## Android常见原理性专题

**1.Handler机制和底层实现**

**2.Handler、Thread和HandlerThread的差别**

1）Handler线程的消息通讯的桥梁，主要用来发送消息及处理消息。

2）Thread普通线程，如果需要有自己的消息队列，需要调用Looper.prepare()创建Looper实例，调用loop()去循环消息。

3）HandlerThread是一个带有Looper的线程，在HandleThread的run()方法中调用了Looper.prepare()创建了Looper实例，并调用Looper.loop()开启了Loop循环，循环从消息队列中获取消息并交由Handler处理。利用该线程的Looper创建Handler实例，此Handler的handleMessage()方法是运行在子线程中的。即Handler利用哪个线程的Looper创建的实例，它就和相应的线程绑定到一起，处理该线程上的消息，它的handleMessage()方法就是在那个线程中运行的，无参构造默认是主线程。HandlerThread提供了quit()/quitSafely()方法退出HandlerThread的消息循环，它们分别调用Looper的quit和quitSafely方法，quit会将消息队列中的所有消息移除，而quitSafely会将消息队列所有延迟消息移除，非延迟消息派发出去让Handler去处理。

HandlerThread适合处理本地IO读写操作（读写数据库或文件），因为本地IO操作耗时不长，对于单线程+异步队列不会产生较大阻塞，而网络操作相对比较耗时，容易阻塞后面的请求，因此HandlerThread不适合加入网络操作。

--handler发消息给子线程，looper怎么启动？

--关于Handler，在任何地方new Handler 都是什么线程下?

--ThreadLocal原理，实现及如何保证Local属性？

--请解释下在单线程模型中Message、Handler、Message Queue、Looper之间的关系

--请描述一下View事件传递分发机制

<https://blog.csdn.net/shengfeixiang/article/details/30751459>

--Touch事件传递流程

<https://www.jianshu.com/p/c41153cb6211>

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

<https://blog.csdn.net/qq_26358311/article/details/79620964>

--View和ViewGroup分别有哪些事件分发相关的回调方法

--View刷新机制

--View绘制流程

--自定义控件原理

--自定义View如何提供获取View属性的接口？

<https://www.2cto.com/kf/201801/714417.html>

--Android代码中实现WAP方式联网

<https://blog.csdn.net/asce1885/article/details/7844159>

--AsyncTask机制

--AsyncTask原理及不足

--如何取消AsyncTask？

--为什么不能在子线程更新UI？

https://blog.csdn.net/haoyuegongzi/article/details/79414081

--ANR产生的原因是什么？

--ANR定位和修正

--oom是什么？

（oom(Out Of Memory)内存溢出）

--什么情况导致oom？

--有什么解决方法可以避免OOM？

--Oom 是否可以try catch？为什么？

（可以，当）

--内存泄漏是什么？

内存泄露就是指该被GC垃圾回收的，但被一个生命周期比它长的对象仍然在引用它，导致无法回收，造成内存泄露，过多的内存泄露会导致OOM。

--什么情况导致内存泄漏？

1）非静态内部类、匿名内部类：非静态内部类、匿名内部类 都会持有外部类的一个引用，如果有一个静态变量引用了非静态内部类或者匿名内部类，导致非静态内部类或者匿名内部类的生命周期比外部类（Activity）长，就会导致外部类在该被回收的时候，无法被回收掉，引起内存泄露, 除非外部类被卸载。

解决办法：将非静态内部类、匿名内部类 改成静态内部类，或者直接抽离成一个外部类。 如果在静态内部类中，需要引用外部类对象，那么可以将这个引用封装在一个WeakReference中。

2）静态的View：当一个Activity经常启动，但是对应的View读取非常耗时，我们可以通过静态View变量来保持对该Activity的rootView引用。这样就可以不用每次启动Activity都去读取并渲染View了。但View attach到我们的Window上，就会持有一个Context(即Activity)的引用。而我们的View有事一个静态变量，所以导致Activity不被回收。

解决办法： 在使用静态View时，需要确保在资源回收时，将静态View detach掉。

3）Handler：在Activity中定义Handler对象，那么Handler持有Activty的引用。而每个Message对象是持有Handler的引用的（Message对象的target属性持有Handler引用），从而导致Message间接引用到了Activity。如果在Activty destroy之后，消息队列中还有Message对象，Activty是不会被回收的。

解决办法： 将Handler放入单独的类或者将Handler放入到静态内部类中（静态内部类不会持有外部类的引用）。如果想要在handler内部去调用所在的外部类Activity，可以在handler内部使用弱引用的方式指向所在Activity，在onDestory时，调用相应的方法移除回调和删除消息。

4）监听器（各种需要注册的Listener，Watcher等）：当我们需要使用系统服务时，比如执行某些后台任务、为硬件访问提供接口等等系统服务。我们需要把自己注册到服务的监听器中。然而，这会让服务持有 activity 的引用，如果程序员忘记在 activity 销毁时取消注册，那就会导致 activity 泄漏了。

解决办法：在onDestory中移除注册

5）. 资源对象没关闭造成内存泄漏：当我们打开资源时，一般都会使用缓存。比如读写文件资源、打开数据库资源、使用Bitmap资源等等。当我们不再使用时，应该关闭它们，使得缓存内存区域及时回收。

解决办法：使用try finally结合，在try块中打开资源，在finally中关闭资源

6）. 属性动画：在使用ValueAnimator或者ObjectAnimator时，如果没有及时做cancel取消动画，就可能造成内存泄露。因为在cancel方法里，最后调用了endAnimation(); ，在endAnimation里，有个AnimationHandler的单例，会持有属性动画对象的引用。

解决办法：在onDestory中调用动画的cancel方法

7）. RxJava：在使用RxJava时，如果在发布了一个订阅后，由于没有及时取消，导致Activity/Fragment无法销毁，导致的内存泄露。

解决办法：使用RxLifeCycle

--内存泄漏和内存溢出区别？

（1）内存泄漏

1）内存泄漏：指程序中已动态分配的堆内存由于某种原因未释放或无法释放，造成系统内存的浪费，导致程序运行速度减慢甚至系统奔溃等严重后果。

2）一次内存泄漏似乎不会有大的影响，但内存泄漏后堆积的结果就是内存溢出。

3）内存泄漏具有隐蔽性，积累性的特征，比其他内存非法访问错误更难检测。这是因为内存泄漏产生的原因是内存块未被释放，属于遗漏型缺陷而不是过错型缺陷。此外，内存泄漏不会直接产生可观察的错误，而是逐渐积累，降低系统的整体性性能。

4）如何有效的进行内存分配和释放，防止内存泄漏，是软件开发人员的关键问题，比如一个服务器应用软件要长时间服务多个客户端，若存在内存泄漏，则会逐渐堆积，导致一系列严重后果。

(2)内存溢出

指程序在申请内存时，没有足够的内存供申请者使用，或者说，给了你一块存储int类型数据的存储空间，但是你却存储long类型的数据，就会导致内存不够用，报错OOM，即出现内存溢出的错误。

--LruCache默认缓存大小

（4MB）

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

--如何通过广播拦截和abort一条短信？

<https://www.cnblogs.com/chenxibobo/p/6136689.html>

--广播是否可以请求网络？

--广播引起anr的时间限制是多少？

10s

--计算一个view的嵌套层级

<https://blog.csdn.net/zx_android/article/details/79558509>

--Activity栈

--Android线程有没有上限？

<https://bbs.csdn.net/topics/380189354>

--线程池有没有上限？

<https://www.jianshu.com/p/d2729853c4da>

--ListView重用的是什么？

--Android为什么引入Parcelable？

<https://www.jianshu.com/p/df35baa91541>

--有没有尝试简化Parcelable的使用？

<https://blog.csdn.net/u013265344/article/details/87979948>

<https://www.cnblogs.com/ldq2016/p/9035666.html>