# 面试题基础储备

## 1、Activity相关

**a、Activity的特点**

1、可见 2、可交互

他之所以可交互，是因为他同时实现了Window.Callback和KeyEvent.Callback, 可以处理与窗体用户交互的事件和按键事件.这两个特点，是他和service最大的 区别。一个Activity在创建与销毁的过程中，会经历一些生命周期。

**b、Activity的生命周期**



结论1、这个界面只要看不到了，它就一定执行了onStop方法

结论2、只要这个界面显示出来了，它就一定执行了onResume方法

结论3、onPause、onStop的情况下这个activity都有可能会系统回收。（简单 来说，只要这个activity不处于活跃状态，那么它就有可能被回收）

**c、页面跳转必然会执行的方法**

onResume，所以可以在onResume方法里面进行数据刷新，保证当前activity 显示的都是最新状态

**d、对话框的activity**

启动activity默认是占满整个屏幕，如果想让这个activity以对话框的方式展示， 则需要配置：android:theme="@android:style/Theme.Dialog"，此时它下面的 activity并不会执行onStop方法，而是仅执行onPause方法

**e、横竖屏切换**

默认情况下，横竖屏切换的时候会重新创建新的activity，新的那个activity会 执行onCreate方法。可以通过AndroidManifest文件进行配置，让它横竖屏切 换的时候不重新创建Activity，配置方法为

android:configChanges="orientation|keyboardHidden|screenSize"，配置 之后就不会创建新的Activity，而是执行当前activity的 onConfigurationChanged。写死屏幕的方向：在AndroidManifest中配置

android:screenOrientation="landscape"

**f、保存数据**

除了在栈顶的activity，其他的activity在系统资源匮乏的时候，都有可能会被 系统回收。如果activity被系统没有onDestory的情况下就被系统回收了，这 时候系统会调用onSaveInstanceState方法，我们可以往bundle里面存放数据。 在activity onCreate里面我们先判断一下bundle是不是为空，如果不为空，就 代表这个activity之前被系统回收掉，应该恢复一下现场。我们就可以从bundle 里面取值。

**g、栈相关**

概念：Android是用栈来管理Activity的，service是没有栈的。所以在service 启动activity一般要加上

intent.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK)来开辟一个 新的栈。

启动模式：android:launchMode，

standard：默认的模式，没有限制，栈里爱有几个有几个

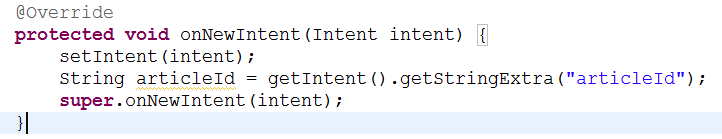
singleTop：栈顶只有一个，栈内可以有多个

singleTask：整个栈只有一个

singleInstance：霸道，整个栈只有自己

**h、onNewIntent**

当启动某一个activity，这个activity在栈中存在，并且需要被复用的情况下， （也就是配置了singleTop、singleTask、singleInstance的 属性），会调用此方 法。如果需要传递参数，需要在onNewIntent里面setIntent，这样才能更新者 个activity的intent

****

具体面试题：1、2、3、4、5、6、7、11、40、42

## 2、Service相关

**a、Service的特点**

不可见、不可交互

**b、Service的启动方式**

startService、bindService

**c、Service的生命周期**

a、通过startService

Service会经历 onCreate 到onStart，然后处于运行状态，stopService的时候调 用onDestroy方法。

这种方式，activity和service是相互独立的。如果是调用者自己直接退出而没 有调用stopService的话，Service会一直在后台运行

b、通过bindService

Service会运行onCreate，然后是调用onBind， 这个时候调用者和Service绑 定在一起。调用者退出了，Service就会调用onUnbind->onDestroyed方法。

这种方式，activity就和service相互捆绑在一起了。所谓绑定在一起就共存亡 了。调用者也可以通过调用unbindService方法来停止服务，这时候Service就 会调用onUnbind->onDestroyed方法。

d、**为什么使用Service**

普通的线程也可以达到在后台做事情的功能，那么为什么使用 Service呢？是因为 Service是系统的组件，它的优先级比普通的线程要高，不容易被系统回收。而且 线程不好控制，Service相对好控制一些。运行在前台的Activity是不会被系统回收 的，而Service如果不想被系统回收，就需要在Service中设置一下

startForeground(int,Notification)

具体的使用场景有：

a、拥有长连接QQ

b、定时轮询

c、服务里面注册广播接收者。有些广播接收者只能通过代码注册，比如屏幕锁屏、 屏幕解锁

e、**IntentService**

普通的service ,默认运行在ui main 主线程.Sdk给我们提供的方便的、带有异 步处理的service类,我们可以在OnHandleIntent() 处理耗时的操作

具体面试题：8、10、12、13、14、15、16、17、64、68

## 3、数据相关

**a、存储**

在Android中数据可以存储在四个地方

内存：读写速度最快，临时的值，程序退出或者界面退出就没有了

SharedPreference：持久存储，主要保存一些配置信息，如一个功能的开关

Sqlite：持久存储，关系型数据库，针对数据之间有很强关系的情况

文件：持久存储，针对数据量较大、并且没啥关联的数据

**b、传递**

a、通过intent传递

基本的数据类型：String、int、float等

对象：这个对象必须继承Parcelable接口。继承Parcelable接口代表这个对象可以 被序列化到内存中，和Serializable类似，只不过Serializable是将对象写到 文件当中。

b、通过Application传递

Application的特点：一个应用程序运行的时候只有一个，它的生命周期是最长 的，比Activity、Service都长，只要这个程序在运行，无论是否在前台，它都 会有一个Application对象。往Application存某一些对象，在页面跳转的时候 会非常方便。

**c、解析**

XML

效率低，体积大

JSON

效率高，体积小，使用广，可以使用Gson解析，也可以使用JSONObject进行 解析

**d、ContentProvider**

目的：将自己应用的数据提供给其他应用

把自己的数据通过uri的形式共享出去，这个uri是事先约定好的。

android 系统下不同程序 数据默认是不能共享访问，通过ContentProvider可以 将自己应用的数据提供给别的应用。

我们在写内容提供者的时候，需要写一个类继承ContentProvider，然后实现里面的 增删改查方法

query(Uri, String[], String, String[], String)

insert(Uri, ContentValues)

update(Uri, ContentValues, String, String[])

delete(Uri, String, String[])

工作中用的很少，我们几乎不需要写ContentProvider，因为没有这个需求。可能会 用到的是调用系统的ContentProvider，比如获取系统联系人，系统短信。

具体面试题：9、19、23、24、25

## 4、广播相关

**a、广播发送**

广播的发送有两类，一种是系统本身就有的，一种是我们自己写的广播。

发送方式：

有序广播sendOrderedBroadcast

按顺序依次的发送给每一个接收者，而每一个接收者在收到这个广播的时 候，可以将这个广播abort掉，也可以将这个广播继续传递到下一个广播 接收者。广播接收者在注册的时候可以指定优先级，用于提高接收到广播 的顺序。

无序广播sendBroadcast

不能被abort掉

**b、广播接收**

广播接收者注册的方式也有两种，一种是动态注册，一种是静态注册。有一些系统 的广播只能使用动态注册，这种广播产生的频率是比较高的。比如电量变化的广播， 屏幕解锁的广播。电量变化的广播。

**c、广播的作用**

广播一般是用于跨进程通讯的时候。onReceive方法也是在主线程当中运行的。在广 播中启动Activity也是需要配置

intent.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK)来开辟一个新的栈。

具体面试题：21、22

## 5、跨进程访问

**a、Android中跨进程访问的方式：**

主要有两种：1、广播 2、AIDL （启动其他应用activity、内容提供者）

**b、什么是AIDL：**

AIDL的全称是android interface definition language 接口定义语言。它所做 的事情就是跨进程范围另外一个服务里面的方法的。跨进程范围另外一个服务里面 的方法。在手机卫士中挂断电话用到过AIDL。我们要获取手机系统的电话服务， 用到了aidl 它其实工作的原理就是绑定到一个远程的服务上。然后这个远程服务 会返回回来一个代理对象。这个代理对象里面的方法，就是我们定义的aidl里面 的方法。

aidl的写法也非常简单，主要分两种。调用系统服务的话，直接找到那个服务的 aidl拷贝到我们的项目中就可以了。另外一种是我们自己写aidl，主要注意的一 点是aidl是没有访问修饰符的。因为aidl本身就是公开的，就是为了给别人调用 的，不需要私有的private方法。

其实，在实际的开发过程中，比较少会写这个aidl，但是你得懂它相关的知识点。

在面试的时候，这个也是经常被问的。所以大家要熟悉这个问题的回答方法。

我以前面试别人的时候，只要他能回答上几个关键点，我就会让他过我这关噢。

1、跨进程 2、aidl在服务端和客户端都需要有一份 3、通过绑定bindService 的方式获取远程服务对象的代理IBinder，就可以调用相关的方法。

**c、跨进程的底层原理**

广播、aidl 的底层实现机制是Binder，Binder是基于共享内存的一种IPC方式

如何自己实现进程间的通信？

具体面试题：30、32、44、52

## 6、Handler相关

**a、为什么使用Handler**

Handler是用来进行线程间的通信。因为Android系统有一个特性，ui操作必须得 在主线程当中进行。所以在子线程进行完耗时操作的时候，如何告诉主线程进行ui 界面更新，这就需要使用到线程间的通讯了。

**b、Handler、Looper、MessageQueue的关系**

任何一个线程，无论是子线程还是主线程，都维护着一个消息队列，这个消息队列 就是MessageQueue，有了这个消息队列之后，是需要不断的从这个消息队列里面取 出消息进行处理的，这个就是Looper做的事情。那么，谁往这个消息队列里面丢消 息呢？那就是Handler了。

Looper是用来管理所属线程的消息队列MessageQueue的。

每一个线程都需要有一个looper，每一个looper管理一个MessageQueue.

Handler.sendMessage的意思是将某一个message放到MessageQueue中去，looper 是个死循环，不断的读MessageQueue中的新消息。

要让looper的死循环运行起来，得调用Looper.loop()方法。

我们通常都会在子线程中，发一个消息到主线程中的messagequeue中去。

Handler到底是往主线程的MessageQueue发送消息呢还是往子线程的 MessageQueue发送消息呢？这取决于Handler在哪里创建。如果Handler在主线程 中创建，那么这个Handler就会把消息发到主线程的消息队列，如果Handler是在子 线程中创建，这个Handler就会把消息发到子线程的消息队列。这里需要注意的是， 子线程的Looper需要我们自己手动启动，要调用Looper.prepare()和Looper.loop()方 法，主线程的Looper系统已经帮我们启动了，因此我们不需要为主线程的 Looper调用loop()方法

**c、子线程-->主线程 主线程-->子线程？**

具体面试题：29、33、56

## 7、ANR相关

**a、什么是ANR：**

ANR是android not response 安卓无响应，这里指的是主线程无响应。将耗时的 操作放在子线程中进行可以有效的避免ANR。

**b、出现的原因：消息的超时信息**

出现ANR的根本原因是主线程堵塞了，来不及处理消息。任何UI操作都是一个消 息，比如触摸、按键，弹框，setText等，当这些消息得不到及时的处理，就会出现 ANR了。每一个发往主线程的消息其实都带有一个超时时间，超过了这个时间，这 个消息还没有处理，就会出现ANR

**c、解决的方法**

如果出现了ANR，我们可以通过log信息以及traces.txt文件进行分析。

traces.txt里面记录的是stack信息。

traces.txt的路径：/data/anr/traces.txt

具体面试题：36、58、

## 8、ListView的优化

**a、为什么要优化？**

因为不优化的话，滑动ListView会卡顿。卡顿的原因在于getView太耗时了。

getView是在主线程执行的，如果这里面太耗时的话，ui肯定卡顿。耗时主要集中 在两个方面：加载布局文件、findViewById

**b、怎么优化**

减少加载布局文件的次数：使用缓存convertView

减少findViewById的次数：使用ViewHolder

**c、数据错乱的问题**

正是因为使用了缓存，所以才有可能造成数据错乱。所以在刷新数据的时候，一 定要考虑周全。

****

具体面试题：38、

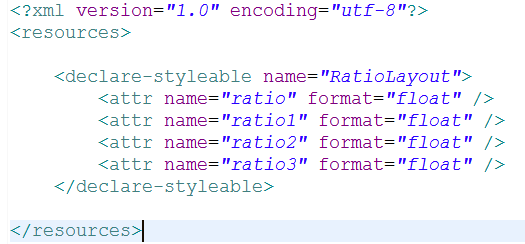
## 9、自定义控件相关

**a、自定义属性**

**步骤：**

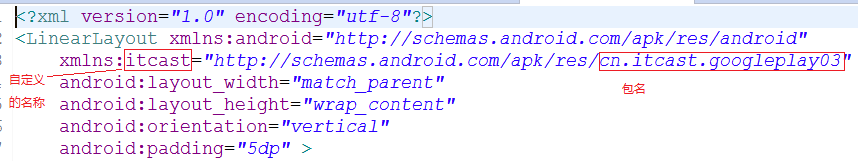
**1、起一个名字**

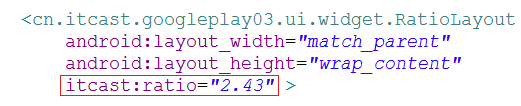
在values的目录下需要建立一个文件，叫做attrs.xml 这个文件是定义属性 规则的。一个属性的名字、取值类型、取值范围

****

**2、用这个属性**

在布局文件中需要加入命名空间

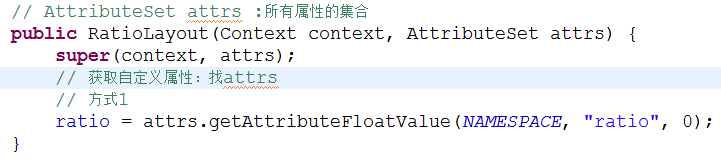
****

****

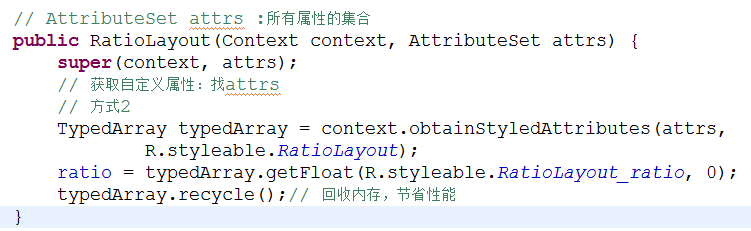
**3、得到这个属性的值**

最后在自定义控件中获取这些值。

方式一：

****

**方式二**

****

**b、绘制流程**

**涉及的方法：**

onMeasure：计算大小

onLayout：计算位置

onDraw：开始绘制

调用顺序是onMeasure-->onLayout--->onDraw

这三个方法是任何一个View在屏幕上呈现出来的时候都一定会调用的三个方 法，三个方法的作用不一样。

**其他知识：**

int widthMeasureSpec---封装了两个信息，一个是具体大小的值，一个是模式

int widthSize = MeasureSpec.getSize(widthMeasureSpec);// 获取宽度

int widthMode = MeasureSpec.getMode(widthMeasureSpec);// 宽度的模式

// MeasureSpec.AT\_MOST;--->wrap\_content至多的模式。

// MeasureSpec.EXACTLY;--->确定的模式，match\_parent,写死成多少dp

// MeasureSpec.UNSPECIFIED;--->未确定的模式 ScrollView:不对孩子的高度进 行限定

具体面试题：35

## 10、动画相关

**分类**

**a、帧动画**

帧动画，就像GIF图片



**b、补间动画**

只是改变了View对象绘制的位置，而没有改变View对象本身。AlphaAnimation、 RotateAnimation、ScaleAnimation、TranslateAnimation

**c、属性动画**

Android 3.0中才引进，更改的是对象的实际属性。如果要在2.x版本也要支持 属性动画，则需要加入兼容包nineoldandroids.jar

具体面试题：59

## 11、设计模式相关

**a、单例：保证就只有一个实例**

单例的写法：

1、私有的构造方法

2、私有的，静态的对象instance

3、公开的，静态的getInstance方法

注意事项：在getInstance的方法前加上线程同步synchronized

**b、适配器：数据和View的桥梁**

**c、观察者:本质就是回调，一堆回调**

## 12、屏幕适配相关

**a、概述：**

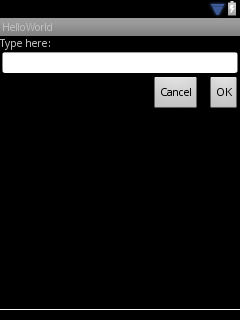
用dp 不用px 这仅仅是适配的一部分而已。这里需要注意的一点是，android是 不可能做到完全适配每一个屏幕的，只能达到大体相近。如果你要达到完全适配， 你就必须在每一个drawable-目录下都放入整套切图，而且要建立不同分辨率的 layout-854\*480的布局文件，这简直是不可想象。没有一个公司会这么做。大部分 的人是怎么做的呢？

**b、切图适配：**

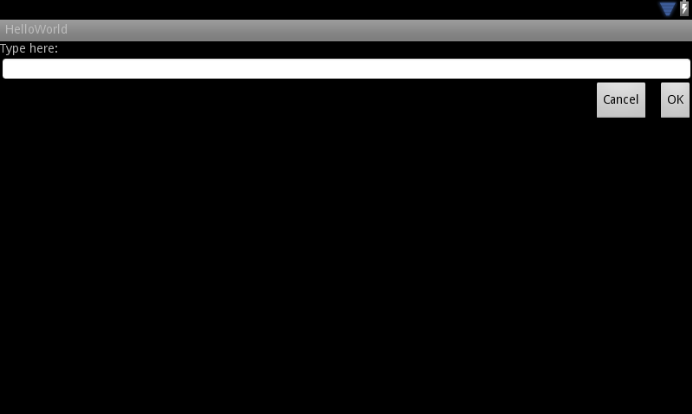
一般都是找一个使用比较广泛的分辨率当作标准。比如1280x720 ，然后设计师 就会根据这个分辨率进行设计，然后切图。我们就需要将设计师出的切图放到 drawable-xhdpi目录下就可以了。一般情况下，我们不需要在drawable-hdpi目录、 drawable-mdpi目录、drawable-xxhdpi目录下放切图了，只在drawable-xhdpi下 放一套就够了。在真正运行的时候，系统会自动对图片进行处理。这时候，绝大部 分切图显示都是正常的。如果你觉得哪几张图片显示不清晰了，就只要再出这几张 切图放到drawable-xxhdpi目录或者其他目录即可。

**c、布局适配：**

尽量用一些确定的东西来确定不确定的东西。多用RelativeLayout、LinearLayout



整体是一个LinearLayout，垂直方向。中间的edittext是match\_parent.最底下整体 布局应该是RelativeLayout，OK按钮是layout\_alignParentRight,确定好OK之后，在 确定cancel的值，切勿不要先确定cancel的位置再确定ok的位置。



按照上诉布局做法，在超大的屏幕上显示的时候发现，edittext变长了点，但是整 个的位置结构是没变的，这个布局的适配就算是过关的。

## 13、图片相关

**a、图片的缓存**

任何一种缓存的目的是都是为了加载更快，那么存在哪里呢？存在内存中是最好的， 因为内存的读取速度最快，但是内存的大小是受限制的。所以，我们只能在内存中 加载一部分图片。使用LRUCache可以对内存中的图片进行有效管理。

三级缓存：内存缓存--本地缓存--网络缓存

**b、大图片的加载**

如果一张图片有几十M，那么这一张图片一加载肯定就OOM了，根本就不需要使 用LRUCache，所以加载大图片的时候需要注意：

1、仅请求图片的大小，inJustDecodeBounds = true，仅请求图片大小，而不会 加载图片到内存；

2、得到图片的大小，再得到所需要显示控件的大小，根据这两个值，合理设置 BitmapFactory.Options的[inSampleSize](http://my.eoe.cn/isnull/archive/564.html" \t "_blank)值，减少图片内存占用

具体面试题：65、

## 14、View的事件传递

**View**：只有dispatchTouchEvent、onTouchEvent

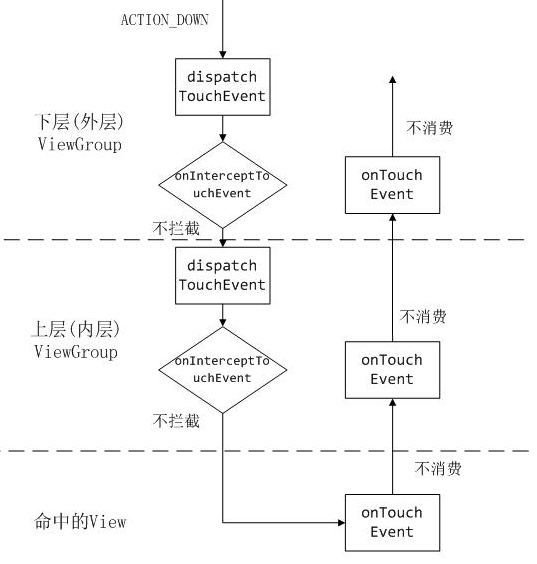
**ViewGroup**：dispatchTouchEvent、onInterceptTouchEvent、onTouchEvent

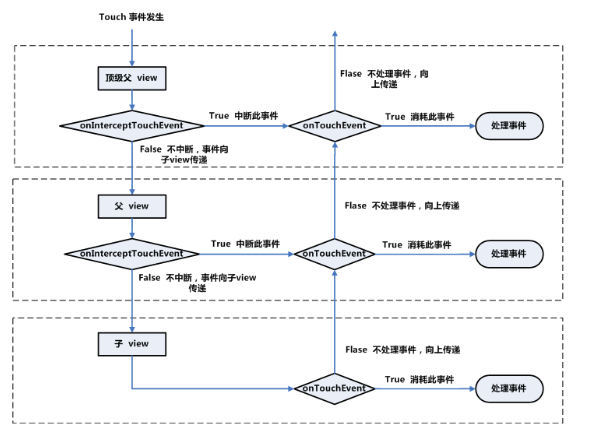
**事件传递**dispatchTouchEvent 为的就是找到响应的那个View 这个方法的返回值会结合 onInterceptTouchEvent的返回值，true代表找到响应的View，false代表没有 找到响应的View

**事件拦截**onInterceptTouchEvent true代表将事件拦截在此ViewGroup中，此View就是响应 的那个ViewGroup

**事件响应**onTouchEvent true代表消费了这个事件，这个事件就不往上传递，false代表 没有消费此事件，此事件继续向上传递。

**事件传递的方向是由父类到子类，事件响应的方向是从子类到父类。**





## 14、内存优化相关

**垃圾回收机制**：垃圾回收器仅会回收没有人引用的对象。

**内存泄露**

内存泄漏本身不会产生什么危害，真正有危害的是内存泄漏的堆积。Android应用内存 泄漏的的原因有以下几个：

1、register之后没有unregister

2、查询数据库后没有关闭游标cursor  file没有close  
 3、构造Adapter时，没有使用 convertView 重用 （内存的溢出）  
 4、Bitmap对象不在使用时调用recycle()释放内存   
 5、对象被生命周期长的对象引用，如activity被静态集合引用导致activity 不能释放

**内存泄漏如何解决：**

通过内存分析工具 MAT(Memory Analyzer Tool)，找到内存泄露的对象

生成hprof文件可以通过adt的工具也可以通过代码生成。debug的dump方法

具体面试题：34、43、55、

## 15、提高ui流畅度

**优化布局层次**。不要不断的嵌套LinearLayout ，多使用RelativeLayout 尽可能的减 少 布局的层次。左边是图片，右边是文字的，可以就使用一个TextView来完成。

详见：

其他面试题

18、20、26、27、28，31、37、39、40、41、

45、46、47、48、49、50、51、53、54、57、62、63、66、67、70、71、72