MC 的技术空间

eMail：hust_mc@outlook.com 主页：http://www.jianshu.com/users/a0d978b30696/latest_articles

Android工程师面试题大全

标签： android java 面试题 工程师 java基础

2015-10-30 22:14 2455人阅读 评论(5) 收藏 举报

分类： Android点滴（25）

目录(?) [+]

校招的日子结束了，结果也算圆满。忙碌了一阵子，现在终于可以安安静静的做做项目看看书写写论文了。下面对这段时间面试遇到的问题并结合网上各位的大神秒下的面试题做个总结，本文会持续更新，希望能在面试中助各位一臂之力！

Java基础：

1、内存泄露的原因：

- 资源对象没关闭。
  如Cursor、File等资源。他们会在finalize中关闭，但这样效率太低。容易造成内存泄露。
  SQLiteCursor，当数据量大的时候容易泄露
- 使用Adapter时，没有使用系统缓存的converView。
- 即时调用recycle（）释放不再使用的Bitmap。
  适当降低Bitmap的采样率，如：

```
BitmapFactory.Options options = newBitmapFactory.Options();
options.inSampleSize = 2;//图片宽高都为原来的二分之一，即图片为原来的四分之一
Bitmap bitmap =BitmapFactory.decodeStream(cr.openInputStream(uri), null, opt
```

- 使用application的context来替代activity相关的context。
  尽量避免activity的context在自己的范围外被使用，这样会导致activity无法释放。
- 注册没取消造成内存泄露
  如：广播
- 集合中的对象没清理造成的内存泄露我们通常把一些对象的引用加入到了集合中，当我们不需要该对象时，并没有把它的引用从集合中清理掉，这样这个集合就会越来越大。如果这个集合是static的话，那情况就更严重了。
- Handler应该申明为静态对象， 并在其内部类中保存一个对外部类的弱引用。如下：

```
static class MyHandler extends Handler
{
       WeakReference<Activity > mActivityReference;
       MyHandler(Activity activity)
      {
            mActivityReference= new WeakReference<Activity>(activity);
       }
     @Override
     public void handleMessage(Message msg)
    {
           final Activity activity = mActivityReference.get();
           if (activity != null)
          {
```

个人资料

超低空MC

访问： 40916次
积分： 779
等级： BLOG 3
排名： 千里之外

原创： 24篇
转载： 13篇
译文： 0篇
评论： 184条

最新评论

MP3歌词的同步与拖拽设计
超低空MC: @h1217256980:希望对你有用，有什么问题欢迎讨论

```
                    mImageView.setImageBitmap(mBitmap);
            }
        }
}
```

2、ArrayList和LinkedList的区别

- ArrayList初试大小为10，大小不够会调用grow扩容：length = length + (length >> 1)
- LinkedList中Node first,last。分别指向头尾

ArrayList和LinkedList在性能上各 有优缺点，都有各自所适用的地方，总的说来可以描述如下：

1. 对ArrayList和LinkedList而言，在列表末尾增加一个元素所花的开销都是固定的。对 ArrayList而言，主要是在内部数组中增加一项，指向所添加的元素，偶尔可能会导致对数组重新进行分配；而对LinkedList而言，这个开销是统一的，分配一个内部Entry对象。
2. 在ArrayList的 中间插入或删除一个元素意味着这个列表中剩余的元素都会被移动；而在LinkedList的中间插入或删除一个元素的开销是固定的。
3. LinkedList不 支持高效的随机元素访问。
4. ArrayList的空 间浪费主要体现在在list列表的结尾预留一定的容量空间，而LinkedList的空间花费则体现在它的每一个元素都需要消耗相当的空间

可以这样说：当操作是在一列 数据的后面添加数据而不是在前面或中间,并且需要随机地访问其中的元素时,使用ArrayList会提供比较好的性能；当你的操作是在一列数据的前面或中 间添加或删除数据,并且按照顺序访问其中的元素时,就应该使用LinkedList了。

3、hashmap和hashtable的不同

- 继承不同。

```
public class Hashtable extends Dictionary implements Map
public class HashMap extends AbstractMap implements Map
```

- Hashtable 中的方法是同步的，而HashMap中的方法在缺省情况下是非同步的。在多线程并发的环境下，可以直接使用Hashtable，但是要使用HashMap的话就要自己增加同步处理了。
- Hashtable中，key和value都不允许出现null值。
  在HashMap中，null可以作为键，这样的键只有一个；可以有一个或多个键所对应的值为null。当get()方法返回null值时，即可以表示 HashMap中没有该键，也可以表示该键所对应的值为null。因此，在HashMap中不能由get()方法来判断HashMap中是否存在某个键， 而应该用containsKey()方法来判断。
- 两个遍历方式的内部实现上不同。
  Hashtable、HashMap都使用了 Iterator。而由于历史原因，Hashtable还使用了Enumeration的方式 。
- 哈希值的使用不同，HashTable直接使用对象的hashCode。而HashMap重新计算hash值。
- Hashtable和HashMap它们两个内部实现方式的数组的初始大小和扩容的方式。HashTable中hash数组默认大小是11，增加的方式是 old*2+1。HashMap中hash数组的默认大小是16，而且一定是2的指数。

4、Iterator和Enumeration的不同

1. 函数接口不同
   Enumeration只有2个函数接口。通过Enumeration，我们只能读取集合的数据，而不能对数据进行修改。
   Iterator只有3个函数接口。Iterator除了能读取集合的数据之外，也能数据进行删除操作。
2. Iterator支持fail-fast机制，而Enumeration不支持。 Enumeration 是JDK 1.0添加的接口。使用到它的函数包括Vector、Hashtable等类，这些类都是JDK 1.0中加入的，Enumeration存在的目的就是为它们提供遍历接口。Enumeration本身并没有支持同步，而在Vector、Hashtable实现Enumeration时，添加了同步。而Iterator 是JDK 1.2才添加的接口，它也是为了HashMap、ArrayList等集合提供遍历接口。Iterator是支持fail-fast机制的：当多个线程对同一个集合的内容进行操作时，就可能会产生fail-fast事件。

ail-fast 机制是Java集合(Collection)中的一种错误机制。当多个线程对同一个集合的内容进行操作时，就可能会产生fail-fast事件。例如：当某一个线程A通过iterator去遍历某集合的过程中，若该集合的内容被其他线程所改变了；那么线程A访问集合时，就会抛出ConcurrentModificationException异常，产生fail-fast事件。

5、接口的注意点

1. 接口中的字段全部默认为 public static类型。
2. 接口中的方法全部默认为 public类型。

3. 接口中可以申明内部类，而默认为public static，正因为是static，只是命名空间属于接口，代码逻辑不属于接口。所以不违法接口定义。
4. 接口本身可以申明为public或者缺省。
5. 抽象类继承自某接口。如果在抽象类中实现了父类（接口）中的方法，在其子类可以不用实现，否则在子类必须实现。

6、final方法

将方法声明为final那有两个原因，第一就是说明你已经知道这个方法提供的功能已经满足你要求，不需要进行扩展，并且也不允许任何从此类继承的类来覆写这个方法，但是继承仍然可以继承这个方法，也就是说可以直接使用。第二就是允许编译器将所有对此方法的调用转化为inline调用的机制，它会使你在调用final方法时，直接将方法主体插入到调用处，而不是进行例行的方法调用，例如保存断点，压栈等，这样可能会使你的程序效率有所提高，然而当你的方法主体非常庞大时，或你在多处调用此方法，那么你的调用主体代码便会迅速膨胀，可能反而会影响效率，所以你要慎用final进行方法定义。

Android知识点

1、Handler机制

- Handler对Activity finish影响。
  在开发的过程中碰到一个棘手的问题，调用Activity.finish函数Acitivity没有执行生命周期的ondestory函数，后面查找半天是因为有一个handler成员，因为它有一个delay消息没有处理，调用Activity.finish，Activity不会马上destory，所以记得在Ativity finish前清理一下handle中的未处理的消息，这样Activity才会顺利的destory
- Looper
  通过调用Looper.prepare()创建Looper()对象并绑定到ThreadLocal变量中。
  Looper里面包含了messageQueue。
  构造器如下：

```
private Looper()
{
        mQueue = new MessageQueue();
        mRun = true;
        mThread = Thread.currentThread();
}
```

- loop()函数
  1）从Looper中取出MessageQueue；
  2）循环从MessageQueue中取出Message；
  3）从Message中取出Target（Handler对象）；
  4）调用tartget的dispatchMessage分发消息。
- Handler对象
  重要成员变量：

```
final MessageQueue mQueue;
final Looper  mLooper;
final Callback mCallback;   //用于回调
```

Handler对象在发送消息的时候，将MSG的target变量设为自己。这样在Looper对象循环取出msg的时候就可以调用对应handler的dispatchMessage()。此函数分发消息的优先级如下：
1. Message在创建的时候调用Obtain设置了Callback。
2. Handler在创建的时候传入了Callback。
3. 交给Handler子类的HandleMessage处理（通常的做法）。

2、Android启动模式

standard和singleTop模式。
这两种比较简单。创建Activity放入当前的任务栈中，若当前是singleInstace，则放入设置的任务栈中。其中如果Activity在栈顶，则调用onNewIntent。

singletask：栈内复用模式。不是在当前任务栈中查找是否存在，实际过程如下：

1. 查找该Activity所需的任务栈是否存在（由taskAffinity控制，或者默认为包名）。
2. 在任务栈当中查找该Activity是否存在。
这里面存在任务栈的切换，也就是当开启的singtask类型的Activity不属于当前任务栈时，则会切换到其任务栈。

singleInstance：单实例模式。
包含了singleTask的所有特性，另外加上：设置为该模式的Activity，只能单独存在于一个任务栈中。当有两个singleInstace的Activity设置成同样的任务栈时，会出现两个同名的任务栈，分别用来存放同名的Activity。
注：在任何跳转的时候，首先调用本Activity的onPause，然后跳转。如果被跳转的activity由于启动方式而没创建新的实例，则会先调用onNewIntent，然后按照正常的生命周期调用。
如
1：A→B，A：onPause；B：onCreate，onStart，onResume。
2：A(singleTop)→A，A：onPause；A：onSaveInstanceState；A：onResume。

3、View的绘制

推荐郭霖大神的博客：
http://blog.csdn.net/guolin_blog/article/details/16330267

4、canvas的使用

推荐以下博客：
http://blog.csdn.net/qinjuning/article/details/6936783

5、ActivityManagerService的相关知识点

推荐以下博客：
http://wiki.jikexueyuan.com/project/deep-android-v2/activity.html

---

6、Activity切换时生命周期交集

Activity之间的协作当一个activity A启动了另外一个activity B，它们的生命周期是有交叉的；
首先A的onPause()被调用；
之后B的onCrate(), onStart()及onResume() 方法会被调用（此时B拥有用户焦点）；
最后，如果A在屏幕上不可见，onStop()方法被调用；
因此，我们在两个activities中传递数据，或者共享资源时（如**数据库**连接），需要在前一个activity的onPause()方法而不是onStop()方法中进行；

7、Hybrid（重要加分项）

1. java和JS的交互
   http://droidyue.com/blog/2014/09/20/interaction-between-java-and-javascript-in-android/
   http://rensanning.iteye.com/blog/2043049
2. WebView开启JavaScript脚本执行
3. WebView设置供JavaScript调用的交互接口。

8、网络编程

1. volley
   https://bxbxbai.github.io/2014/09/14/android-working-with-volley/
   http://blog.csdn.net/guolin_blog/article/details/17656437
2. 如何控制TCP连接时的拥塞
   http://blog.csdn.net/yechaodechuntian/article/details/25429143
3. 三次握手
   http://blog.csdn.net/whuslei/article/details/6667471
4. Android客户端和服务端如何使用Token和Session
   http://wyong.blog.51cto.com/1115465/1553352

5. 移动端获取网络数据优化的几个点
    1. 连接复用 :
       节省连接建立时间，如开启 keep-alive。
       对于 Android 来说默认情况下 HttpURLConnection 和 HttpClient 都开启了 keep-alive。只是 2.2 之前 HttpURLConnection 存在影响连接池的 Bug，具体可见：Android HttpURLConnection 及 HttpClient 选择
    2. 请求合并:
       即将多个请求合并为一个进行请求，比较常见的就是网页中的 CSS Image Sprites。如果某个页面内请求过多，也可以考虑做一定的请求合并。
    3. 减少请求数据的大小：
       对于post请求，body可以做gzip压缩的，header也可以作数据压缩(不过只支持http 2.0)。
    4. 返回的数据的body也可以作gzip压缩，body数据体积可以缩小到原来的30%左右。(也可以考虑压缩返回的json数据的key数据的体积，尤其是针对返回数据格式变化不大的情况，支付宝聊天返回的数据用到了)
    5. 根据用户的当前的网络质量来判断下载什么质量的图片(电商用的比较多)。

9、android开发中，可能会导致内存泄露的问题

1. 不要让生命周期长于Activity的对象持有到Activity的引用
2. 尽量使用Application的Context而不是Activity的Context
3. 尽量不要在Activity中使用非静态内部类，因为非静态内部类会隐式持有外部类实例的引用（具体可以查看细话Java：”失效”的private修饰符了解）。如果使用静态内部类，将外部实例引用作为弱引用持有。
4. 垃圾回收不能解决内存泄露，了解Android中垃圾回收机制
   **更多内容可以参考以下博客：
   http://spencer-dev.lofter.com/post/d7b9e_6faf120

10、activity的启动过程：

http://www.cloudchou.com/android/post-788.html

以上是我遇到和搜集到的各类题目以及相应的解答，接下来一段时间也会持续更新，大家遇到什么经典或者不会的问题也可以给我留言，在此统一分享给大家。祝大家能够找到自己心仪的工作，前途一片光明！

上一篇　比spinner更轻量的下拉菜单——DropDownList控件
下一篇　难忘的面试周——百度校招面经

我的同类文章

Android点滴（25）

- MP3歌词的同步与拖拽设计 2016-05-30 阅读 4049
- Android真正的静默安装 2016-01-29 阅读 1156
- 浅谈android线程池 2015-12-29 阅读 137
- 难忘的面试周——百度校招… 2015-11-08 阅读 1448
- 比spinner更轻量的下拉菜单… 2015-10-23 阅读 494
- Java多态性的理解 2015-09-15 阅读 443
- 深入理解java异常处理机制 2015-09-07 阅读 402
- Integer.valueOf(String) 方法… 2015-07-30 阅读 265
- 《Android第一行代码》笔记 2015-06-03 阅读 555
- Android action使用大全 2015-06-01 阅读 404

更多文章

参考知识库

JavaScript知识库
2752 关注 | 773 收录

Android知识库
13735 关注 | 1525 收录

查看评论

4楼 dengnen86 2015-12-10 21:57发表

还没找到工的一个菜菜android程序员留下一个脚印

Re: 超低空MC 2015-12-11 10:03发表

回复dengnen86：加油！

3楼 Fndroid 2015-11-05 19:32发表

马克

2楼 申屠晨涛 2015-11-04 13:04发表

收藏

1楼 Griezmann 2015-10-30 22:46发表

收藏了 留着也许用得上

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场