanged。

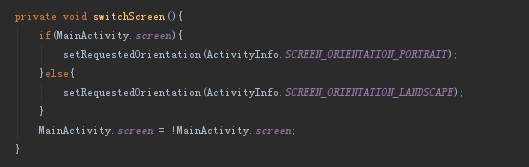
批注1，onSaveInstanceState：(1)在Activity被覆盖或退居后台之后，系统资源不足将其杀死，此方法会被调用；(2)在用户改变屏幕方向时（一般情况），此方法会被调用；(3)在当前Activity跳转到其他Activity或者按Home键回到主屏，自身退居后台时，此方法会被调用。第一种情况我们无法保证什么时候发生，系统根据资源紧张程度去调度；第二种是屏幕翻转方向时，系统先销毁当前的Activity，然后再重建一个新的，调用此方法时，我们可以保存一些临时数据；第三种情况系统调用此方法是为了保存当前窗口各个View组件的状态。onSaveInstanceState和onPause的调用顺序是不一定的。

批注2，onRestoreInstanceState：(1)在Activity被覆盖或退居后台之后，系统资源不足将其杀死，然后用户又回到了此Activity，此方法会被调用；(2)在用户改变屏幕方向时（一般情况），重建的过程中，此方法会被调用。我们可以重写此方法，以便可以恢复一些临时数据。onRestoreInstanceState的调用顺序是在onStart之后。

批注3，如果我们给当前的Activity在主配置文件中，设置了android:configChanges属性（*orientation|screenSize*），那么我们在Activity中切换方向就只能监听到onConfigurationChanged，上面的onSaveInstanceState，onRestoreInstanceState在切换屏幕的时候都是无法监听的。我们提到的改变方向是通过setRequestedOrientation方法来实现的（也可以设置android:screenOrientation="*sensor*"）。

当用户打开客户端的时候，以demo为例，默认是竖屏的；当然如果用户开启了屏幕旋转，那么，他旋转下屏幕，我们的demo就变成横屏了。这个情况不在我们的讨论范围内，我们先讨论，通过通过代码来设置横竖屏（即通过setRequestedOrientation方法设置）

在Activity中使用以下代码：（MainActivity.screen只是一个布尔类型，标记横竖屏的情况）



这里需要分两种情况，

#### ①在主配置文件中，不设置android:configChanges

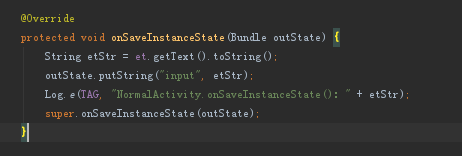
不设置android:configChanges=” *orientation|screenSize*”

这种情况下，我们的客户端会执行以下生命周期

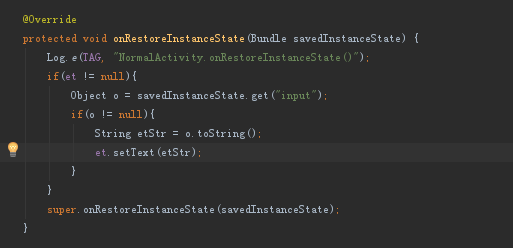
首先会销毁原先的Activity（onPause->onSaveInstanceState->onStop->onDestroy）,然后重建该Activity（onCreate->onStart->onRestoreInstanceState->onResume）

看到这样的执行过程，我们想到了，我们的demo一点影响都没有，因为demo太简单了。可是，如果是正常的项目，那么就会出问题，因为横竖屏切换后，原先页面上面的所有数据，都被destroy了，横屏变成竖屏，或者竖屏变成横屏，整个页面，都会重新执行onCreate（比如说，用户在输入框中输入的文字，在切换的时候，会消失）。所以，我们需要做一些工作。

基于对onSaveInstanceState方法的了解，我们在横竖屏切换的时候，应该在这个方法中保存一些用户的数据。比如说，用户输入了一些东西。



我们将用户输入的内容通过Bundle保存起来。这个Bundle会在执行onRestoreInstanceState的时候传递过去。接下来，我们在onRestoreInstanceState方法中恢复用户的数据，就可以了：



以上的简单处理，就可以保存下来用户的输入数据。

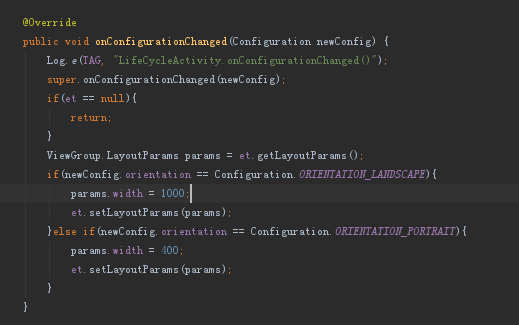
总结一下，如果我们在onCreate中启动一个线程比如说handler.postDelayed(r, 10000);那么，即使我们横竖屏切换，执行了onDestroy，我们的线程也不会停止，下次再次onCreate的时候，还会重新启动一个线程。所以，要么我们在activity中不做类似的业务操作，要么，我们在对应的生命周期中做好“销毁”处理。

同理，如果我们在onCreate中启动一个服务Service，（比如说以startService方式启动），那么他就会走Service的onCreate和onStartCommand，如果我们不在onDestroy中做stop处理，横竖屏切换的时候，还会执行startService，那么就还会再次执行Service的onStartCommand方法，如果我们这个方法中做了一下业务逻辑，那么，我们的代码就会再走一遍，业务很有可能就混乱了。

#### ②在主配置文件中，设置android:configChanges

设置android:configChanges=”orientation|screenSize”

这种情况下，我们在做横竖屏切换的时候，并不会走上面一种情况的周期，代码只会执行onConfigurationChanged这个方法。那么这个方法，我们该写一些什么内容呢？比如说，我们在代码中设置了某些View固定的长度，比如说横屏的时候，长度是1000px，可是切换到竖屏的时候，很有可能，1000px已经超出了手机的宽度。那么这个时候，我们就需要在onConfigurationChanged中做一些判断，然后动态修改控件的长度。当然这只是举个例子。



## 二，Service

### 1.Service的概念和作用

Android中的Service，其意思是“服务”，它是在后台运行，不可交互的。Service自己不能运行，需要通过某一个Activity或者其它Context对象来调用，如Context .startService() 和Context.bindService()两种方式启动Service 。

Android中的服务，它与Activity不同，它是不能与用户交互的，不能自己启动的，运行在后台的程序，如果我们退出应用时，Service进程并没有结束，它仍然在后台运行，那我们什么时候会用到Service呢？比如我们播放音乐的时候，有可能想边听音乐边干些其他事情，当我们退出播放音乐的应用，如果不用Service，我们就听不到歌了，所以这时候就得用到Service了，又比如当我们一个应用的数据是通过网络获取的，不同时间（一段时间）的数据是不同的这时候我们可以用Service在后台定时更新，而不用每打开应用的时候在去获取。

如果在Service的onCreate或者onStart方法中做一些很耗时的动作，最好是启动一个新线程来运行这个Service，因为，如果Service运行在主线程中，会影响到程序的UI操作或者阻塞主线程中的其它事情。

关于Service和AIDL的使用，参考：http://www.jianshu.com/p/123a94ac9b42

### 2.Service的启动

#### ①通过startService启动

Service启动的时候会经历生成开始（onCreate->onStartCommand）过程，Service停止的时候直接进入销毁过程（onDestroy）。而如果是调用者直接退出而没有调用stopService，Service会一直在后台运行。直到下次调用者再启动起来，并明确调用stopService。

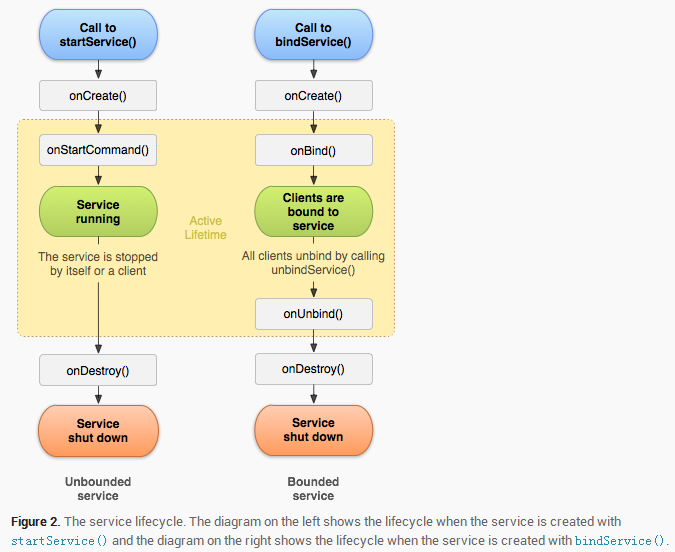
#### ②通过bindService启动

通过bindService方法启动Service，其运行onCreate->onBind方法，如果调用退出了，Service会调用onUnbind，onDestroyed方法。Service的onCreate方法只会被调用一次。如果先绑定了，那么启动的时候就直接运行Service的onStartCommand方法，如果先启动，那么绑定的时候就直接运行onBind方法。如果先绑定上了，就停止不了，也就是stopService不能用了，只能先unbindService ,再stopService，所以，先启动还是先绑定，是有区别的。

stopService方法调用，会经历（onDestroy）,如果没有startService，执行这个方法，不会有任何效果。

unbindService方法调用，会经历（onUnbind->onDestroy），没有bindService是不可以执行unbindService。

### 3.Service的生命周期



同样，参考官网的一个图片，我们来说说Service生命周期需要注意的地方

1>两种开启Service的方式有一些生命周期的不同；如果只是startService，那么我们一般就只是当作一个异步任务。这种情况比较少，一般，我们都会配合bindService，来把Service当作一个服务器端，我们调用bindService的就是客户端。

2>startService可以不停的调用，第一次调用会执行onCreate，后面不会执行，后面会执行onStartCommand；bindService，如果不停调用，实际上并不会产生作用。

3> onStart这个方法，在2.0以后就不使用了，改用onStartCommand。

### 4.生命周期和横竖屏切换的联系

①Service在主配置文件中，没有android:configChanges可以配置，所以当横竖屏切换的时候，Service的onConfigurationChanged默认执行。

②以bindService的方法开启Service，横竖屏切换的时候，会因为绑定服务端（Service）的客户端（Activity）会被销毁，所以，需要主动执行unBind()方法，否则会报错；即使我们不主动执行，最终，我们的Service也会执行onDestroy。但是如果客户端配置了android:configChanges，使得客户端只会执行onConfigurationChanged的话，那么Service也就只会执行onConfigurationChanged了。这里有一点需要注意下，如果是有多个客户端同时连接了服务端，那么其中一个客户端onDestroy后，并不会使得服务端就销毁了。

③以startService的方法开启Service，横竖屏切换的时候，只会收到onConfigurationChanged事件，其他的回调，都不会执行。

## 三，附代码

地址：https://github.com/zqmao/LifeCycle