总结内存泄露的相关知识点，以面试题的形式进行列举。

链接：https://blog.csdn.net/feather_wch/article/details/78289202

内存泄露汇总

版本：2018/8/10-1

内存泄露

1、内存泄露是什么?

> 1. 分配的内存在使用后并没有释放，而导致内存一直被占用，直到结束
> 2. 危害：持续的内存泄漏会导致可用内存越来越少，最终导致内存溢出(out of memory)

2、内存溢出是什么?

> 1. `OOM(Out of Memory)`，app所占的 堆内存 超过了系统规定的最大值 `HeapSize` 就会出现 内存溢出 导致应用崩溃

3、强引用(Strong reference)是什么

> `A a = new A()`， 强引用 本身存在 栈内存 中(其指向的 A对象 存储在堆中)，不会被 GC回收 ， GC回收 的是 堆内存 中的数据。

4、软引用

> 1. 不具有 强引用 ， 当 虚拟机内存不足 时，会被回收。
> 2. 适合做 网页缓存、图片缓存 等。

```
A a = new A();
SoftReference<A> srA = new SoftReference<A>(a);
```

5、弱引用

> 1. 无论内存是否充足 ， 只具有 弱引用 的对象都会立即回收。

```
A a = new A();
WeakReference<A> wrA = new WeakReference<A>(a);
```

6、虚引用(Phantom Reference)

> 1. 只有虚引用的对象 任何时候都会被回收。

7、强、软、弱、虚引用总结图

引用类型	GC回收时间	用途	生存时间
强引用	never	对象的一般状态	JVM停止运行时
软引用	内存不足时	对象缓存	内存不足时终止
弱引用	GC时	对象缓存	GC后终止
虚引用	unknow	unknow	unknow

8、查找内存泄露的几个关键点

> 1. app 上限制的 heapsize 是多少。
> 2. 什么情况会导致 无法GC
> 3. 怎么复现哪个界面 内存泄露

9、如何知道 app上限值 的 heapSize

```
//1. 获取`ActivityMnager`
ActivityManager manager = (ActivityManager) getSystemService(Context.ACTIVITY_SERVICE);
//2. 获取heapSize
int heapSize = manager.getMemoryClass();
//3. 最大的heapSize
int maxHeapSize = manager.getLargeMemoryClass(); //AndroidManifest.xml中 android:largeHeap="tru
```

> android:largeHeap="true" 能将 heapSize 增大两到三倍，但并不是本质的解决 OOM 的方法

10、什么情况会导致无法GC

> 1. 对象依旧持有引用，因此不会被GC

11、如何判断哪个界面导致内存泄露？如何定位？

> 1. 利用 LeakCanary
> 2. 通过 Android Studio 3.0 的 Profiler

12、如何避免内存泄露

> 生命周期 比 Activity 长的类不要去 强引用 Activity

内存泄露常见场景(9种)

13、内存泄露常见场景有哪些

> 1. 非静态内部类
> 2. 静态类
> 3. 静态变量
> 4. WebView
> 5. HandlerThread
> 6. Bitmap
> 7. 集合
> 8. 动画
> 9. 资源释放

非静态内部类

14、非静态的内部类会持有外部类的隐式引用

> 1. 非静态内部类、匿名类：非静态的匿名类会持有外部类的一个隐式引用。
> 2. 如 Adpater、Handler和Runnable 等会持有 Activity 的引用)
> 3. 解决办法： 内部类 请用 static 变成静态内部类

15、Handler导致的内存泄露

1. Runnable对象 持有了 Activity 的强引用。
2. Handler 持有 Activity 的强引用。

```
//静态内部类：Handler
private static class MyHandler extends Handler{
    //1-弱引用持有Activity，可以被GC回收
    private WeakReference<ImageViewActivity> mActivityWeakReference;
    //2-Handler初始化，持有Activity弱引用
    public MyHandler(ImageViewActivity activity){
        mActivityWeakReference = new WeakReference<ImageViewActivity>(activity);
    }
    //3-用activity去调用一些业务逻辑的方法
    @Override
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        //4-获取到Activity，执行业务逻辑
        ImageViewActivity imageViewActivity = mActivityWeakReference.get();
        if(imageViewActivity != null){
            imageViewActivity.doSth();
        }
    }
}
private static Runnable myRunnable = new Runnable() {
    @Override
    public void run() {
        //TODO 业务逻辑
    }
};


//Activity中使用
MyHandler myHandler = new MyHandler(this);
myHandler.post(myRunnable);
```

在 OnDestory() 中调用 handler.removeCallbacksAndMessages(null); 清除掉 Handler 中残留的任务，能提前 GC 的时间，而不是 任务 都执行完毕后，才进行 GC

静态类

16、静态类持有 Activity 的引用

1. 禁止 静态类如Application、单例类、其他static类 持有 Activity 的引用。
2. 或者可以持有 Activity 的弱引用。

17、单例模式

1. 单例模式的对象中的 链表或者成员变量 错误的持有了 如：Activity的this指针 ， 却没有及时释放会导致泄露，因为单例的特点是其生命周期和 Applicaiton 一致。
2. 解决办法：避免该种场景，或者采用 弱引用 来解决该问题。

静态变量

18、静态变量导致内存泄露

> 如: Activity内部静态变量持有Activity的this等

WebView

19、WebView导致内存泄露的原因和解决办法

> 1. `AccessibilityStateChangeListener` 最终会持有 `WebView.mContext` ,也就是会持有 `Activity` 导致Activity无法释放。
> 2. 该监听器是在 `ViewRootImpl` 的初始化中进行注册，在 `dispatchDetachedFromeWindow` 中进行反注册。
> 3. 本质 `WebView`的`onDetahcedFromWindow` 并未正常执行完成，从而导致 `ViewRootImpl` 的 `dispatchDetachedFromeWindow` 不会执行。

```
if (mWebView != null) {
//1. WebView destory前将其从`父容器`中移除
   ViewParent parent = mWebView.getParent();
   if (parent != null) {
      ((ViewGroup) parent).removeView(mWebView);
   }
//2. WebView调用onDestory()方法
   mWebView.removeAllViews();
   mWebView.destroy();
   mWebView = null;
}
```

Bitmap

20、Bitmap导致的内存泄露

> 1. `Bitmap` 没有用 `recycle()` 释放内存

HandlerThread

21、HandlerThread导致的内存泄露

> . 关联 `Activity` 的 `HandlerThread` 生命周期过长，需要在 `Activity`销毁方法 中调用 `thread.getLooper().quit()`

动画

22、无限循环动画导致内存泄露

> 1. 属性动画 不停止会导致内存泄漏:

> 2. 无限循环动画会持有 `Activity`的`View` ， 而 `View` 持有了 `Activity` ， 最终导致泄露。
> 3. 解决办法是: 在 `onDestory` 中调用 `animator.cancel()`

资源释放

23、资源没有释放导致内存泄露

> 1. IO
> 2. Socket
> 3. 数据库
> 4. EventBus没有解除注册。
> 5. 广播没有解除注册
> 6. ...

集合

24、集合导致的内存泄露

> 1. 静态集合持有对象的引用，导致引用无法释放，引起内存泄露。
> 2. 解决办法： 在页面的 `onDestory` 中对集合进行清空。

工具

25、内存泄露检查常用工具

> 1. LeakCanary：参考资料有教程
> 2. Android Studio Porfile
> 3. MAT
> 4. adb shell的dumpsys指令

26、Android内存调试工具：dumpsys

> dumpsys：是android自带的调试工具，能够查看内存的使用情况
> 1、通过 `adb shell dumpsys meminfo <packageName>` 来查看内存使用状况
> 2、打开Activity，运行指令后，可以看到 `View` 和 `Activity` 的数量
> 3、关闭Activity，运行指令就可以看到关闭Acitivity后 `View` 和 `Activity` 的数量
> 4、这样就可以简单判断出Acitivity是否出现内存泄漏

学习和参考资料

1. LeakCanary使用教程
2. （Android Studio 3.0）Android Profiler内存泄漏检查
3. Android开发中常见的内存泄露案例

4. WebView内存泄露解析
5. Android内存泄露
6. 译文：Android可能内存泄露的8种场景
7. 手把手教你在Android Studio 3.0上分析内存泄漏

额外收获

1. `Layout Inspector` 已经替代了 `Hierarchy Viewer`： 用于查看布局层级