22·安卓的系统架构（概率40%）

[http://wenku.baidu.com/view/3ef50eddd15abe23482f4d7e.html](http://wenku.baidu.com/view/3ef50eddd15abe23482f4d7e.html" \t "http://blog.csdn.net/dangai0201/article/details/_blank)

**应用层**：使用Java语言进行开发的一些应用程序

**应用框架层**：主要是谷歌发布的一些操作支持的类库（API框架），开发人员可以使用这些类库方便的进行程序开发，但是在开发时必须遵守框架的开发原则

**系统运行库层**：当使用安卓框架层进行开发时，安卓操作系统会自动使用一些c/c++的库文件来支持所使用的各个组件，使其可以更好的为程序服务；

**Linux内核层**：安卓操作系统主要是基于Linux内核，程序的安全性，驱动程序，进程管理等都由Linux内核提供。

18·配文件中存放什么？（概率60%）

版本号、权限，四大组件，意图，第三方key等

7.activity的生命周期（概率80%）

onCreate，onStart，onResume，onPause，onStop，onDestroy；onRestart;(6+1)

44.Activity的启动模式（四种）（概率100%）

Standard(每次创建一个新的activity)、singleTop()、singleTask、singleInstance

**standard**

默认模式，可以不用写配置。在这个模式下，都会默认创建一个新的实例。因此，在这种模式下，可以有多个相同的实例，也允许多个相同Activity叠加。

**singleTop**

可以有多个实例，但是不允许多个相同Activity叠加。即，如果Activity在栈顶+的时候，启动相同的Activity，不会创建新的实例，而会调用其onNewIntent方法。

**singleTask**

只有一个实例。在同一个应用程序中启动他的时候，若Activity不存在，则会在当前task创建一个新的实例，若存在，则会把task中在其之上的其它Activity destory掉并调用它的onNewIntent方法。

如果是在别的应用程序中启动它，则会新建一个task，并在该task中启动这个Activity，singleTask允许别的Activity与其在一个task中共存，也就是说，如果我在这个singleTask的实例中再打开新的Activity，这个新的Activity还是会在singleTask的实例的task中。

**singleInstance**

只有一个实例，并且这个实例独立运行在一个task中，这个task只有这个实例，不允许有别的Activity存在。

37·AIDL的全称是什么，如何工作（概率20%）

AIDL的英文全称是Android Interface Define Language

当A进程要去调用B进程中的service时，并实现通信，我们通常都是通过AIDL来操作的

A工程：

首先我们在net.blogjava.mobile.aidlservice包中创建一个RemoteService.aidl文件，在里面我们自定义一个接口，含有方法get。ADT插件会在gen目录下自动生成一个RemoteService.java文件，该类中含有一个名为RemoteService.stub的内部类，该内部类中含有aidl文件接口的get方法。

说明一：aidl文件的位置不固定，可以任意

然后定义自己的MyService类，在MyService类中自定义一个内部类去继承RemoteService.stub这个内部类，实现get方法。在onBind方法中返回这个内部类的对象，系统会自动将这个对象封装成IBinder对象，传递给他的调用者。

其次需要在AndroidManifest.xml文件中配置MyService类，代码如下：

<!-- 注册服务 -->

<service android:name=".MyService">

<intent-filter>

<!-- 指定调用AIDL服务的ID -->

<action android:name="net.blogjava.mobile.aidlservice.RemoteService" />

</intent-filter>

</service>

为什么要指定调用AIDL服务的ID,就是要告诉外界MyService这个类能够被别的进程访问，只要别的进程知道这个ID，正是有了这个ID,B工程才能找到A工程实现通信。

说明：AIDL并不需要权限

B工程：

首先我们要将A工程中生成的RemoteService.java文件拷贝到B工程中，在bindService方法中绑定aidl服务

绑定AIDL服务就是将RemoteService的ID作为intent的action参数。

说明：如果我们单独将RemoteService.aidl文件放在一个包里，那个在我们将gen目录下的该包拷贝到B工程中。如果我们将RemoteService.aidl文件和我们的其他类存放在一起，那么我们在B工程中就要建立相应的包，以保证RmoteService.java文件的报名正确，我们不能修改RemoteService.java文件

bindService(new Inten("net.blogjava.mobile.aidlservice.RemoteService"), serviceConnection, Context.BIND\_AUTO\_CREATE);

ServiceConnection的onServiceConnected(ComponentName name, IBinder service)方法中的service参数就是A工程中MyService类中继承了RemoteService.stub类的内部类的对象。

Android采用RPC的方式来实现（remote procedure call）远程通信，并且Android通过接口定义语言AIDL来生成两个进程之间互相访问的代码。例如，你在Activity里的代码需要访问Service中的一个方法，那么就可以通过这种方法来实现了。

?创建一个AIDL文件（参见代码）

?实现AIDL文件生成的Java接口（参见代码）

?将你的接口暴露给客户端

?客户端调用

45.广播机制的作用（概率40%）

广播机制，本质上它就是一种组件间的通信方式，如果是两个组件位于不同的进程当中，那么可以用Binder机制来实现，如果两个组件是在同一个进程中，那么它们之间可以用来通信的方式就更多了，这样看来，广播机制似乎是多余的。然而，广播机制却是不可替代的，它和Binder机制不一样的地方在于，广播的发送者和接收者事先是不需要知道对方的存在的，这样带来的好处便是，系统的各个组件可以松耦合地组织在一起，这样系统就具有高度的可扩展性，容易与其它系统进行集成。

实现思路：

在第1步中，广播的接收者把广播接收器注册到ActivityManagerService中；在第2步中，广播的发送者同样是把广播发送到ActivityManagerService中，由ActivityManagerService去查找注册了这个广播的接收者，然后把广播分发给它们。

在第2步的分发的过程，其实就是把这个广播转换成一个消息，然后放入到接收器所在的线程消息队列中去，最后就可以在消息循环中调用接收器的onReceive函数了。这里有一个要非常注意的地方是，由于ActivityManagerService把这个广播放进接收器所在的线程消息队列后，就返回了，它不关心这个消息什么时候会被处理，因此，对广播的处理是异步的，即调用sendBroadcast时，这个函数不会等待这个广播被处理完后才返回

实例连接：http://wenku.baidu.com/view/b73dd968af1ffc4ffe47ac0c.html

33·注册广播的方式（概率30%）

android中，不同进程之间传递信息要用到广播，可以有两种方式来实现。

第一种方式：在Manifest.xml中注册广播，是一种比较推荐的方法，因为它不需要手动注销广播（如果广播未注销，程序退出时可能会出错）。

具体实现在Manifest的application中添加：

<receiver android:name=".mEvtReceiver">

<intent-filter>

<action android:name="android.intent.action.BOOT\_COMPLETED" />

</intent-filter>

</receiver>

上面两个android:name分别是广播名和广播的动作（这里的动作是表示系统启动完成），如果要自己发送一个广播，在代码中为：

Intent i = new Intent("android.intent.action.BOOT\_COMPLETED");

sendBroadcast(i);

这样，广播就发出去了，然后是接收。

接收可以新建一个类，继承至BroadcastReceiver，也可以建一个BroadcastReceiver的实例，然后得写onReceive方法，实现如下：

protected BroadcastReceiver mEvtReceiver = new BroadcastReceiver() {

@Override

public void onReceive(Context context, Intent intent) {

String action = intent.getAction();

if (action.equals("android.intent.action.BOOT\_COMPLETED")) {

//Do something

}

}

};

第二种方式，直接在代码中实现，但需要手动注册注销，实现如下：

IntentFilter filter = new IntentFilter();

filter.addAction("android.intent.action.BOOT\_COMPLETED");

registerReceiver(mEvtReceiver, filter); //这时注册了一个recevier ,名为mEvtReceiver，然后同样用上面的方法以重写onReceiver，

最后在程序的onDestroy中要注销广播，实现如下：

@Override

public void onDestroy() {

super.onDestroy();

unregisterReceiver(mPlayerEvtReceiver);

}

Android系统中的广播是广泛用于应用程序之间通信的一种手段，它类似于事件处理机制，不同的地方就是广播的处理是系统级别的事件处理过程（一般事件处理是控件级别的）。在此过程中仍然是离不开Intent对象，理解广播事件的处理过程，灵活运用广播处理机制，在关键之处往往能实现特别的效果，

在Android 中如果要发送一个广播必须使用sendBroadCast 向系统发送对其感兴趣的广播接收器中。

使用广播必须要有一个intent 对象必设置其action动作对象

使用广播必须在配置文件中显式的指明该广播对象

每次接收广播都会重新生成一个接收广播的对象

在BroadCast 中尽量不要处理太多逻辑问题，建议复杂的逻辑交给Activity 或者 Service 去处理

36·ContentProvider的URI的配置？（概率20%）

清单文件之指定URI或者代码里面指定URI，contentProvider通过URI访问数据

10.framgment生命周期？（概率30%）

MyFragment onAttach() 粘贴到activity上

MyFragment onCreate() fragment创建

MyFragment onCreateView() fragment创建自己的视图

MainActivity onCreate()

MyFragment onActivityCreated() 可以处理fragment数据的初始化

MainActivity onStart()

MyFragment onStart()

MainActivity onResume()

MyFragment onResume()

按后退键

MyFragment onPause()

MainActivity onPause()

MyFragment onStop()

MainActivity onStop()

MyFragment onDestoryView() 销毁掉自己的视图

MyFragment onDestory()

MyFragment onDetach() 解除和activity的关系

MainActivity onDestory()

1.常用的存储方式有哪些？（概率50%）

1: Shared Preferences(**SP存储**)(以键值对的形式存储私有的,简单数据类型)

2: Internal Storage(**内存存储**)(私有的数据存储在设备的内存里面)

3: External Storage(**外部存储**)(将公共的数据存储在SD卡里面)

4: SQLite Databases(**数据库存储**)(将数据存储在私有的数据库里面)

(数据库存储包括SQLite和ContentProvider,其实ContentProvider的底层就是在操作数据库)

5: Network Connection(**网络存储**)(存储数据在web端,在你自己的数据服务器上)

2.安卓中动画有几类，他们的特点和区别？（概率80%）

1: Property Animation(**属性动画**别加入在3.0)

2: View Animation (**补间动画**分为:透明,旋转,位移,缩放)

注意:补间动画是假动画,点击区域不变.

3: Drawable Animation(**帧动画**帧动画类似于放电影，通过播放已经排列放好的图片来实现。)

3.handler机制原理（概率100%）

为什么引进？UI控件不是线程安全的,子线程不能更新UI,所以安卓引入了Handler机制,

为什么系统不对UI控件的访问加锁机制,缺点有两个

1、加上锁机制会让UI访问的逻辑变的复杂,

2、其次锁机制会降低UI访问的效率,因为锁机制会阻塞某些线程的执行,

Handler主要用于线程间的通信。

一个Handler允许发送和处理Message和Runable对象，

UI主线程会自动分配一个**Looper**（消息轮询器），每个Looper中封装着**MessageQueue**（消息队列），遵循先进先出原则。Looper负责不断的从自己的消息队列里取出队头的任务或消息执行。

一般是在子线程执行完耗时操作之后，通过Handler的sendMessage或post方法将Message和Runable对象传递给MessageQueue，而且在这些对象离开MessageQueue时，Handler负责执行他们（用到handleMessage方法，主要执行刷新UI的代码）。

其中Message类就是定义了一个信息，这个信息中包含一个描述符和任意的数据对象，这个信息被用来传递给Handler.Message对象提供额外的两个int域和一个Object域。

4除了handler和子线程，还有一个处理线程的是什么，主要方法是什么？（概率60%）

**AsynTask**

doInBackground() 这个方法运行在后台线程中，主要负责执行那些很耗时的操作，如访问网络。该方法必须重写。

onPostExecute(Result) 这个方法运行于UI主线程，在doInBackground(Params…)方法执行后调用，该方法用于接收后台任务执行后返回的结果，刷新UI显示

6线程之间的通信方式（如何避免ANR）（概率70%）

（1）. AsyncTask

（2.）子线程 + handler

5.tcp和udp区别.（概率80%）

**TCP---(长链接)(管发管收,数据安全)**传输控制协议,提供的是面向连接、可靠的字节流服务,传输数据前经过“**三次握手**”建立连接，保证数据传输的可靠性，但效率比较低。一般用于对于**数据传输安全性较高**的场合。

25·简述音乐播放的方式与区别？（概率20%）

1）：MediaPlayer：主要用于播放音频，可以播放视频，但是一般不用其进行视频播放。

2）：SoundPool：主要用于播放一些短促的声音片段，主要优势是cpu资源占有量低，反应延迟小。

**UDP---(短连接,只管发,不管收)**用户数据报协议，是一个简单的面向数据报的运输层协议，面向无连接。UDP不提供可靠性，数据传输可能发生错序，丢包，但效率较高。一般用于对于**实时性（聊天）要求较高**的场合。

26·工作中如何利用线程池的？（概率60%）

优势:线程池会缓存一定数量的线程,通过线程池就可以避免因为频繁创建和销毁线程所带来的系统开销(一般会根据cpu支持的线程最大数获取线程数,取的4分之1,一般情况下只有一个APP在前端运行,没什么问题)

使用场景:比如说同时要下载多张图片,这个时候需要开线程,但是如果图片数量太多,每张图片图片都需要一个线程,那么就会造成cpu的销毁,所以这个时候就需要使用线程池来指定最大线程数量,其余需要下载的图片在线程池中等等下载.

Java通过Executors提供四种线程池，分别为：

newCachedThreadPool创建一个可缓存线程池，如果线程池长度超过处理需要，可灵活回收空闲线程，若无可回收，则新建线程。

newFixedThreadPool 创建一个定长线程池，可控制线程最大并发数，超出的线程会在队列中等待。

newScheduledThreadPool 创建一个定长线程池，支持定时及周期性任务执行。

newSingleThreadExecutor 创建一个单线程化的线程池，它只会用唯一的工作线程来执行任务，保证所有任务按照指定顺序(FIFO, LIFO, 优先级)执行。

42.移动互联数据的交换格式 （概率20%）

移动互联数据交互格式有XML和JSON

1.JSON和XML的数据可读性基本相同

2.JSON和XML同样拥有丰富的解析手段

3.JSON相对于XML来讲，数据的体积小

4.JSON与JavaScript的交互更加方便

5.JSON对数据的描述性比XML较差

6.JSON的速度要远远快于XML

json解析方式:手动解析(JSONOBJ和JSONArray)自动解析方式有:gson

XML:解析方式:dom,sax,pull:

Dom:一次性全部加载进内容里面进行解析,不能中途停止

Sax:基于事件驱动性,一行一行解析,不能中途停止,

Pull:基于事件驱动下,一行一行解析,可以中途停止

40·安卓自定义组件？（概率100%）

Android自定义组件有三种实现思路:

1) 继承某个现有组件，在其基础上添加额外功能,如继承Gallery实现CoverFlow效果

2) 复合型组件定义: 继承某个Layout，实现复合组件自定义，如TextView和EditText组合实现登录注册组件

3) 继承View，实现onDraw()方法，实现自己绘制组件，如翻页效果组件

14·MVC的使用？（概率90%）

mvc是**model,view,controller**的缩写，mvc包含三个部分：(逻辑,控制,展示分开)

l模型（model）对象：是应用程序的主体部分，所有的业务逻辑都应该写在该层。

l视图（view）对象：是应用程序中负责生成用户界面的部分。也是在整个mvc架构中用户唯一可以看到的一层，接收用户的输入，显示处理结果。

l控制器（control）对象：是根据用户的输入，控制用户界面数据显示及更新model对象状态的部分

android鼓励弱耦合和组件的重用，在android中mvc的具体体现如下：

1)模型层（model）：对数据库的操作、对网络等的操作都应该在model里面处理，当然对业务计算等操作也是必须放在的该层的。

2)视图层（view）：一般采用xml文件进行界面的描述

3)控制层（controller）：android的控制层的acitvity

20·NDK是什么？（概率50%）

NDK 是 Native Development Kit 的简称。它是**一个工具集，集成了 Android 的交叉编译环境**，并提供了一套比较方便的 Makefile ，可以帮助开发者快速开发 C 或是 C++ 的动态库，并自动的将 so 和 java 程序打包成 apk ，在 Android 上运行。Android 上，应用程序的开发，大部分基于 Java 语言来实现。要使用 c 或是 c++ 的程序或库，就需要使用 NDK 来实现。

29·SIM卡的EF文件有何作用？（概率10%）

SIM卡的文件系统有自己的规范，主要是为了和手机通讯，SIM本身可以有自己的操作系统，EF就是作 存储并和手机通讯用的。

30.最长的短信多长，多少个字节？（概率10%）

70个汉字，140个字节

23.listView优化（概率60%）

1)、**convertView复用**，对convetView进行判空，当convertView不为空时重复使用，为空则初始化，从而减少了很多不必要的View的创建

2)**定义一个ViewHolder**，封装Listview Item条目中所有的组件，将convetView的tag设置为ViewHolder,不为空时通过ViewHolder的属性获取对应组件即可

3)、当ListView加载数据量较大时可以采用分页加载和图片异步加载

41.listview分页加载的步骤（概率20%）

通常实现分页加载有两种方式，一种是在ListView底部设置一个按钮，用户点

击即加载。另一种是当用户滑动到底部时自动加载。

在ListView底部设置一个按钮，用户点击即加载实现思路:

// 加上底部View，注意要放在setAdapter方法前

ListView.addFooterView(moreView);

bt.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

pg.setVisibility(View.VISIBLE);// 将进度条可见

bt.setVisibility(View.GONE);// 按钮不可见

handler.postDelayed(new Runnable() {

@Override

public void run() {

loadMoreDate();// 加载更多数据

bt.setVisibility(View.VISIBLE);

pg.setVisibility(View.GONE);

mSimpleAdapter.notifyDataSetChanged();// 通知listView刷新数据

}

}, 2000);

}

});

当用户滑动到底部时自动加载实现思路: 实现OnScrollListener 接口重写

onScrollStateChanged 和onScroll方法，使用onscroll方法实现”滑动“后处理检查是否还有新的记录，如果有,添加记录到adapter, adapter调用 notifyDataSetChanged 更新数据;如果没有记录了，则不再加载数据。使用onScrollStateChanged可以检测是否滚到最后一行且停止滚动然后执行加载.

50.listview失去焦点怎么处理（概率20%）

在listview子布局里面写，可以解决焦点失去的问题

android:descendantFocusability="blocksDescendants"

46.图片轮播的实现思路（概率30%）

图片轮播是由viewpage和handler两者结合所实现的，

11·图片异步加载怎么做？（概率80%）

三级缓存:

1:首先从强引用中获取(Lrucache),如果强引用中没有,再去软引用中获取(SoftReference),如果软引用再去SD卡获取(SD卡被Dis Lrucache替代掉),如果这三级缓存中都没有,就开启网络去请求图片,请求回来的图片,在加入到强引用中(Lrucache),如果强引用已经满了,这时候,强引用通过Lrucache算法,删除掉一些最近不常用的图片,这时候咱们把这些图片放到软引用里面,(当内存不足到时候软引用就会被被垃圾回收机制回收),如果垃圾回收机制没有,软引用也满了,将不常用的图片移除掉,同时加入到SD卡,(现在被Dis Lrucache替代)

(key 一般使用的是网络请求图片的url,来保证唯一性,一般情况下我会做一下MD5加密,这样做的目的是为了保证不会出现非法字符)

现在SoftReference已经被替代掉,现在我们只使用两级缓存,在加网络, Lrucache, Dis Lrucache

LRC算法:最近最少使用的(内部如何实现的)

强引用: Lrucache

软引用: SoftReference

弱引用:weak(随时被垃圾回收机制回收)

虚引用:phantomreference

可定义图片异步加载工具类，核心方式实现思路如下：

1.先从内存缓存（Map<String,SoftReference<Bitmap>>中获取图片显示

2.获取不到的话从本地SD卡里获取并显示

3.都获取不到的话通过子线程从网络加载图片并保存到内存及SD卡中并通过handler显示

9.安卓内存的优化？（概率90%）

[http://blog.csdn.net/awangyunke/article/details/20380719](http://blog.csdn.net/awangyunke/article/details/20380719" \t "http://blog.csdn.net/dangai0201/article/details/_blank)

handler内存泄露（thread， AsyncTask)

[http://www.linuxidc.com/Linux/2013-12/94065.htm](http://www.linuxidc.com/Linux/2013-12/94065.htm" \t "http://blog.csdn.net/dangai0201/article/details/_blank)

context内存泄露

[http://blog.sina.com.cn/s/blog\_5da93c8f0102w86x.html](http://blog.sina.com.cn/s/blog_5da93c8f0102w86x.html" \t "http://blog.csdn.net/dangai0201/article/details/_blank)

Bitmap内存泄露

1）静态变量引起内存泄露(内存泄露多了就变成溢出)

在代码优化的过程中，我们需要对代码中的静态变量特别留意。静态变量是类相关的变量，它的生命周期是从这个类被声明，到这个类彻底被垃圾回收器回收才会被销毁。所以，一般情况下，静态变量从所在的类被使用开始就要一直占用着内存空间，直到程序退出。如果不注意，静态变量引用了占用大量内存的资源，造成垃圾回收器无法对内存进行回收，就可能造成内存的浪费

2）使用Application(单例模式)的Context

在Android中，Application Context的生命周期和应用的生命周期一样长，而不是取决于某个Activity的生命周期。如果想保持一个长期生命的对象，并且这个对象需要一个 Context，就可以使用Application对象。可以通过调用Context.getApplicationContext()方法或者 Activity.getApplication()方法来获得Application对象。

3）及时关闭资源

Cursor是Android查询数据后得到的一个管理数据集合的类。正常情况下，如 果我们没有关闭它，系统会在回收它时进行关闭，但是这样的效率特别低。如果查询得到的数据量较小时还好，如果Cursor的数据量非常大，特别是如果里面 有Blob信息时，就可能出现内存问题。所以一定要及时关闭Cursor。

4）使用Bitmap及时调用recycle() // 把 重复循环的方法设置 为 null，释放内存

前面的章节讲过，在不使用Bitmap对象时，需要调用recycle()释放内存，然后将它设置为null。虽然调用recycle()并不能保证立即释放占用的内存，但是可以加速Bitmap的内存的释放。

在代码优化的过程中，如果发现某个Activity用到了Bitmap对象，却没有显式的调用recycle()释放内存，则需要分析代码逻辑，增加相关代码，在不再使用Bitmap以后调用recycle()释放内存。

5）对Adapter进行优化

下面以构造ListView的BaseAdapter为例说明如何对Adapter进行优化。

@软引用和弱引用。

如果一个对象只具有软引用，那么如果内存空间足够，垃圾回收器就不会回收它；如果内存空间不足了，就会回收这些对象的内存。只要垃圾回收器没有回收它，该对象就可以被程序使用。软引用可用来实现内存敏感的高速缓存。软引用可以和一个引用队列（ReferenceQueue）联合使用，如果软引用所引用的对象被垃圾回收，Java虚拟机就会把这个软引用加入到与之关联的引用队列中。

如果一个对象只具有弱引用，那么在垃圾回收器线程扫描的过程中，一旦发现了只具有弱引用的对象，不管当前内存空间足够与否，都会回收它的内存。不过，由于垃圾回收器是一个优先级很低的线程，因此不一定会很快发现那些只具有弱引用的对象。弱引用也可以和一个引用队列（ReferenceQueue）联合使用，如果弱引用所引用的对象被垃圾回收，Java虚拟机就会把这个弱引用加入到与之关联 的引用队列中。

弱引用与软引用的根本区别在于：只具有弱引用的对象拥有更短暂的生命周期，可能随时被回收。而只具有软引用的对象只有当内存不够的时候才被回收，在内存足够的时候，通常不被回收。

6）UI优化

在Android应用开发过程中，屏幕上控件的布局代码和程序的逻辑代码通常是分开 的。界面的布局代码是放在一个独立的xml文件中的，这个文件里面是树型组织的，控制着页面的布局。通常，在这个页面中会用到很多控件，控件会用到很多的 资源。Android系统本身有很多的资源，包括各种各样的字符串、图片、动画、样式和布局等等，这些都可以在应用程序中直接使用。这样做的好处很多，既 可以减少内存的使用，又可以减少部分工作量，也可以缩减程序安装包的大小。

13·内存泄露如何解决？（概率90%）

[http://wenku.baidu.com/view/03dc4b46b307e87100f69604.html](http://wenku.baidu.com/view/03dc4b46b307e87100f69604.html" \t "http://blog.csdn.net/dangai0201/article/details/_blank)

1）、 数据库的cursor没有关闭

2）、 构造adapter没有使用缓存contentview

Listview:性能优化:

1: Listview是需要和适配器配合使用的,常用的适配器有arrayAdapter,SimplerAdapter, SimpleCursorAdapter,BaseAdapter,一般情况下我使用的是继承BaseAdapter,实现它的四个方法,其中最重要的两个方法,getCount,和getView,在不做任何处理的情况下getview每次都重新生成一个view,这样的结果就是很消耗内存,谷歌官网提供了优化listview内存的方法,通过ConvertView的复用和使用ViewHolder来减少findbyid的次数,从而达到控件的复用和减少cpu的消耗,但是因为控件的复用,在加载图片的时候,复用的控件里面有残留的图片,从而造成图片的错位,所以我们必须在初始化的时候设置默认图片来清除缓存的图片,同时在加载的时候图片的时候,如果不进行listview的性能优化,就会造成OOM,所以一般情况下我们会加载二次采样过后的url,同时我们会进行三级缓存的处理,(同上)

二次采样:

[http://m.blog.csdn.net/article/details?id=51245133](http://m.blog.csdn.net/article/details?id=51245133" \t "http://blog.csdn.net/dangai0201/article/details/_blank)

既然是二次采样，那当然要分为两步了，下面我们来说说每次采样的主要工作：

1.第一次采样

第一次采样我主要是想要获得图片的压缩比例，假如说我有一张图片是200\*200，那么我想把这张图片的缩略图显示在一个50\*50的ImageView上，那我的压缩比例应该为4，那么这个4应该怎么样来获得呢？这就是我们第一步的操作了，我先加载图片的边界到内存中，这个加载操作并不会耗费多少内存，加载到内存之后，我就可以获得这张图片的宽高参数，然后根据图片的宽高，再结合控件的宽高计算出缩放比例。

2.第二次采样

在第一次采样的基础上，我来进行二次采样。二次采样的时候，我把第一次bab采样后算出来的结果作为一个参数传递给第BitmapFactory，这样在加载图片的时候系统就不会将整张图片加载进来了，而是只会加载该图片的一张缩略图进来，这样不仅提高了加载速率，而且也极大的节省了内存，而且对于用户来说，他也不会有视觉上的差异

3.代码实现

第一次采样:首先获取new BitmapFactory.Options()获取option对象, inJustDecodeBounds设置为true只会加载图片的边框进来，并不会加载图片具体的像素点,通过BitmapFactory.decodeFile(filePath, options);来进行第一次加载图片,通过outWidth原图的宽度,通过outHeight获取原图的高,接下来通过图片的跨度和控件的高度来获取一个采样率sampleSize,接下来设置inJustDecodeBounds为false,同时设置缩放比,设置inPreferredConfig的图片的图片格式有(分别为ALPHA\_8，RGB\_565，ARGB\_4444，ARGB\_8888),最后进行图片的加载.

衍生的listview优化问题：减少创建View的对象，充分使用contentview，可以使用静态类来处理优化getView的过程 3、Bitmap对象不使用时采用recycle()释放内存 4、Activity中的对象生命周期大于Activity

）Bitmap对象不使用recycle（）释放内存

）Activity中的对象生命周期大于Activity；

内存泄露大多数都是由于相同的错误导致的，即：对Context拥有较长时间的引用。

避免Context相关的内存泄露，记住以下事情：

不要保留对Context-Activity长时间的引用（对Activity的引用的时候，必须确保拥有和Activity一样的生命周期）

尝试使用Context-Application来替代Context-Activity

如果你不想控制内部类的生命周期，应避免在Activity中使用非静态的内部类，而应该使用静态的内部类，并在其中创建一个对Activity的弱引用。这种情况的解决办法是使用一个静态的内部类，其中拥有对外部类的WeakReference，如同ViewRoot和它的Winner类那样

GC（垃圾回收）不能解决内存泄露问题

48.全局异常处理（概率50%）

[Android使用UncaughtExceptionHandler捕获全局异常](http://blog.csdn.net/hehe9737/article/details/7662123" \t "http://blog.csdn.net/dangai0201/article/details/_blank)（未捕获的异常处理程序）

Android系统的“程序异常退出”，给应用的用户体验造成不良影响。为了捕获应用运行时异常并给出友好提示，便可继承[UncaughtExceptionHandler](http://orgcent.com/tag/UncaughtExceptionHandler/" \t "http://blog.csdn.net/dangai0201/article/details/_blank)类来处理。通过Thread.setDefaultUncaughtExceptionHandler()方法将异常处理类设置到线程上即可。

Demo下载地址：[http://code.google.com/p/android-custom-view/downloads/list](http://code.google.com/p/android-custom-view/downloads/list" \t "http://blog.csdn.net/dangai0201/article/details/_blank)

38·屏幕适配？（概率100%）

ه最好可以通过权重(layout\_weight)的方式来分配每个组件的大小，也可以通过具体的像素(dip)来确定大小。

尽量使用Relativelayout 。

已知应用支持平台设备的分辨率,可以提供多个layout\_320\*480 ...

drawable-hdpi,drawable-mdpi,drawable-ldpi分别代表分辨率为480\*800,360\*480,240\*360, 放置图片大小相差1.5倍

最后还需要在AndroidManifest.xml里添加下面一段，没有这一段自适应就不能实现：

<supports-screens

android:largeScreens="true"

android:normalScreens="true"

android:anyDensity = "true"/>

在</application>标签和</manifest> 标签之间添加上面那段代码。即可。

备注：三者的解析度不一样，就像你把电脑的分辨率调低，图片会变大一样，反之分辨率高，图片缩小

还可以通过.9.png实现图片的自适应

39·版本更新?数据库的更新？跨版本？（概率30%）

// 应用升级( 版本号升级 ) 或软件升级，数据库就会跟着升级

应用升级==版本号升级 == 同时数据库也会 跟着升级

软件升级 == 同时数据库也会跟着升级

版本：

首先在启动页面,首先请求网络,获取服务器端版本号,同时获取本地APP的版本号,如果服务器的版本号高于本地的版本号,就开启线程去现在APP,同时安装APP,如果服务器的版本号和本地版本号相同,就不下载app,直接进去app.

在清单文件更改版本号，上传更新的应用到平台，利用平台的更新提醒用户是否更新新版本。

数据库更新：

如果数据库版本号大于当前版本号,这个时候就会执行onupgress方法,为指定的表增加字段或

先更改版本号，在onupgress方法内判断版本号是否一致，不一致则更新数据库。

43.xmpp的底层是怎样实现的(即时聊天)？（概率20%）

socket，双广播，连接广播和通知广播

21百度地图和谷歌地图的优缺点？（概率60%）

**优势：**

1一打开百度地图页面，显示的就是目前所在城市地图，谷歌地图不然。

2百度地图的界面，百度地图很多都是在显眼位置用文字做提示，一目了然，谷歌界面有点炫，很多用图标来表示，但是用起来相对复杂，不熟悉的用户需要确定这个图标是否是所需要选择的。

而百度地图搜索框下就直接有“公交”“驾车”的选项，界面简洁直接很多，谷歌地图需要点击获取路线，再点击小图标来选择“公交”这些选项，而且图标不够引人注目，使用起来相对复杂些，试想一个很少用网络地图的人分别使用二者地图，那个更方便呢？应该是[百度](http://www.zhihujingxuan.com/baidu/" \t "http://blog.csdn.net/dangai0201/article/details/_blank)，大多数用户通过地图是查询位置-这个地方在哪，周围有什么地物，查询路线-从一个方到另一个地方如何到达，用百度地图，这些问题很容易解决，而用谷歌地图得摆弄一会儿。

3百度地图的查询路线更好,首先还是界面更简洁直接，然后是数据的问题，先不管为他们提供数据的公司如何，刚刚特意同时查了下，同样地起点和终点， 百度地图给出了5条路线，谷歌地图给出的是4条路线，而且百度地图路线的选择时可以选择“较快捷”“少换乘”“少步行”三种方式，更方便。打车费用也直接 显示出来，而且个人体会10元内的打车费用基本正确。

4百度地图有个视野内搜索功能，可以很方便地查找到目标位置，谷歌地图没有。这个很重要。

5一个重要界面布局问题，百度地图是地图视野在左，而谷歌地图是地图视野在右，我们现在时喜欢从左到右看东西啊，这点百度地图更符合用户习惯。（当然也可能是我常用百度地图，所以这么觉得）

PS:刚在知乎上有朋友说到——因“左图右列表”强调的是“地图”，反之强调的是“搜索”；我们发现用户在经过这么些年的教育后，“不在乎”地图在哪边，相对其它体验，这一点影响用户选择服务的权重并不高。

6百度地图已经和都市圈合作推出了部分城市如北京、上海、深圳、广州的三维地图，谷歌地图没有。

7百度地图地图视野上方有个截图功能，可以直接拉框截图，谷歌地图没有。

8以北京实时路况信息为例，直接输入北京查询，百度地图显示的是主城区的实时路况，而谷歌显示的区域很大，包括了密4云、廊坊市了，但是实时路况信息还是显示的主城区，显示路况的彩色条就全挤在一起了，设计得不如百度好。

8使用百度的占大多数，经常可以听到“百度一下啊”的话语。随之带来百度地图的使用率也相对较高。

9百度地图的地图百宝箱、房产、天气等信息。

**劣势：**

1如果需要从网络地图上获取数据的话，谷歌地图完胜，而且还有谷歌地球在一旁辅助，如虎添翼，从网上各种各样的谷歌地图相关工具软件就可以看出。

2路线查询时输入地点名称却有多个符合条件时，会给出一个下拉菜单给予选择，同一地点谷歌地图提供的数据精确到街道，而百度地图精确到区。

3谷歌地图可以将侧边栏隐藏起来，看到更大的地图视野，而百度地图是点击全屏来显示更大的视野，两种切换方式比起来谷歌的更舒服。

4谷歌地图有个分享链接功能，而且还提供“粘贴 HTML 以便嵌入到网站”这一功能，百度没有。

5界面布局百度地图更简洁，但是色彩搭配上和画面感上谷歌地图更好。

6谷歌的品牌形象占优。