# Android面试题整理

1. Android相关
2. Android基础知识点

1.横竖屏切换的时候，Activity 各种情况下的生命周期

Android：ConfigChanges属性是用来捕获手机状态的变化的属性。对应生命周期分两种情况：第一种情况是(在android4.0版本之后,不设置android:configChanges属性，或者设置属性该属性不为orientation|screenSize共存即可)横竖屏切换时生命周期表现为oncreate()->onstart()->onresume()->onpause()->onsaveinstancestate()->onstop()->ondestory()->oncreate()->onstart->onrestoreinstancestate()->onresume(),另一种情况(在android4.0之后设置android:configChanges属性值orienttation和screenSize共存,或者在android4.0之前设置该属性值为”orientation|keyboradHidden”横竖屏切换对应的生命周期为oncreate()->onstart()->onresume()->onConfigurationChanged()，再进行横竖屏切换，仍单单只执行onConfigurationChanged()。原因在Android4.0之后支持多分辨率，screenSize是为了屏蔽屏幕大小自适应。

1. Activity与Fragment之间生命周期比较

创建过程Activity带领着Fragment，销毁过程是Fragment带领Activity。

生命周期相对应关系:Activity.oncreate对应着Fragment.onattach,Fragment.oncreate,Frgment.oncreateview,Fragment.onactivitycreated;Activity.onstart对应Fragment.onstart;Activity.onresume对应Fragment.onresume;Activity.onpause对应Fragment.onpause;Activity.onstop对应Fragment.onstop;Activity.ondestory对应Fragment.ondestoryview,Fragment.ondestory,Fragment.ondetach。

总的顺序：Activity.oncreate -> Fragment.onattach -> Fragment.oncreate -> Fragment.oncreateview -> Fragment.onactivitycreate -> Activity.onstart -> Fragment.onstart -> Activity.onresume -> Fragment.onresume -> Fragment.onpause -> Activity.onpause -> Fragment.onstop -> Activity.onstop -> Fragment.destoryview -> Fragment.destory -> Fragment.detach -> Activity.destory

1. Activity上有Dialog的时候按Home键时的生命周期

按照Activity的正常生命周期运行，home键相当于驻后台，再次调用回调onrestart方法。

1. Activity状态保存与恢复

利用onSaveInstanceState(Bundle outState)和onRestoreInstanceState(Bundle inState)以及onCreate(Bundle saveInstanceState)来保存恢复一些数据。

onRestoreInstanceState(Bundle inState)和onCreate(Bundle saveInstanceState)的三个区别：其一，onSaveInstanceState方法不一定会被调用，onCreate方法的bundle值得判空，而当onRestoreInstanceState()一旦被调用，bundle就一定有值；其二，数据恢复时机不同，onRestoreInstance是在onStart()之后调用；其三，oncreate方法必须调用super.onCreate(saveInstanceState)方法，否则报错，而onRestoreInstanceState()方法则可以控制调不调用super.onRestoreInstanceState(inState)方法

1. fragment各种情况下的生命周期

其他情况与Activity一致，唯一不一致的是在按Home键切回应用时，对应的生命周期是：Fragement.onstart，Fragment.onresume

1. Fragment状态保存,startActivityForResult是哪个类的方法，在什么情况下使用？

Fragment的状态保存：1.类似Activity的状态保存恢复机制，不同的是Fragment只有onSaveInstanceState方法没有onRestoreInstanceState方法，它的状态恢复是在onActivityCreated方法中完成的；2.为了保存更多的数据(Bundle是一个map实现类，能存储数据量至多1mb左右)，可在Fragment的oncreate方法中设置setRetainInstance(true)，然后在父容器的onSaveInstanceState方法中通过FragmentManager来保存该Fragment的实例，再在oncreate方法中恢复存储状态。(setRetainInstance方法的特点是保存状态，异常退出，不重复初始化Fragment)

Fragment和FragmentActivity的startActivityForResult特点：以requestCode值0xffff作为界限，大于0xffff的调用Fragment的onActivityResult方法， 小于等于0xffff的调用FragmentActivity的onActivityResult方法，原理在调用Fragment的startActivityForResult方法时，内部源码将requestCode的值限制在大于0xffff的情况了，而在调用FragmentActivity的startActivityForResult方法时，内部源码将requestCode的值限制在小于等于0xffff的值，在FragmentActivity的onActivityResult的回调中通过，requestCode进行回调分发。

1. 如何实现Fragment的滑动？
2. fragment之间传递数据的方式？

Activity与Fragment之间传递数据：方法一利用Fragment的setArguments设置参数，进行传递；方法二，在Fragment中利用getActivity获取持有Activity的实例进行数据获取。

Fragment之间传递数据：同一个container，直接调用setArguments方法设置；不同container，三种方法：方法一，直接在Activity中操作，Activity中都有它们的实例，直接进行操作；方法二，在Fragment中利用getActivity获取Activity实例进行操作；方法三，在Fragment中设置回调接口，在Activity中实现接口，再进行设置传值。

1. Activity 怎么和Service 绑定？

先声明两个类，一个Activity，一个Service，在Service中声明一个内部类继承于Binder，并初始化得到其对应实例，在Service的重写方法onBind中返回该实例，在Activity中初始化一个ServiceConnection实例，调用bindService(Intent intent, ServiceConnection serviceConnection, int flag)，就实现了binder的绑定了。

1. service和activity怎么进行数据交互？

分两种情况：一种是一般启动，此时可通过Intent传递数据，另一种在Service执行一些操作后想要更新UI线程时，通过binderservice绑定的service可在ServiceConnection的回调方法onServiceConnected中获取Service实例，用于操作数据交互，还可以通过广播进行数据交互。

1. 请描述一下Service 的生命周期

两种启动Service方式：

1.startService启动service时的生命周期表现为，onCreate -> onStartCommand -> onDestory，多次启动不会重复调用onCreate方法创建实例，只会重复调用onStartCommand方法，该service将一直在后台运行，直到调用stopService方法或者自身调用stopSelf方法。

2.bindService启动service时的生命周期表现为，onCreate -> onBind -> onDestory，不会调用onStartCommand方法，多次启动不会重复调用onCreate和onBind方法，结束service的方法是调用context.unBindService方法或者对应的context消失。

特殊情况，同时启动又绑定Service，依照startService的方式看待。

1. 说说ContentProvider、ContentResolver、ContentObserver 之间的关系

ContentProvider是四大组件，用来实现跨应用数据共享，ContentResolver是ContentProvider用来操作别的应用数据的相关工具类，ContentObserver类似于触发器，用于Uri引起的数据改变。

1. 广播的分类

传统广播，有序广播，系统广播，App应用内广播(例利用LocalBroadCastManager进行注册的广播或者注册广播时修改其exported属性为false，注册广播时指定包名)，粘性广播(Android5.0已经废弃)

1. 广播使用的方式和场景

动态注册和静态注册两种方式，场景类似于接口与回调的机制。

1. 在manifest 和代码中如何注册和使用BroadcastReceiver?
2. 本地广播和全局广播有什么差别？

前者发送广播只会在应用内接收、效率高、数据安全性高，后者发送广播会在全局接收、速度慢、数据安全性不高。

1. BroadcastReceiver，LocalBroadcastReceiver 区别

同16。

1. AlertDialog,popupWindow,Activity区别

AlertDialog是非阻塞式对话框，AlertDialog弹出时，后台还可以做事情；而PopupWindow是阻塞式对话框，PopupWindow弹出时，程序会等待，在PopupWindow退出前，直到调用dismiss方法后，程序才会向下执行。

1. Application 和 Activity 的 Context 对象的区别

都是Context的子类，生命周期不同，Application的Context对应的是应用的生命周期，Activity的Context对应的是Activity的生命周期。某些工具类使用Activity的context可能会造成内存泄露，结论：跟UI相关的都要使用Activity的context，非UI相关的Service、Activity、Application等实例都是可以的。

1. Android属性动画特性

工作原理：设置属性动画时间间隔，设置属性动画的初始值和结束值，设置属性动画变化期间的变化逻辑，根据逻辑变化改变值，并将对应值赋值给对应属性，每次赋值后调用控件的invalidate方法进行重绘实现动画效果。

1. 如何导入外部数据库?

将db文件拷贝到应用指定的目录文件中SQLiteDatabase.openOrCreateDatabase方法，利用SQLiteDataBase.openDataBase(path,null,SQLiteDatabase.OPEN\_READONLY)方法获取SQLiteDatabase实例，并将外部数据导入，进而对外部数据库进行操作。

1. 谈谈对接口与回调的理解

可以把使用某一接口的类创建的对象的引用赋给该接口声明的变量，那么该接口变量就可以调用被类实现的接口的方法。

1. 介绍下SurfaceView

需要在另外的线程绘制界面、需要迅速的更新界面或渲染UI界面需要较长的时间，这种时候要使用SurfaceView。使用步骤：1.用findViewById获取SurfaceView的实例；2.用其getHolder方法获取holder实例；3.利用holder的holder.lockCanvas()封锁并获取Canvas实例；4.在Canvas上绘制相应图形；5用holder.unlockCanvasAndPost(canvas)方法解锁并将Canvas显示到SurfaceView上面。

24.序列化的作用，以及Android两种序列化的区别

序列化作用：实现数据的持久化，对象通过序列化可在网路中或进程中传递对象。

Serializable实现的序列化代码简单，但是效率不高，Parcelable实现的序列化代码复杂，但是效率高。

(二)Android源码相关分析

1.Android动画框架实现原理

建立在view的绘制机制上，Tween动画原理：在ParentView 调用dispatchDraw方法时绘制子childView时发现它有动画要实现的时候，会不断的调用invalidate方法进行重绘，调用dispatchDraw方法，直到动画结束，过程中操作的属性值实际是一个矩阵。属性动画原理：设置动画的属性的初始和结束值，以多次调用对应的set方法设置属性值。帧动画：反复调用onDraw方法中invalidate方法重绘

2.Requestlayout，onlayout，onDraw，DrawChild区别与联系

requestLayout：会调用view树的Measure过程和Layout过程，但是不会调用任何View的draw过程；

onLayout：Viewgroup中会调用所有子元素进行布局，在View中无需实现该方法；

onDraw：绘制视图本身（每个View需要重载，Viewgroup不需要实现该方法）；

DrawChild：重新回调每个子视图的绘制方法；

Invalidate：只调用View的onDraw方法不调用onMeasure和onLayout方法。

3.invalidate和postInvalidate的区别及使用

Invalidate主线程调用View的重绘过程

postInvalidate子线程调用View的重绘过程

4.Activity-Window-View三者的差别

Acivity通过attach方法创建了一个Window的实现类PhoneWindow，当Activity调用onCreate方法中的setContentView方法时，创建了DecorView，并将指定的布局文件填充进DecorView，调用到Viewgroup的removeAllView方法将所有的view移除，再调用addView添加view。

5.低版本SDK如何实现高版本api？

引入对应的support包，或者判断版本做对应实现。

6.描述一次网络请求的流程

1. HttpUrlConnection 和 okhttp关系

HttpUrlConnection是google提供的用来访问网络的方式，底层是Socket实现的，支持Http1.0和1.1；Okhttp与HttpUrlConnection同级，底层也是Socket实现的，但是它支持Http2.0，网络访问的速度更快支持spdy、多路复用和连接池。

8.Bitmap对象的理解

9.looper架构

Handler，Looper和MessageQueue：在初始化Looper的时候会初始化一个MessageQueue并调用Looper.loop方法进行循环调用MessageQueue的next方法获取下消息，初始化Handler的时候，初始化方法会通过Looper.getMyLooper方法获取同线程下存在的Looper对象，这样Handler就和Looper还有MessageQueue关联起来了，在handler调用post或者send方法时，调用MessageQueue的equeueMessage方法将Message消息存入消息队列，然后一直在循环的loop方法通过next方法取出消息，调用msg.target.dispatchMessage方法，这里的msg.target就是对应的Handler，最后分发给对应的Handler的dispatchMessage方法处理，该方法中会根据判断调用对应的callback、handleMessage、或者run方法。

10.ActivityThread，AMS，WMS的工作原理

11.自定义View如何考虑机型适配

使用密度无关像素单位，布局时使用百分比，使用.9图，针对不同布局使用多套资源文件，切图时尽量使用大分辨切图。

12.自定义View的事件

13.AsyncTask+HttpClient 与 AsyncHttpClient有什么区别？

一个是异步任务与网络请求方式的结合，另一个是网络访问框架。

1. LaunchMode应用场景

四种启动模式：

Standrad：一般模式，创建就存入栈中。

singleTop：创建实例后，如果顶部也是同样的Activity的话，则移除顶部，在入栈。如果顶部不一致，不管任务栈是否存在，直接创建入栈。

singleTask：创建实例后，如果不存在同样实例，则直接入栈，如果存在同样实例，则移除同样实例及其后入栈的实例，再将该实例进行入栈

singleInstance：创建实例后即创建新的栈将该实例存入栈，以类似singleTask的方式移除所有栈中重复实例。

1. AsyncTask 如何使用?

需在主线程初始化。内部有一个Handler和两个线程池组成（一个用来进行工作线程的排队，一个是用来执行工作线程）。

使用的主要是四个方法：

onPreExecute：在主线程中执行，执行异步任务前的准备工作。

doInBackground：在工作线程中执行，用来执行耗时任务。

onProgressUpdate：在主线程中执行，用于更新任务。

onPostExecute:在主线程中执行，用于执行完工作线程后处理逻辑。

16.SpareArray原理

17.请介绍下ContentProvider 是如何实现数据共享的？

利用Android的Binder机制实现的数据共享；使用流程：在需要被调用的进程中实现ContentProvider，继承ContentProvider实现它的六个方法，分别是onCreate、getType、insert、delete、query、update方法，默认还要创建UriMatcher实例用来筛选匹配获取数据的命令的，内部连接该进程中需要的数据提供者，可以是数据库或者私有文件，在执行增删改查方法中需匹配UriMatcher的值，增删改查操作执行完毕后需调用，才能进行对应的操作，再在该进程对应的Androidmanifest中注册该ContentProvider，设置对应的android：authorities属性进行调用认证匹配，还可设置读写权限分离进行数据保护，然后在客户端初始化一个ContentResolver进行远程增删改查，还可以在客户端设置自定义ContentObserver重新onChange方法进行监听数据更新回调。

18.Android Service与Activity之间通信的几种方式

方法一：通过Binder机制的ServiceConnection的回调方法onServiceConnected中获取对应的Service实例，通过这个实例设置一个回调方法；

方法二：在对应的Activity中注册一个广播，在Service内部调用广播发送，进行数据传输。

19.IntentService原理及作用是什么？

原理利用HandlerThread创建了一个属于该线程的Handler，在主线程中调用了该Handler类的send方法将Activity的Intent传递给对Handler所在的线程即子线程再调用onHandleIntent做耗时操作。作用是直接提供了一个运行在后台线程的Service类。

20.说说Activity、Intent、Service 是什么关系

21Application Context和Activity Context的区别

都是Context的子类，对应的Application的Context的生命周期是应用的生命周期，而Activity对应的生命周期是Activity的生命周期；一般关于UI的创建需使用Activity的context，非UI的创建可使用Activity、Service或者Application等的context。

1. SP是进程同步的吗?有什么方法做到同步？

不是进程同步的，可通过Contentprovider包装成做到同步。

23.谈谈多线程在Android中的使用

Handler、AsyncTask、IntentService

24.进程和 Application 的生命周期

25.封装View的时候怎么知道view的大小

在View的measure过程中确认大小，通过父布局的Measurespec mode和自身的Measurespec mode确认测量大小，最终在onLayout中确认最终长宽。

26.RecycleView原理

27.AndroidManifest的作用与理解

（三）常见的一些原理性问题

1.Handler机制和底层实现

Handler、Looper、MessageQueue、Message四者构成handler机制，handler初始化之初会调用Looper.getMyLooper获取当前线程对应的Looper，在Looper初始化之初会初始一个MeassgeQueue消息队列，此时Handler、Looper、和MessageQueue就相互关联了，当调用Handler的post或者send方法时，handler调用相关联的MessageQueue的enQueueMessage方法将Message消息存入消息队列，而looper在初始化后调用loop方法开启消息循环，调用MessageQueue的next方法获取消息，一旦有消息在队列中，立即取出消息再调用msg.target的dispatchMessage方法进行消息分发，msg.target就是对应的handler实例，即最终调用了Handler的dispatchMessage分发消息，调用对应的callback，run或者handleMessage方法。

2.Handler、Thread和HandlerThread的差别

Handler结合Thread解决应用在子线程无法主线程UI的问题，HandlerThread是一个能创建对应Handler的线程。

1. handler发消息给子线程，looper怎么启动？

该Handler创建于子线程，对应的子线程需要自己创建对应线程的Looper，Looper.prepare

Loop.loop方法构建对应的消息循环体。

1. 关于Handler，在任何地方new Handler 都是什么线程下?

在任何线程new Handler只要存在对应线程的Looper对象，就可以创建对应的线程的Handler。

1. ThreadLocal原理，实现及如何保证Local属性？

将数据存储于对应线程的table变量中，利用Thread引用找到对应Thread的存储ThreadLocal数据的table变量。

1. 请解释下在单线程模型中Message、Handler、Message Queue、Looper之间的关系
2. 请描述一下View事件传递分发机制

Activity对点击事件的分发管理：Activity的dispatchTouchEvent进行事件分发，派发到子View PhoneWindow，PhoneWindow再调用dispatchTouchEvent进行事件分发，再传递到DecorView，DecorView在调用dispatchTouchEvent进行事件分发到setContentView中设置的顶端Viewgroup。

Viewgroup的事件分发：调用dispatchTouchEvent进行事件分发，再调用自身的onInterceptTouchEvent方法，返回true并且disAllowIntercept为false，自己消化点击事件不进行子View事件分发，返回false或者disAllowIntercept为true，调用在点击区域内子view进行事件分发。

View的事件分发：调用自身的dispatchTouchEvent方法，再查看是否设置了onTouch方法，如果设置的onTouch方法并且返回true，并且对应控件是可点击的，则点击事件就此被消化，如果以上两个条件有一个不满足，则事件继续传递，传递到对应View的onTouchEvent方法中消化点击事件，里面如果有onClick事件则会消化事件。

1. Touch事件传递流程

同上

9.事件分发中的onTouch 和onTouchEvent 有什么区别，又该如何使用？

onTouch优先级大于onTouchEvent，当View的调用事件分发方法dispatchTouchEvent方法时，会先判断是否设置onTouchListener，如果设置了该接口，且其回调函数返回true，且disallowIntercept为false，那事件就被onTouch方法消化，否则事件分发才能进入onTouchEvent方法中。

10.View和ViewGroup分别有哪些事件分发相关的回调方法

View dispatchTouchEvent onTouch onTouchEvent、onClick方法

ViewGroup dispatchTouchEvent onInterceptTouchEvent方法

1. View刷新机制

Invalidate层层往上调用到ViewRootImpl的scheduleTraversals方法对对应的view的draw过程重绘,requestLayout是层层往上调到ViewRootImpl的scheduleTraversals方法通过该方法最终调用performTraversals进行遍历子元素进行measure、layout过程。

1. View绘制流程
2. 自定义控件原理

继承View或Viewgroup实现对应的measure，layout，draw过程。

1. 自定义View如何提供获取View属性的接口？

在style文件中添加declare-styleable定义属性列表，在View的构造方法中，通过obtainStyledAttributes(attrs,对应的style文件)获取TypeArray实例，在通过TypeArray的get\*\*\*\*(R.styleable.属性名称，对应的子属性);再在引用该自定义控件的布局文件中引用命名空间，使用对应属性。

15.Android代码中实现WAP方式联网

1. AsyncTask机制

两个线程池，一个handler实现的异步任务，四个方法分别是onPreExecute，doInBackGroud，onProgressUpdate，onPostExecute方法。

1. AsyncTask原理及不足

Android多版本串并行不一致、非静态内部类容易存在内存泄露问题、一个AsynTask只用execute一次，调用cancel方法取消AsyncTask任务，如果doInBackground方法已调用，则需等待doInBackGround方法执行结束，只是最终不执行onPostExecute方法。

1. 如何取消AsyncTask？

调用AsyncTask的cancel(true)方法结束异步任务，只是能保证不调用onPostExecute方法，不能保证doInBackground执行与否。

19.为什么不能在子线程更新UI？

因为UI线程是非同步的，子线程更新UI会造成未知的界面错乱，这样设计的原因保证页面刷新的及时性。

1. ANR产生的原因是什么？

主线程执行耗时任务，长时间无响应产生ANR异常。

21.ANR定位和修正

利用/data/anr/traces.txt文件定位修正。

1. oom是什么？

内存溢出异常。

1. 什么情况导致oom？

使用内存大于系统分配给应用的内存。

1. 有什么解决方法可以避免OOM

使用到加载Bitmap的地方要利用inSampleSize或者解码格式减小加载图片时使用的内存，

避免在android中使用枚举类，

使用更轻量的数据结构如SparseArray，

避免在onDraw方法中声明临时变量，因为绘制过程中onDraw方法会反复被调用，

等等。

1. Oom 是否可以try catch？为什么？

可以try catch，先释放内存再用使用更小的内存即可。

1. 内存泄漏是什么？

本应该释放的内存，因为某种原因无法得到释放。

1. 什么情况导致内存泄漏？

静态变量导致的内存泄露：例在Activity中存在Context类型的静态变量sContext，引用到Activity的context，在Activity销毁时，因为静态变量保留其引用导致无法资源无法得到释放。

单例模式存在的内存泄露：例单例模式中的static 实例保留Activitiy传入的context变量，当Activity销毁时，也会引起内存泄露。

属性动画开启循环动画，未调用动画停止方法时会发生内存泄露。

1. 如何防止线程的内存泄漏？

采用静态内部类防止内存泄露，减少

29.内存泄露场的解决方法

30.内存泄漏和内存溢出区别？

31.LruCache默认缓存大小

32.ContentProvider的权限管理(解答：读写分离，权限控制-精确到表级，URL控制)

33.如何通过广播拦截和abort一条短信？

34.广播是否可以请求网络？

35.广播引起anr的时间限制是多少？

36.计算一个view的嵌套层级

37.Activity栈

38.Android线程有没有上限？

39.线程池有没有上限？

40.ListView重用的是什么？

41.Android为什么引入Parcelable？

42.有没有尝试简化Parcelable的使用？

（四）开发中常见的一些问题

1.ListView 中图片错位的问题是如何产生的?

2.混合开发有了解吗？

3.知道哪些混合开发的方式？说出它们的优缺点和各自使用场景？（解答：比如:RN，weex，H5，小程序，WPA等。做Android的了解一些前端js等还是很有好处的)；

4.屏幕适配的处理技巧都有哪些?

5.服务器只提供数据接收接口，在多线程或多进程条件下，如何保证数据的有序到达？

6.动态布局的理解

7.怎么去除重复代码？

8.画出 Android 的大体架构图

9.Recycleview和ListView的区别

10.ListView图片加载错乱的原理和解决方案

11.动态权限适配方案，权限组的概念

12.Android系统为什么会设计ContentProvider？

13.下拉状态栏是不是影响activity的生命周期

14.如果在onStop的时候做了网络请求，onResume的时候怎么恢复？

15.Bitmap 使用时候注意什么？

16Bitmap的recycler()

17.Android中开启摄像头的主要步骤

18.ViewPager使用细节，如何设置成每次只初始化当前的Fragment，其他的不初始化？

19.点击事件被拦截，但是想传到下面的View，如何操作？

20.微信主页面的实现方式

21.微信上消息小红点的原理