**1.常用的存储方式有哪些？**

（五种，说出哪五种，五种存储方式什么情况下用。）注意sharepreferes对象支持读取不支持写入，写入引用Editor。

SQLite：

SQLite是一个轻量级的数据库，支持基本SQL语法，是常被采用的一种数据存储方式。Android为此数据库提供了一个名为SQLiteDatabase的类，封装了一些操作数据库的API。

SharedPreference：

除SQLite数据库外，另一种常用的数据存储方式，其本质就是一个xml文件，常用于存储较简单的参数设置。

File：

即常说的文件（I/O）存储方法，常用语存储大数量的数据，但是缺点是更新数据将是一件困难的事情。

ContentProvider:

Android 系统中能实现所有应用程序共享的一种数据存储方式，由于数据通常在各应用间的是互相私密的，所以此存储方式较少使用，但是其又是必不可少的一种存储方式。 例如音频，视频，图片和通讯录，一般都可以采用此种方式进行存储。每个ContentProvider都会对外提供一个公共的URI（包装成Uri对 象），如果应用程序有数据需要共享时，就需要使用ContentProvider为这些数据定义一个URI，然后其他的应用程序就通过 Content Provider传入这个URI来对数据进行操作。

网络存储：

从网络读取数据和写入数据。 Android提供了通过网络来实现数据的存储和获取的方法。

 我们可以调用WebService返回的数据或是解析HTTP协议实现网络数据交互。

**2.安卓中动画有几类，他们的特点和区别？**

   两种，一种是补间动画，还有一种是帧动画，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。

补间动画的实现定义开始和结束，中间由系统计算得出效果有透明，位移，放大缩小等等。

自定义录音或者进度条过程，会应用到帧动画，补间动画，一般用于应用的欢迎界面。

(菜单弹出，例如赞和评论，是个动画过程。)

**3.handler机制原理**

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。

字面以外加上关联点。

**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？**

AsynTask，doInbackGround+onPostExecute

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

**@5.tcp和udp区别.**

TCP---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。

UDP---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要求较高的场合。

**6线程之间的通信方式（如何避免ANR）**

（1）. AsyncTask，其中doInBackground()和onPostExecute(Result)两个方法非常重要

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。

（2.）子线程 + handler

在子线程中执行完耗时操作需要刷新UI时，通过handler.sendMessage()发消息给主线程， 然后在主线程Handler中的handleMessage()方法中执行刷新UI操作

1. **activity的生命周期**

**/**/ 开始 重新启动 停止 销毁 暂停

七个，oncreate，onstart，onrestart，onstop，onresume，ondestroy，onpause；（清楚整个周期的过程）

1）.启动Activity：系统会先调用onCreate方法，然后调用onStart方法，最后调用onResume，Activity进入运行状态。

2）.当前Activity被其他Activity覆盖其上或被锁屏：系统会调用onPause方法，暂停当前Activity的执行。

3）.当前Activity由被覆盖状态回到前台或解锁屏：系统会调用onResume方法，再次进入运行状态。

4）.当前Activity转到新的Activity界面或按Home键回到主屏，自身退居后台：系统会先调用onPause方法，然后调用onStop方法，进入停滞状态。

5）.用户后退回到此Activity：系统会先调用onRestart方法，然后调用onStart方法，最后调用onResume方法，再次进入运行状态。

6）.当前Activity处于被覆盖状态或者后台不可见状态，即第2步和第4步，系统内存不足，杀死当前Activity，而后用户退回当前Activity：再次调用onCreate方法、onStart方法、onResume方法，进入运行状态。

7）.用户退出当前Activity：系统先调用onPause方法，然后调用onStop方法，最后调用onDestory方法，结束当前Activity。

**8.ArrayList和LinkedList区别？**

存数据，ArrayList数组存储数据，索引值以下标来搜索，查询比较方，删除增加比较麻烦，但是linkedList以链表式存储数据，对于增删比较方便。

**9.安卓内存的优化？\***

1）静态变量引起内存泄露

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。

**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。

**10.framgment生命周期？**

oncreate，onstart，onrestart，onstop，onresume，ondestroy，onpause，onAttach，onCreateView，onDettach，onDestroyView;（和activity的生命周期比较，会更好）

 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()

**@11·图片异步加载怎么做？**

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

**@12·service和Thread区别？**

[http://wenku.baidu.com/link?url=p-P0sCqgr7y1w-mkd-B8DIwpIppzEud9MCVtttYWp1AWAmfZEfLyqKYKvAJWtXE2N2X8WS4vNHutJRBaG\_RAC9i1AwqY5Wh7Zh4s-XHixju](http://wenku.baidu.com/link?url=4OAMuQ6CYG6IjdyYH6y5zo9kX222dhotZo6sAmhVH7I-6inE8tZVWfs9US4njk2753sYEAK4zK9na5HxjTyKVzFcMQfl9GF5CLwbCLJUynu（详细介绍的路径）)

1）.Thread：Thread 是程序执行的最小单元，它是分配CPU的基本单位。可以用 Thread 来执行一些异步的操作。

2).Service：Service 是android的一种机制，当它运行的时候如果是Local Service，那么对应的 Service 是运行在主进程的 main 线程上的。如：onCreate，onStart 这些函数在被系统调用的时候都是在主进程的 main 线程上运行的。如果是Remote Service，那么对应的 Service 则是运行在独立进程的 main 线程上

**13·内存泄露如何解决？**

<http://wenku.baidu.com/view/03dc4b46b307e87100f69604.html>

1）、   数据库的cursor没有关闭

2）、 构造adapter没有使用缓存contentview

   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

**14·MVC的使用？**

mvc是model,view,controller的缩写，mvc包含三个部分：   
  
　　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，控制器更重要的一种导航功能，想用用户出发的相关事件，交给m哦得了处理。   
  
　　android鼓励弱耦合和组件的重用，在android中mvc的具体体现如下：   
  
　 　1)视图层（view）：一般采用xml文件进行界面的描述，使用的时候可以非常方便的引入，当然，如何你对android了解的比较的多了话，就一定 可以想到在android中也可以使用javascript+html等的方式作为view层，当然这里需要进行java和javascript之间的通 信，幸运的是，android提供了它们之间非常方便的通信实现。   
  
　　2)控制层（controller）：android的控制层的重 任通常落在了众多的acitvity的肩上，这句话也就暗含了不要在acitivity中写代码，要通过activity交割model业务逻辑层处理， 这样做的另外一个原因是android中的acitivity的响应时间是5s，如果耗时的操作放在这里，程序就很容易被回收掉。   
  
　　3)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。

**15·常用的设计模式**

**单例设计模式**

       所谓单例设计模式简单说就是无论程序如何运行，采用单例设计模式的类（Singleton类）永远只会有一个实例化对象产生。具体实现步骤如下：

      (1) 将采用单例设计模式的类的构造方法私有化（采用private修饰）。

      (2) 在其内部产生该类的实例化对象，并将其封装成private static类型。

      (3) 定义一个静态方法返回该类的实例。

**工厂设计模式**

       程序在接口和子类之间加入了一个过渡端，通过此过渡端可以动态取得实现了共同接口的子类实例化对象

**代理设计模式**

       指由一个代理主题来操作真实主题，真实主题执行具体的业务操作，而代理主题负责其他相关业务的处理。比如生活中的通过代理访问网络，客户通过网络代理连接网络（具体业务），由代理服务器完成用户权限和访问限制等与上网相关的其他操作（相关业务）。

**观察者设计模式**

       所谓观察者模式，举个例子现在许多购房者都密切观察者房价的变化，当房价变化时，所有购房者都能观察到，以上的购房者属于观察者，这便是观察者模式。

       java中可以借助Observable类和Observer接口轻松实现以上功能。当然此种模式的实现也不仅仅局限于采用这两个类。

**适配器模式**

        如果一个类要实现一个具有很多抽象方法的接口，但是本身只需要实现接口中的部分方法便可以达成目的，所以此时就需要一个中间的过渡类，但此过渡类又不希望 直接使用，所以将此类定义为抽象类最为合适，再让以后的子类直接继承该抽象类便可选择性的覆写所需要的方法，而此抽象类便是适配器类。

**16·面向对象的特征？**

**封装**

封装最好理解了。封装是面向对象的特征之一，是对象和类概念的主要特性。

封装，也就是把客观事物封装成抽象的类，并且类可以把自己的数据和方法只让可信的类或者对象操作，对不可信的进行信息隐藏。

**继承**

面向对象编程 (OOP) 语言的一个主要功能就是“继承”。继承是指这样一种能力：它可以使用现有类的所有功能，并在无需重新编写原来的类的情况下对这些功能进行扩展。

**多态**

多态性（polymorphisn）是允许你将父对象设置成为和一个或更多的他的子对象相等的技术，赋值之后，父对象就可以根据当前赋值给它的子对象的特性以不同的方式运作。简单的说，就是一句话：允许将子类类型的指针赋值给父类类型的指针。

实现多态，有二种方式，覆盖，重载。

覆盖，是指子类重新定义父类的虚函数的做法。

重载，是指允许存在多个同名函数，而这些函数的参数表不同（或许参数个数不同，或许参数类型不同，或许两者都不同）。

**17·程序结束的时候，如何处理Activity的？\***

记录打开的Activity：（**http://zdpeng.iteye.com/blog/1576055**）

每打开一个Activity，就记录下来。在需要退出时，关闭每一个Activity即可。

**18·配置文件中存放什么？**

四大组件，意图，权限，第三方key，版本号等

**19·Final ，finally，finalized，区别；**

final用于声明属性，方法和类，分别表示属性不可变，方法不可覆盖，类不

可继承。

finally是异常处理语句结构的一部分，表示总是执行。

finalize是Object类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此

方法，可以覆盖此方法提供垃圾收集时的其他资源回收，例如关闭文件等。

**20·NDK是什么？**

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。

**21百度地图和谷歌地图的优缺点？**

**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://www.360doc.com/content/14/1221/20/_blank" \o "查看 百度 中的全部文章)，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。

**22·安卓的系统架构**

<http://wenku.baidu.com/view/3ef50eddd15abe23482f4d7e.html>

应用层：使用Java语言进行开发的一些应用程序

应用框架曾：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。

**23.listView优化**

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

**24·安卓上安装了多个浏览器，能否指定页面访问；**

通过直接发送URI把参数带过去，或者通过Manifest里的intentfilter的data属性

**25·简述音乐播放的方式与区别？**

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。

--------------------------------------

**26·工作中如何利用线程池的？】**

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。

**27·onTouch和onTouchEvent区别？**

1.onTouch方法：  
onTouch方法是View的 OnTouchListener接口中定义的方法。  
当一个View绑定了OnTouchLister后，当有touch事件触发时，就会调用onTouch方法。  
（当把手放到View上后，onTouch方法被一遍一遍地被调用）  
  
2.onTouchEvent方法：  
onTouchEvent方法是override 的Activity的方法。  
重新了Activity的onTouchEvent方法后，当屏幕有touch事件时，此方法就会别调用。

**28·JSON的结构？**

json是一种轻量级的数据交换格式，

json简单说就是对象和数组，所以这两种结构就是对象和数组两种结构，通过这两种结构可以表示各种复杂的结构

1、对象：对象表示为“{}”扩起来的内容，数据结构为 {key：value,key：value,...}的键值对的结构，在面向对象的语言中，key为对象的属性，value为对应的属性值，所以很容易理解，取值方法为 对象.key 获取属性值，这个属性值的类型可以是 数字、字符串、数组、对象几种。

2、数组：数组在json中是中括号“[]”扩起来的内容，数据结构为 ["java","javascript","vb",...]，取值方式和所有语言中一样，使用索引获取，字段值的类型可以是 数字、字符串、数组、对象几种。

经过对象、数组2种结构就可以组合成复杂的数据结构了。

**29·SIM卡的EF文件有何作用？**

### 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？**

70个汉字，140个字节

**31·&和&&的区别**

一个位运算符，一个是逻辑运算符

**32·Java的基本特征**

**封装**

封装最好理解了。封装是面向对象的特征之一，是对象和类概念的主要特性。

封装，也就是把客观事物封装成抽象的类，并且类可以把自己的数据和方法只让可信的类或者对象操作，对不可信的进行信息隐藏。

**继承**

面向对象编程 (OOP) 语言的一个主要功能就是“继承”。继承是指这样一种能力：它可以使用现有类的所有功能，并在无需重新编写原来的类的情况下对这些功能进行扩展。

**多态**

多态性（polymorphisn）是允许你将父对象设置成为和一个或更多的他的子对象相等的技术，赋值之后，父对象就可以根据当前赋值给它的子对象的特性以不同的方式运作。简单的说，就是一句话：允许将子类类型的指针赋值给父类类型的指针。

实现多态，有二种方式，覆盖，重载。

覆盖，是指子类重新定义父类的虚函数的做法。

重载，是指允许存在多个同名函数，而这些函数的参数表不同（或许参数个数不同，或许参数类型不同，或许两者都不同）。

**33·注册广播的方式**

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

**34·异步和同步的区别？**

同步：提交请求->等待服务器处理->处理完毕返回 这个期间客户端浏览器不能干任何事

（顺序执行处理事件）

异步: 请求通过事件触发->服务器处理（这是浏览器仍然可以作其他事情）->处理完毕

（可同时执行多个事件的并发）

**35·Afinal和XUtils的区别？**

Xutils是基于Afinal的

xUtils 包含了很多实用的android工具。

xUtils 源于Afinal框架，对Afinal进行了适当的精简，和一些适度的扩展和重构。

xUtils 具有Afinal的一些特性如：无需考虑bitmap在android中加载的时候oom的问题和快速滑动的时候图片加载位置错位等问题； 简洁，约定大于配置...

// oom==内存溢出 ( 释放内存 bitmap )

// 对XUtils 的理解: Dbutils 、ViewUtils、HttpUtils、BitmapUtils

DbUtils模块：android中的orm框架，一行代码就可以进行增删改查。

ViewUtils模块：android中的ioc框架，完全注解方式就可以进行UI绑定和事件绑定。

HttpUtils模块：支持同步，异步方式的请求，支持大文件上传；支持GET,POST,PUT,MOVE,COPY,DELETE,HEAD请求，支持multipart上传设置subtype如related。

BitmapUtils模块：加载bitmap的时候无需考虑bitmap加载过程中出现的oom和android容器快速滑动时候出 现的图片错位等现象；内存管理使用lru算法，更好的管理bitmap内存；可配置线程加载线程数量，缓存大小，缓存路径，加载显示动画等...

**// Android 中ANR异常的处理方法 ：( 意思是: 应用程序没有响应 )**

**有两种情况会报ANR: 自己添加的**

1. 主线程在5秒内没有响应输入事件（事件处理/UI线程）
2. 广播在10秒内没有完成返回( BroadcastReceiver )

**下面是导致android中出现ANR异常的的情况:**

1. **在主线程内进行网络操作**
2. **在主线程内进行一些缓慢的磁盘操作(例如执行没有优化的SQL查询)**

**处理的逻辑是:**

**1.New一个新的线程，进行数据请求**

**2.获取数据后，调用handler.sendmessage方法**

**3.在handler的handle()方法中更新UI**

**// Android 如何指定浏览器访问 指定页面?**

指定action 及 uri 就可以访问制定好的 浏览器了

1>简而言之: 一个程序至少有一个进程，一个进程至少有一个线程

2>线程的划分尺度小于进程，使的多线程程序的并发性高

3>进程在执行的过程中拥有独立的内存单元，而多个线程共享内存，从而提高了程序的运行效率

4>每个独立的线程有一个程序运行的入口、顺序执行序列和程序的出口。但是线程不能独立执行，必须依存在应用程序中，由应用程序提供多个线程执行控制

5>多线程的意义在于一个应用程序中，有多个执行部分可以同时执行， 但是操作系统并没有将多个线程看做多个独立的应用，来实现进程的调度和管理及分配资源，这就是进程和线程的重要区别

**36·ContentProvider的URI的配置？**

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

**37·**AIDL的全称是什么，如何工作

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

**38·屏幕适配？**

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

**// 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

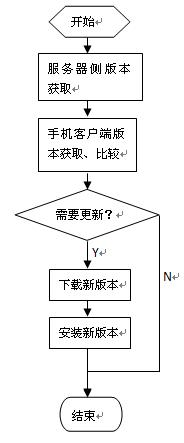
**39·版本更新?数据库的更新？跨版本？**

版本：

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：

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。



**40·安卓自定义组件？**

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

**41.listview分页加载的步骤**

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.

**42.移动互联数据的交换格式**

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

**43.xmpp的底层是怎样实现的？**

socket，双广播，连接广播和通知光爆

**44.Activity的启动模式（四种）**

Standard、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。

**45.广播机制的作用 ????????**

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

**46.图片轮播的实现思路**

图片轮播是由viewpage和handler两者结合所实现的，

**47.Sleep和wait的区别**

1.这两个方法来自不同的类分别是，sleep来自Thread类，和wait来自Object

类。

2.最主要是sleep方法没有释放锁，而wait方法释放了锁，使得其他线程可

以使用同步控制块或者方法。sleep不出让系统资源；wait是进入线程等待池

等待，出让系统资源，其他线程可以占用CPU。一般wait不会加时间限制，

因为如果wait线程的运行资源不够，再出来也没用，要等待其他线程调用

notify/notifyAll唤醒等待池中的所有线程，才会进入就绪队列等待OS分配系

统资源。sleep(milliseconds)可以用时间指定使它自动唤醒过来，如果时间不到

只能调用interrupt()强行打断。

3.wait，notify和notifyAll只能在同步控制方法或者同步控制块里面使用，而

sleep可以在任何地方使用

4. Sleep需要捕获异常,而wait不需要

**48.全局异常处理**

[Android使用UncaughtExceptionHandler捕获全局异常](http://blog.csdn.net/hehe9737/article/details/7662123)（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 "_blank" \o "UncaughtExceptionHandler)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>

**49.静态变量合实例变量的区别**

语法：静态变量前要加static来修饰，而实例变量前则不加；

程序运行时的区别：实例变量属于某个对象的属性，必须创建了实例对象，其中的实例变量才会被分配空间，才能使用这个实例变量；静态变量不属于某个实例对象，而是属于类，所以也称为类变量，静态变量就会被分配空间，静态变量就可以被使用了。

**50.listview失去焦点怎么处理**

在listview子布局里面写，可以解决焦点失去的问题

android:descendantFocusability="blocksDescendants"

**51.什么是匿名内部类，在什么时候调用**

   内部类：内部类可以是static的或者非static的，static内部类只能包含静态方法和静态类变量，只能访问外部类的静态元素，内部类可以实例化，多次使用。

匿名内部类：它只能使用一次，不区分static和非static。如果用到外部类的变量的话，必须是类变量或者实例变量，就是必须是类定义的变量，或者final的局部变量。匿名内部类如果是继承某个类的话是可以重写那个类的方法的，这个和普通内部类一样。

实现事件监听器的时候

用匿名内部类编码非常简洁，也容易读懂，不重复利用时使用。

**52.android中asset和raw的区别；**

\*res/raw和assets的相同点：

1.两者目录下的文件在打包后会原封不动的保存在apk包中，不会被编译成二进制。

\*res/raw和assets的不同点：

1.res/raw中的文件会被映射到R.java文件中，访问的时候直接使用资源ID即R.id.filename；assets文件夹下的文件不会被映射到R.java中，访问的时候需要AssetManager类。  
2.res/raw不可以有目录结构，而assets则可以有目录结构，也就是assets目录下可以再建立文件夹

\*读取文件资源：

1.读取res/raw下的文件资源，通过以下方式获取输入流来进行写操作

InputStream is = getResources().openRawResource(R.id.filename);

//通过 资源 id 直接打开 raw 下的 流文件

2.读取assets下的文件资源，通过以下方式获取输入流来进行写操作

AssetManager am = null;

am = getAssets();

InputStream is = am.open("filename");

// 通过 AssectManager 管理器 来打开 assect 下的流文件

**53.什么是线程**

线程，有时被称为[轻量级进程](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=65949654" \t "_blank)(Lightweight Process，LWP)，是程序执行流的最小[单元](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=275431" \t "_blank)。一个标准的线程由线程[ID](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=265569" \t "_blank)，当前指令指针(PC)，[寄存器](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=35071" \t "_blank)集合和[堆栈](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=331232" \t "_blank)组成。另外，线程是[进程](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=288622" \t "_blank)中的一个实体，是被系统独立调度和分派的基本单位，线程自己不拥有[系统](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=349837" \t "_blank)资源，只拥有一点在运行中必不可少的资源，但它可与同属一个进程的其它线程共享进程所拥有的全部资源。一个线程可以[创建](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=100768" \t "_blank)和[撤消](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=126328" \t "_blank)另一个线程，同一进程中的多个线程之间可以并发执行。由于线程之间的相互制约，致使线程在运行中呈现出间断性。线程也有[就绪](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=425221" \t "_blank)、[阻塞](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=646488" \t "_blank)和[运行](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=493544" \t "_blank)三种基本状态。每一个程序都至少有一个线程，那就是程序本身。

　　线程是程序中一个单一的顺序控制流程。在单个程序中同时运行多个线程完成不同的工作，称为[多线程](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=452880" \t "_blank)。

// 自己找的

**// 线程与进程的区别:**

**定义:**

1. 进程是系统进行资源分配和调度的一个独立单位
2. 线程是进程的一个实体，是CPU调度和分派的基本单位，线程自己基本上不拥有系统资源，只拥有一点在系统运行中必不可少的资源，但是它可与同属一个进程的其他线程共享进程所拥有的全部资源

**关系:**

1. 一个线程可以创建和撤销另一个线程，同一个进程中的多个线程可以同时并发执行，他可以与同进程中的其他线程共享数据，但拥有自己的栈空间，拥有独立的执行顺序

**区别:**

进程和线程区别: 他们是不同的操作系统资源管理方式。

进程有独立的地址空间，一个进程崩溃后，不会对其他进程产生影响

线程只是一个进程中不同的执行路径，线程有自己的堆栈和局部变量，线程之间没有单独的地址空间，一个线程死掉就等于整个进程死掉， 所以多线程的操作 要比多进程的程序健壮。 但在进程切换时耗费资源较大，效率要差一些

对于一些要求同时进行并且又要共享某些变量的并发操作，只能用线程，不能用进程，（因为进程在切换时耗费的资源要大一些，效率差一些）

**54.线程和进程的区别**

进程是表示资源分配的基本单位，又是调度运行的基本单位。

线程是进程中执行运算的最小单位，亦即执行处理机调度的基本单位。

线程的优点：

（1）易于调度。

（2）提高并发性。通过线程可方便有效地实现并发性。进程可创建多个线程来执行同一程序的不同部分。

（3）开销少。创建线程比创建进程要快，所需开销很少。。

（4）利于充分发挥多处理器的功能。通过创建多线程进程（即一个进程可具有两个或更多个线程），每个线程在一个处理器上运行，从而实现应用程序的并发性，使每个处理器都得到充分运行。

进程和线程的关系：

（1）一个线程只能属于一个进程，而一个进程可以有多个线程，但至少有一个线程。  （2）资源分配给进程，同一进程的所有线程共享该进程的所有资源。

（3）处理机分给线程，即真正在处理机上运行的是线程。

（4）线程在执行过程中，需要协作同步。不同进程的线程间要利用消息通信的办法实现同步。

**55.接口和抽象类的区别**

abstract可以修饰抽象方法，而一个类只要有一个抽象方法，就必须用

abstract定义该类，即抽象类。

用interface修饰的类，里面的方法都是抽象方法，因此在定义接口的时候，可以

直接不加那些修饰，系统会默认的添上去。接口里面的字段都是公有常量，即

public static final修饰的字段。

//接口是单继承 、抽象类是多继承

//接口里面只有抽象方法

// 抽象类: 有抽象方法和普通方法 、 抽象类==父类( 抽象类是 顶级父类 )

// 比较两个对象是否相等 == 比较的是地址、 = 比较的是值

//final不能和 static 、actaret 一起使用

**56.xml有几种解析方式、区别**

基本的解析方式有三种: DOM,SAX,Pull

dom解析：解析器读入整个文档，然后构建一个驻留内存的树结构，然后就可以使用 DOM 接口来操作这个树结构。优点是对文档增删改查比较方便，缺点占用内存比较大。

sax解析：基于事件驱动型,优点占用内存少，解析速度快，缺点是只适合做文档的读取，不适合做文档的增删改，不能中途停止。

pull解析：同样基于事件驱动型,android 官方API提供,可随时终止,调用next() 方法提取它们（主动提取事件）

**57.webview的辅助类**

WebSettings、WebViewClient、WebChromeClient

**WebSettings**

设置WebView的一些属性、状态等,例如允许使用javascript，允许使用缓存，允许使用内置的缩放组件

setJavaScriptEnabled(true); 设置支持js

**WebViewClient**

主要帮助WebView处理各种通知、请求事件（例如，点击链接时候如何显示界面，页面开始加载，加载完毕之后有何动作等）

shouldOverrideUrlLoading() onPageStarted() onPageFinished()

**WebChromeClient**

辅助WebView处理Javascript的对话框、网站图标、网站Title、加载进度等

核心方法有onJsAlert() onJsConfirm() onJsPrompt() onProgressChanged() onReceivedIcon()

onReceivedTitle()

**58.webview加载网页的两种方式**

loadUrl(url)  通过url地址加载互联网或本地网页

如：loadUrl("http://www.baidu.com"); // 互联网地址

    loadUrl("file://android\_asset/html/index.html"); /／本地网页

loadData(data,type,encoding) 直接加载网页内容，容易乱码，一般用 [loadDataWithBaseURL](http://localhost/android-new/reference/android/webkit/WebView.html" \l "loadDataWithBaseURL(java.lang.String, java.lang.String, java.lang.String, java.lang.String, java.lang.String))代替

**59.Intent可以被传递的数据类型**

intent是连接Activity, Service, BroadcastReceiver, ContentProvider四大组件的信使,，可以传递八种基本数据类型以及string, Bundle类型，以及实现了Serializable或者Parcelable的类型。

Intent可以划分成显式意图和隐式意图。

显式意图：调用Intent.setComponent()或Intent.setClass()方法明确指定了组件名的Intent为显式意图，显式意图明确指定了Intent应该传递给哪个组件。

隐式意图：没有明确指定组件名的Intent为隐式意图。 Android系统会根据隐式意图中设置的动作(action)、类别(category)、数据（URI和数据类型）找到最合适的组件来处理这个意图。

**60.拦截短信**

短信其实是一个有序广播

1：创建一个继承BroadcastReceiver类的广播接收器。

2：在清单文件中注册广播并在<intent-filter>中设置优先级 priority（只要比你想拦截程序的短信接收器优先级高就可以了）

3：在onReceiver（）中调用abortBroadcast()就可以了

**61.Throws与Throw的区别**

1.throw则是指抛出的一个具体的异常类型。

2.通常在一个方法（类）的声明处通过throws声明方法（类）可能抛出的异常信息，而在方法（类）内部通过throw声明一个具体的异常信息。

3.throws通常不用显示的捕获异常，可由系统自动将所有捕获的异常信息抛给上级方法；

throw则需要用户自己捕获相关的异常，而后在对其进行相关包装，最后在将包装后的异常信息抛出。

如果在方法中会有异常被抛出而你又不希望在这个方法体内对此异常进行处理，可以使用throws在声明方法的时候同时声明他可能会跑出的异常。

4.两者位置不同.

**62.Service是如何启动**

1通过startService  
　　Service会经历onCreate->onStart

　　stopService的时候直接onDestroy  
　　  
　　如果是调用者(TestServiceHolder)自己直接退出而没有调用stopService的  
　　话，Service会一直在后台运行。  
　　下次TestServiceHolder再起来可以stopService。

2通过bindService  
　　Service只会运行onCreate，这个时候TestServiceHolder和TestService绑定在一起  
　　  
　　TestServiceHolder退出了，Srevice就会调用onUnbind->onDestroyed  
　　所谓绑定在一起就共存亡了。

**63.android虚拟机的进程和linux的关系**

DVM指dalivk的虚拟机。每一个Android应用程序都在它自己的进程中运行，都拥有一个独立的Dalvik虚拟机实例。而每一个DVM都是在Linux中的一个进程，所以说可以认为是同一个概念。

**64.程序运行的权限和 文件系统权限的区别**

运行时权限 Dalvik( android 授权) 文件系统 linux 内核授权

**65.访问网络如何加密**

1：对称加密（ＤＥＳ，ＡＥＳ）和非对称（ＲＳＡ公钥与私钥）。

（支付宝里的商户的公钥和私钥）

2：MD5（算法）

3：Base64

**66.访问网络都有哪两种方式**

get与post是两种不同的提交方式

get是把参数数据队列加到提交表单的ACTION属性所指的URL中，值和表单内各个字段一一对应，在URL中可以看到。post是通过HTTP post机制，将表单内各个字段与其内容放置在HTML HEADER内一起传送到ACTION属性所指的URL地址。用户看不到这个过程。

对于get方式，服务器端用Request.QueryString获取变量的值，对于post方式，服务器端用Request.Form获取提交的数据。

get传送的数据量较小，不能大于2KB。post传送的数据量较大，一般被默认为不受限制。但理论上，IIS4中最大量为80KB，IIS5中为100KB。

get安全性非常低，post安全性较高。

HTTP 定义了与服务器交互的不同方法，最基本的方法是 GET 和 POST。事实上 GET 适用于多数请求，而保留 POST 仅用于更新站点。

**67.一键退出的实现**

定义一个类继承Application，定义一个集合存放所有的activity，定义一个添

加的方法，再写一个退出的方法，使用for循环全部调用finish方法，然后在每个

Activity的onCreate方法中调用自定义类里的添加方法，然后在需要使用一键退出

的地方调用类中的退出方法即可。

**68.gc是什么**

GC是垃圾收集的意思（Gabage Collection）,内存处理是编程人员容易出现问题的地方，忘记或者错误的内存回收会导致程序或系统的不稳定甚至崩溃，Java提供的GC功能可以自动监测对象是否超过作用域从而达到自动回收内存的目的，Java语言没有提供释放已分配内存的显示操作方法。

**69.listview的分页加载**

ListView分页加载通常有两种实现方式：一种是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。

在此说一下第一种方式点击底部按钮即加载的实现思路 ：

通过addFooterView()方法为listview底部加入一个“加载更多”按钮和加载进度的视图，默认按钮显示，加载进度为gone。当用户点击“加载更多”按钮时，按钮设置为gone，加载进度显示，并开启子线程从网络上加载下一页数据，加载完毕，发消息给handler，再handler的handleMessage（）方法中通过adapter的notifyDataSetChanged方法通知listview刷新，显示刚加入的数据，并恢复按钮显示，加载进度隐藏。

**70.webservice的UDDI是什么**

UDDI 是一种[目录服务](http://baike.baidu.com/view/1351410.htm" \t "_blank)，企业可以使用它对 Web services 进行注册和搜索。UDDI，英文为 "Universal Description, Discovery and Integration"，可译为“通用描述、发现与集成服务”。

**71、webService中uddi是指什么**

UDDI是一种目录服务，企业可以使用它对Web Service进行注册和搜索。UDDI,英文为“Universal Description and Integration”，可译为“通用描述、发现与集成服务”。

**72、说说java的反射机制原理以及应用场合**

JAVA反射机制是在运行状态中，对于任意一个类，都能够知道这个类的所有

属性和方法；对于任意一个对象，都能够调用它的任意一个方法和属性；这种动态

获取的信息以及动态调用对象的方法的功能称为java语言的反射机制。

类对象: 封装了类的描述信息的对象,类加载的产物,由jvm创建java.lang.Class

应用场景: Gson 序列化反序列化

获取类对象的方式:1. 类名.class2. 类的对象.getClass()3. Class.forName("

包名.类名")

**73、contentprovider怎么实现数据共享**

一个程序可以通过实现一个Content provider的抽象接口将自己的数据完全暴露出去，而且Content providers是以类似数据库中表的方式将数据暴露。Content providers存储和检索数据，通过它可以让所有的应用程序访问到，这也是应用程序之间唯一共享数据的方法。

要想使应用程序的数据公开化，可通过2种方法：创建一个属于你自己的Content provider或者将你的数据添加到一个已经存在的Content provider中，前提是有相同数据类型并且有写入Content provider的权限。

如何通过一套标准及统一的接口获取其他应用程序暴露的数据？

Android提供了ContentResolver，外界的程序可以通过ContentResolver接口访问ContentProvider提供的数据。

**74、java有几种引用类型**

强引用，软引用、弱引用、幽灵引用、

**75、如何让Activity变成一个窗口？**

答：在清单文件AndroidManifest.xml中相应的<activity>标签内设置属性android:theme=”@android:style/The

**76，启动一个线程时run（）还是start（）**

start（）

**77，ArrayList的便利**

ArrayList<String> al=new ArrayList<String>();

for(int i=0;i<10;i++){

al.add(String.valueOf(i));

}

Iterator<String> it=al.iterator();

while(it.hasNext()){

System.out.println(it.next());

}

**78、String,StringBuffer区别**

JAVA平台提供了两个类：String和StringBuffer，它们可以储存和操作字符串，

即包含多个字符的字符数据。这个String类提供了数值不可改变的字符串。而这个

StringBuffer类提供的字符串进行修改。

StringBuilder与StringBuffer的区别 。

StringBuffer

1，一个类似于 String 的字符串缓冲区，对它的修改不会像String那样重创建对象。

2，使用append()方法修改Stringbuffer的值，使用toString()方法转换为字符串。

Stringbuild是jdk1.5后用来替换stringBuffer的一个类，大多数时候可以替换

StringBuffer。和StringBuffer的区别在于Stringbuild是一个单线程使用的类，不值

执行线程同步所以比StringBuffer的速度快，效率高。

**79、集合的实现类**

HashSet：元素无需、不可重复；ArrayList：元素有序，可重复

HashMap以兼职对的形式伯村数据，key值可重复，value值不可重复

**80、android中常用的布局**

第一种：帧布局（框架布局）FrameLayout，在这个布局中，所有的子元素统统放于这块区域的左上角，并且后面的子元素直接覆盖在前面的子元素之上，将前面的子元素部分和全部遮挡。

第二种：线性布局LinearLayout，最常用的一种布局方式，所有子控件的对齐方式，取决于如何定义 orientation的属性：vertical 垂直方向 ，如果按照这种方向所有的子控件将按照垂直的方式分布在布局上，每行只允许有一个子元素，horizontal水平方向 ，这时子控件将会以水平的方向分布在布局中。

第三种：绝对布局AbsoluteLayout，又可以叫做坐标布局，可以直接指定子元素的绝对位置，这种布局简单直接，直观性强，但是由于手机屏幕尺寸差别比较大，使用绝对定位的适应性会比较差。

第四种：相对布局RelativeLayout，允许子元素指定它们相对于其父元素或兄弟元素的位置，这是实际布局中最常用的布局方式之一。它灵活性大很多，当然属性也多，操作难度也大，属性之间产生冲突的的可能性也大，使用相对布局时要多做些测试。

第五种：表格布局TableLayout，表格布局TableLayout以行列的形式管理子元素，每一行是一个TableRow布局对象，当然也可以是普通的View对象，TableRow里每放一个元素就是一列，总列数由列数最多的那一行决定。

第六种：网格布局 GridLayout，在Android 4.0中，新引入的GridLayout网格布局，GridLayout布局使用虚细线将布局划分为行，列和单元格，也支持一个控件在行，列上都有交错排列。而GridLayout使用的其实是跟LinearLayout类似的API，只不过是修改了一下相关的标签而已，所以对于开发者来说，掌握GridLayout还是很容易的事情。

**81、char，varchar区别**

1、CHAR。CHAR存储定长数据很方便，CHAR字段上的索引效率级高，比如定义char(10)，那么不论你存储的数据是否达到了10个字节，都要占去10个字节的空间。

2、VARCHAR。存储变长数据，但存储效率没有CHAR高。如果一个字段可能的值是不固定长度的，我们只知道它不可能超过10个字符，把它定义为 VARCHAR(10)是最合算的。VARCHAR类型的实际长度是它的值的实际长度+1。为什么“+1”呢？这一个字节用于保存实际使用了多大的长度。

**82.io流读取输出方法**

文件读取：

public static void main(String[] args) throws IOException

{

//创建一个文件读取流对象，并初始化时指定名称的文件相关联。

//要保证此文件是已存在，如不存在会发生异常：FileNotFoundException

FileReader fr = new FileReader("demo.txt");

//调用读取流对象的read方法。

//read方法一次只读一个字符，而且会自动向下读取。当读取完后会返回-1.

int ch = 0;

while((ch = fr.read())!=-1)

System.out.print((char)ch);

fr.close();

}

**83，如何序列化**

1). 实现序列化 Serializable

2). 属性私有化，并提供get，set方法

3). 提供无参构造

4). 属性名必须与json串中属性名保持一致 （因为Gson解析json串底层用到了Java的反射原理）

**84、如何在android中调用js**

loadUrl("javascript:test()");

**85、**Intent的原理

intent是连接Activity, Service, BroadcastReceiver, ContentProvider四大组件的信使,，可以传递八种基本数据类型以及string, Bundle类型，以及实现了Serializable或者Parcelable的类型。

Intent可以划分成显式意图和隐式意图。

显式意图：调用Intent.setComponent()或Intent.setClass()方法明确指定了组件名的Intent为显式意图，显式意图明确指定了Intent应该传递给哪个组件。

隐式意图：没有明确指定组件名的Intent为隐式意图。 Android系统会根据隐式意图中设置的动作(action)、类别(category)、数据（URI和数据类型）找到最合适的组件来处理这个意图。

**86、广播分为几种**

第一种:在清单文件中声明,添加

<receive android:name=".IncomingSMSReceiver ">

<intent-filter>

<action android:name="android.provider.Telephony.SMS\_RECEIVED")

<intent-filter>

<receiver>

第二种使用代码进行注册如:

IntentFilter filter = new IntentFilter("android.provider.Telephony.SMS\_RECEIVED");

IncomingSMSReceiver receiver = new IncomgSMSReceiver();

registerReceiver(receiver.filter);

**87、intent和activity的区别**

Activity活动，可以理解为窗口，是人机交互的核心，所以又被人们说成是所有程序的根本。Intent意图，负责所有Activity窗口通讯。Service服务是相对Activity来说的，不需要人机交互，但可以为Activity提供交互必需要的一些东西。

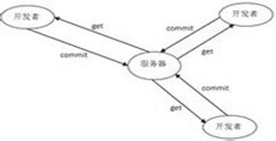
Android中，Activity是所有程序的根本，所有程序的流程都运行在Activity之中，Activity具有自己的生命周期. 由系统控制生命周期，程序无法改变，但可以用onSaveInstanceState保存其状态对于Activity，关键是其生命周期的把握（如下图），其次就是状态的保存和恢复（onSaveInstanceState onRestoreInstanceState），以及Activity之间的跳转和数据传输（intent）。

Intent

Android中提供了Intent机制来协助 应用间的交互与通讯，Intent负责对应用中一次操作的动作、动作涉及数据、附加数据进行描述，Android则根据此Intent的描述，负责找到对 应的组件，将 Intent传递给调用的组件，并完成组件的调用。Intent不仅可用于应用程序之间，也可用于应用程序内部的Activity/Service之间的交互。

**88、**解释SVN

Svn是一种集中式文件版本管理系统。集中式管理的工作流程如下图：集中式代码管理的核心是服务器，所有开发者在开始新一天的工作之前必须从服务器获取代码，然后开发，最后解决冲突，提交。所有的版本信息都放在服务器上。如果脱离了服务器，开发者基本上是不可以工作。下面举例说明：开始新一天的工作：1，从服务器下载项目组最新代码。2：进入自己的分支，进行工作，每隔一个小时向服务器自己的分支提交一次代码（很多人都有这个习惯。因为有时候自己对代码改来改去，最后又想还原到前一个小时的版本，或者看看前一个小时自己修改了那些代码，就需要这样做了）。3：下班时间快到了，把自己的分支合并到服务器主分支上，一天的工作完成，并反映给服务器。这就是经典的svn工作流程，从流程上看，有不少缺点，但也有优点。缺点：1、 服务器压力太大，数据库容量暴增。2、 如果不能连接到服务器上，基本上不可以工作，看上面第二步，如果服务器不能连接上，就不能提交，还原，对比等等。3、不适合开源开发（开发人数非常非常多，但是Google app engine就是用svn的）。但是一般集中式管理的有非常明确的权限管理机制（例如分支访问限制），可以实现分层管理，从而很好的解决开发人数众多的问题。优点：1、 管理方便，逻辑明确，符合一般人思维习惯。2、 易于管理，集中式服务器更能保证安全性。3、 代码一致性非常高。4、 适合开发人数不多的项目开发。5、大部分软件配置管理的大学教材都是使用svn和vss。



**89、ipc机制（及原理）**

）IPC（Inter-Process Communication，进程间通信）,

aidl是 Android Interface definition language的缩写，它是一种android内部进程通信接口的描述语言，通过它我们可以定义进程间的通信接口.编译器可以通过扩展名为aidl的文件生成一段代码，通过预先定义的接口达到两个进程内部通信进程的目的.

BroadcastReceiver也可以实现进程间通信

ContentProvider 提供进程间数据共享

2).Android的 Service工作流程

1.Android虚拟机启动之前系统会先启动service Manager进程；

2.service Manager打开binder驱动，并通知binder kernel驱动程序这个进程将作为System Service Manager；

3.然后该进程将进入一个循环，等待处理来自其他进程的数据。

4.用户创建一个System service后，通过defaultServiceManager得到一个远程ServiceManager的接口，通过这个接口我们可以调用addService函数将System service添加到Service Manager进程中；

5.然后client可以通过getService获取到需要连接的目的Service的IBinder对象，这个IBinder是Service的BBinder在binder kernel的一个参考，

6.所以service IBinder 在binder kernel中不会存在相同的两个IBinder对象，每一个Client进程同样需要打开Binder驱动程序。对用户程序而言，我们获得这个对象就可以通过binder kernel访问service对象中的方法。

7.Client与Service在不同的进程中，通过这种方式实现了类似线程间的迁移的通信方式，对用户程序而言当调用Service返回的IBinder接口后，访问Service中的方法就如同调用自己的函数。

**90、八种基本数据类型**

byte、short、int、long、float、double、char、boolean

**91、上传图片怎么做**

用图片异步加载通过开启子线程实现

**92、异常**

答：编译时异常： 程序正确，但因为外在的环境条件不满足引发。例如：用户错

误及I/O问题----程序试图打开一个并不存在的远程Socket端口。这不是程序本

身的逻辑错误，而很可能是远程机器名字错误(用户拼写错误)。对商用软件系统，

程序开发者必须考虑并处理这个问题。Java编译器强制要求处理这类异常，如果

不捕获这类异常，程序将不能被编译。

运行期异常： 这意味着程序存在bug，如数组越界，0被除，入参不满足规范.....

这类异常需要更改程序来避免，Java编译器强制要求处理这类异常。

**93、==equals区别**

1、 ==是判断两个变量或实例是不是指向同一个内存空间

equals是判断两个变量或实例所指向的内存空间的值是不是相同

2、==是指对内存地址进行比较

equals()是对字符串的内容进行比较

3、==指引用是否相同

equals（）指的是值是否相同

**94 public，private，protect权限**

public在任何情况下都可用；

protect可在当前类，同一个包中和子孙类中使用；

default：可在当前类和同一个包中使用

private只能在当前类中使用

**95、HttpClient与HttpUrlConnection的区别**

后者更为常用。HttpClient是个开源框架，封装了访问http的请求头，参数，内容体，响应等等， HttpURLConnection是java的标准类，什么都没封装，用起来太原始，不方便HttpClient实际上是对Java提供方法的一些封装，在HttpURLConnection中的输入输出流操作，在这个接口中被统一封装成了HttpPost(HttpGet)和HttpResponse，这样，就减少了操作的繁琐性。

**96，json解析方式的两种区别**

1，SDK提供JSONArray，JSONObject

2，google提供的 Gson

通过fromJson()实现对象的反序列化（即将json串转换为对象类型）

通过toJson()实现对象的序列化 （即将对象类型转换为json串）

通过google提供的Gson解析json时，定义JavaBean的规则是什么？

1). 实现序列化 Serializable

2). 属性私有化，并提供get，set方法

3). 提供无参构造

4). 属性名必须与json串中属性名保持一致 （因为Gson解析json串底层用到了Java的反射原理）

**97、什么时候用递归**

树用递归方便，用其它方法好像跟本不可能．

递归用在特定的函数如 f(x)= f(x-1) + 2

像这种情况，你想算出f(x)就必需算出f(x-1)，而f(x)和f(x-1)实际上都是用共一个方法，只是参数相差一．这种时候用递归就很快．

**98、jdbc中怎么处理事物**

事务，也是数据库事务，指的是作为单个逻辑工作单元执行的一系列操作。正常的情况下，操作应该顺利进行，与操作相关的所有数据库信息也成功地更新；

但是，如果在这一系列过程中任何一个环节出了差错，导致操作失败了，数据库中所有信息都必须保持操作前的状态不变。否则，数据库的信息将会一片混乱而不可预测。

一个逻辑工作单元要称为事务，必须满足ACID（原子性，一致性，隔离性和持久性）

事务的结束只能有两种形式：提交和回滚。操作完全成功则提交，产生永久性的修改；操作不完全成功则回滚，恢复到事务开始前的状态。它们将结束一个事务。

（1）关闭自动提交事务。通过设置连接的自动提交事务属性为false，

**99、android时间分发机制**

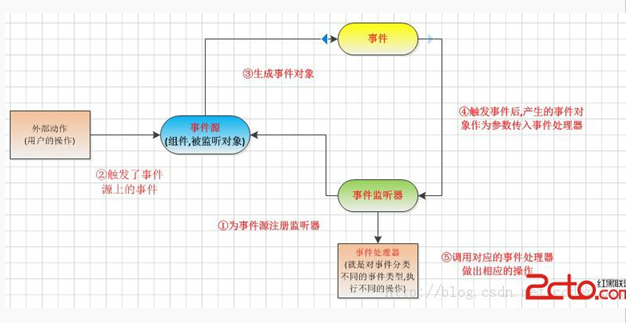
当你点击了某个控件，首先会去调用该控件所在布局的dispatchTouchEvent方法，然后在布局的dispatchTouchEvent方法中找到被点击的相应控件，再去调用该控件的dispatchTouchEvent方法

**100，static内部发出对非static的调用**

不能调用

**基于回调机制的处理以及监听机制的处理**

监听机制的处理模型图：



①直接使用匿名内部类,作为事件监听器

ps:就是我们平时最常用的那种,setXxxListener后就重写里面的对应方法

通常都是临时使用一次,复用性不高

②使用内部类作为事件监听器

使用优点:可以在该类中复用,可直接访问外部类的所有界面组件

③使用外部类作为事件监听器

就是另外创建一个处理事件的[Java](http://www.2cto.com/kf/ware/Java/" \t "_blank)文件,该形式比较少见

因为外部类不能直接访问用户界面类中的[组件](http://www.2cto.com/kf/all/zujian/" \t "_blank),要通过构造方法将组件传入使用,

这样的结果就是代码不够简洁

④直接使用Activity作为事件监听器

只需要让Activity类实现~Listener事件监听接口,在Activity中定义重写对应的事件处理器方法

<http://www.2cto.com/kf/201403/287434.html>

回调机制的处理：

Android平台中，每个View都有自己的处理事件的回调方法，通过重写View中的这些回调方法来实现需要的响应事件。当某个事件没有被任何一个View处理时，便会调用Activity中相应的回调方法。Android提供了以下回调方法供用户使用：

1. onKeyDown：  
   功能：该方法是接口KeyEvent.Callback中的抽象方法，所有的View全部实现了该接口并重写了该方法，该方法用来捕捉手机键盘被按下的事件。

2. onKeyUp：  
   功能：该方法同样是接口KeyEvent.Callback中的一个抽象方法，并且所有的View同样全部实现了该接口并重写了该方法，onKeyUp方法用来捕捉手机键盘按键抬起的事件。

3. onTouchEvent：

   功能：该方法在View类中的定义，并且所有的View子类全部重写了该方法，应用程序可以通过该方法处理手机屏幕的触摸事件。

４.　onTrackBallEvent：  
　　　功能：　接下来将介绍的是手机中轨迹球的处理方法onTrackBallEvent。所有的View同样全部实现了该方法。

５.　 onFocusChanged：  
　　　功能：　前面介绍的各个方法都可以在View及Activity中重写，接下来介绍的onFocusChanged却只能在View中重写。该方法是焦点改变的回调方法，当某个控件重写了该方法后，当焦点发生变化时，会自动调用该方法来处理焦点改变的事件。

http://blog.csdn.net/liranke/article/details/6855601

**UI中padding和margin的区别**

Padding 用来指定组件内的内容距离组件边界的距离;

Margin用来指定控件与控件之间的距离

**View如何刷新**

实现模式对话框的OnOk()函数，在其中添加下面的代码就可以了：  
（(C\*\*\*View)(AfxGetMainWnd()->GetActiveView())）->Invalidate();  
其中C\*\*\*View替换为你自己的View类的名字。  
当然，调用Invalidate()只是按你的问题要求实现了刷新显示而已，至于如何实现图片大小的更新，仍需要其他代码（比如在View类中添加代表图片大小的成员变量，在OnDraw()中按照成员变量绘制图片，在OnOK()中更新View成员变量的值）

**This和getApplicationContext ()的区别**

getApplicationContext():生命周期是整个应用，应用摧毁，它才摧毁。

this:代表当前,在Activity当中就是代表当前的Activity，换句话说就是Activity.this在Activity当中可以缩写为this.

**横竖屏切换activity周期的变化**

1、新建一个Activity并把各个生命周期打印出来

2、运行Activity，得到如下信息

onCreate-->  
onStart-->  
onResume-->

3、按crtl+f12切换成横屏时

onSaveInstanceState-->  
onPause-->  
onStop-->  
onDestroy-->  
onCreate-->  
onStart-->  
onRestoreInstanceState-->  
onResume-->

4、再按crtl+f12切换成竖屏时，打印信息如下

onSaveInstanceState-->  
onPause-->  
onStop-->  
onDestroy-->  
onCreate-->  
onStart-->  
onRestoreInstanceState-->  
onResume-->  
onSaveInstanceState-->  
onPause-->  
onStop-->  
onDestroy-->  
onCreate-->  
onStart-->  
onRestoreInstanceState-->  
onResume-->

5、修改AndroidManifest.xml，把该Activity添加 android:configChanges="orientation"，执行步骤3

onSaveInstanceState-->  
onPause-->  
onStop-->  
onDestroy-->  
onCreate-->  
onStart-->  
onRestoreInstanceState-->  
onResume-->

6、再执行步骤4，发现不会再打印相同信息，但多打印了一行onConfigChanged

onSaveInstanceState-->  
onPause-->  
onStop-->  
onDestroy-->  
onCreate-->  
onStart-->  
onRestoreInstanceState-->  
onResume-->  
onConfigurationChanged-->

7、把步骤5的android:configChanges="orientation" 改成 android:configChanges="orientation|keyboardHidden"，执行步骤3，就只打印onConfigChanged

onConfigurationChanged-->

8、执行步骤4

onConfigurationChanged-->  
onConfigurationChanged-->

总结：

1、不设置Activity的android:configChanges时，切屏会重新调用各个生命周期，切横屏时会执行一次，切竖屏时会执行两次

2、设置Activity的android:configChanges="orientation"时，切屏还是会重新调用各个生命周期，切横、竖屏时只会执行一次

3、设置Activity的android:configChanges="orientation|keyboardHidden"时，切屏不会重新调用各个生命周期，只会执行onConfigurationChanged方法

总结一下整个Activity的生命周期

补充一点，当前Activity产生事件弹出Toast和AlertDialog的时候

ActivityActivity的生命周期不会有改变运行时按下HOME键(跟被完全覆盖是一样的)：onSaveInstanceState --> onPause --> onStop       onRestart -->onStart--->onResume

Activity未被完全覆盖只是失去焦点：onPause--->onResume

**如何实现ViewPager**

先在布局文件中调用v4包下的ViewPager，在java代码中初始化ViewPager，并设置adapter（继承PagerAdapter），再设置监听onPageChangeListener

**Sqlite中的事务如何实现**

开启事务： start transaction   
 sql:…update account set money=money-500 where id=1;--检验隔离性  
 sql…update account set money=money+500 where id=2;   
commit提交-验证原子性、一致性

**Android中常用的菜单以及区别**

1、左右推出的Menu

使用方法：

一.首先导入SlidingMenu这个库文件

二.在你的项目中引入SlidingMenu这个库

eclipse中的流程：右击你的项目 -- Properties -- Android  -- Add  选择SlidingMenu库

三.在你的项目中使用这个SlidingMenu

配置设定方法中 setMode() 是常用属性有：

1）SlidingMenu.LEFT    左测菜单

2）SlidingMenu.RIGHT   右侧菜单

3）SlidingMenu.LEFT\_RIGHT    左右2测菜单

2、Path菜单

3、GridView菜单

4、popuwindow菜单

5、系统菜单

**重载和重写的区别**

方法的重载属于,编译时多态,方法名相同参数列表不同，返回值必须相同或都没有返回值类型。         方法的覆盖属于运行时多态，子类覆盖父类的方法,子类指向父类引用,在调用方法的时候用父类的引用调用。

**Set list map集合的特点**

List接口 元素有序可重复.

实现类有：ArrayList    数组实现轻量级，运行快，线程不安全。JDK1.2 查询快

 Vector   数组实现重量级，运行慢，线程安全。JDK1.0保证元素的无序唯一，自定义对象存进HashSet为了保证元素内容不重复需要覆盖hashCode()与equals()方法。

          SortedSet(不重要) 元素有序（Unicode升序）唯一

  LinkedList  链表实现 常用语堆栈与队列的实现 增删操作快

Set 接口 元素无序不可重复

实现类有：HashSet，底层用hashCode()算法实现，

  TreeSet要求元素有序，自定义的对象需要实

现Comparable接口的 compareTo（object o）方法

 Map(接口): 与Collection接口无关,有一个子接口SortedMap特点: 元素是key-value, key

 唯一,无序; value可重复

实现类: HashMap 轻量级 线程不安全的,允许key或value为null JDK1.2

        HashTable 重量级 线程安全的 不允许key或value为null   JDK1.0

       Properties是HashTable的子类,主键和值都是字符串

SortedMap:(不重要)

特点: key唯一,有序(Unicode升序)

实现类:TreeMap

**gradView上拉加载和下拉刷新怎么实现**

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，调用 addFooterView，添加记录到adapter, adapter调用 notifyDataSetChanged 更新数据;如果没有记录了，把自定义的mFooterView去掉。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

**用过哪些框架**

XUtils AFinal 第三方开源框架

**横向scrollView和ViewPager之间的冲突**

重写ScrollView，

publicclassHomeFeatureLayoutextendsHorizontalScrollView {

privateArrayList<ListItem> items = null;

privateGestureDetector gestureDetector;

View.OnTouchListener gestureListener;

privatestaticfinalint SWIPE\_MIN\_DISTANCE = 5;

privatestaticfinalint SWIPE\_THRESHOLD\_VELOCITY = 300;

privateint activeFeature = 0;

publicHomeFeatureLayout(Context context, ArrayList<ListItem> items){

super(context);

setLayoutParams(newLayoutParams(LayoutParams.FILL\_PARENT, LayoutParams.WRAP\_CONTENT));

setFadingEdgeLength(0);

this.setHorizontalScrollBarEnabled(false);

this.setVerticalScrollBarEnabled(false);

LinearLayout internalWrapper = newLinearLayout(context);

internalWrapper.setLayoutParams(newLayoutParams(LayoutParams.FILL\_PARENT, LayoutParams.FILL\_PARENT));

internalWrapper.setOrientation(LinearLayout.HORIZONTAL);

addView(internalWrapper);

this.items = items;

for(int i = 0; i< items.size();i++){

LinearLayout featureLayout = (LinearLayout) View.inflate(this.getContext(),R.layout.homefeature,null);

TextView header = (TextView) featureLayout.findViewById(R.id.featureheader);

ImageView image = (ImageView) featureLayout.findViewById(R.id.featureimage);

TextView title = (TextView) featureLayout.findViewById(R.id.featuretitle);

title.setTag(items.get(i).GetLinkURL());

TextView date = (TextView) featureLayout.findViewById(R.id.featuredate);

header.setText("FEATURED");

Image cachedImage = newImage(this.getContext(), items.get(i).GetImageURL());

image.setImageDrawable(cachedImage.getImage());

title.setText(items.get(i).GetTitle());

date.setText(items.get(i).GetDate());

internalWrapper.addView(featureLayout);

}

gestureDetector = newGestureDetector(newMyGestureDetector());

setOnTouchListener(newView.OnTouchListener() {

@Override

publicboolean onTouch(View v, MotionEventevent) {

if (gestureDetector.onTouchEvent(event)) {

returntrue;

}

elseif(event.getAction() == MotionEvent.ACTION\_UP || event.getAction() == MotionEvent.ACTION\_CANCEL ){

int scrollX = getScrollX();

int featureWidth = getMeasuredWidth();

activeFeature = ((scrollX + (featureWidth/2))/featureWidth);

int scrollTo = activeFeature\*featureWidth;

smoothScrollTo(scrollTo, 0);

returntrue;

}

else{

returnfalse;

}

}

});

}

classMyGestureDetectorextendsSimpleOnGestureListener {

@Override

publicboolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) {

try {

//right to left

if(e1.getX() - e2.getX() > SWIPE\_MIN\_DISTANCE &&Math.abs(velocityX) > SWIPE\_THRESHOLD\_VELOCITY) {

activeFeature = (activeFeature < (items.size() - 1))? activeFeature + 1:items.size() -1;

smoothScrollTo(activeFeature\*getMeasuredWidth(), 0);

returntrue;

}

//left to right

elseif (e2.getX() - e1.getX() > SWIPE\_MIN\_DISTANCE &&Math.abs(velocityX) > SWIPE\_THRESHOLD\_VELOCITY) {

activeFeature = (activeFeature >0)? activeFeature - 1:0;

smoothScrollTo(activeFeature\*getMeasuredWidth(), 0);

returntrue;

}

} catch (Exception e) {

// nothing

}

returnfalse;

}

}

**集合和数组的区别**

1）：数组在定义时必须定义长度，而集合在定义时不必定义长度。

2）：数组在定义时必须要声明数组的数据类型，而集合不必。但是一般情况下我们都是存储统一数据类型的数据，我们可以使用泛型的写法来限制集合里面的数据类型。

**什么方法不能被重写**

被final修饰的方法，构造方法，

**编译时异常和运行时异常的区别**

异常表示程序运行过程中可能出现的非正常状态，运行时异常表示虚拟机的通常操作中可能遇到的异常，是一种常见运行错误。java编译器要求方法必须声明抛出可能发生的非运行时异常，但是并不要求必须声明抛出未被捕获的运行时异常。

**Gps定位和网络定位的区别**

GPS定位需设置权限android.permission.ACCESS\_LOCATION\_FINE

网络定位需设置权限android.permission.ACCESS\_LOCATION\_COARSE

即GPS定位比网络定位在精确度方面更精准，但在速度方面稍差一些，而且GPS定位适合在室外环境下使用，网络定位在室内，室外都可以。

**Collection 和 Collections的区别**Collection是集合类的上级接口，继承与他的接口主要有Set 和List  
Collections是针对集合类的一个帮助类，他提供一系列静态方法实现对各种集合的搜索、排序、线程安全化等操作。

Native是啥

**Service实现的两种方式**

1.Context.startService()方式启动

①Context.startService()方式的生命周期： 启动时，startService –> onCreate() –> onStart()停止时，stopService –> onDestroy()如果调用者直接退出而没有停止Service，则Service 会一直在后台运行 Context.startService()方法启动服务，在服务未被创建时，系统会先调用服务的onCreate()方法，接着调用onStart()方法。如果调用startService()方法前服务已经被创建，多次调用startService()方法并不会导致多次创建服务，但会导致多次调用onStart()方法。采用startService()方法启动的服务，只能调用Context.stopService()方法结束服务，服务结束时会调用onDestroy()方法附代码

2.Context.bindService()方式启动：①Context.bindService()方式的生命周期： 绑定时,bindService -> onCreate() –> onBind()调用者退出了，即解绑定时,Srevice就会unbindService –>onUnbind() –> onDestory()Context.bindService()方式启动 Service的方法：绑定Service需要三个参数：bindService(intent, conn, Service.BIND\_AUTO\_CREATE);第一个：Intent对象第二个：ServiceConnection对象，创建该对象要实现它的onServiceConnected()和 onServiceDisconnected()来判断连接成功或者是断开连接第三个：如何创建Service，一般指定绑定的时候自动创建附代码

**手机上使用的传感器有什么**

#define SENSOR\_TYPE\_ACCELEROMETER       1 //加速度

#define SENSOR\_TYPE\_MAGNETIC\_FIELD      2 //磁力

#define SENSOR\_TYPE\_ORIENTATION         3 //方向

#define SENSOR\_TYPE\_GYROSCOPE           4 //陀螺仪

#define SENSOR\_TYPE\_LIGHT               5 //光线感应

#define SENSOR\_TYPE\_PRESSURE            6 //压力

#define SENSOR\_TYPE\_TEMPERATURE         7 //温度

#define SENSOR\_TYPE\_PROXIMITY           8 //接近

#define SENSOR\_TYPE\_GRAVITY             9 //重力

#define SENSOR\_TYPE\_LINEAR\_ACCELERATION 10//线性加速度

#define SENSOR\_TYPE\_ROTATION\_VECTOR     11//旋转矢量

**SurfaceView实现的原理**

SurfaceView，它拥有独立的绘图表面，即它不与其宿主窗口共享同一个绘图表面。由于拥有独立的绘图表面，因此SurfaceView的UI就可以在一个独立的线程中进行行绘制。又由于不占用主线程资源，SurfaceView一方面可以实现复杂而高效的UI，另一方面又不会导致用户输入得不到及时响应。

**Int Integer的区别**

 int 是基本类型，直接存数值

　　1.Java 中的数据类型分为基本数据类型和复杂数据类型

　　int 是前者>>integer 是后者（也就是一个类）

　　2.初始化时>>

　　int i =1;

　　Integer i= new Integer(1);(要把integer 当做一个类看)

　　int 是基本数据类型（面向过程留下的痕迹，不过是对java的有益补充）

　　Integer 是一个类，是int的扩展，定义了很多的转换方法

类似的还有：float Float;double Double;string String等

**Application类的作用**

Application和Actovotu,Service一样是android框架的一个系统组件，当android程序启动时系统会创建一个 application对象，用来存储系统的一些信息。通常我们是不需要指定一个Application的，这时系统会自动帮我们创建，如果需要创建自己 的Application，也很简单创建一个类继承 Application并在manifest的application标签中进行注册(只需要给Application标签增加个name属性把自己的 Application的名字定入即可)。

android系统会为每个程序运行时创建一个Application类的对象且仅创建一个，所以Application可以说是单例 (singleton)模式的一个类.且application对象的生命周期是整个程序中最长的，它的生命周期就等于这个程序的生命周期。因为它是全局 的单例的，所以在不同的Activity,Service中获得的对象都是同一个对象。所以通过Application来进行一些，数据传递，数据共享 等,数据缓存等操作。

**嵌入式操作系统**

嵌入式实时操作系统是指当外界事件或数据产生时，能够接受并以足够快的速度予以处理，其处理的结果又能在规定的时间之内来控制生产过程或对处理系统作出快速响应，并控制所有实时任务协调一致运行的嵌入式操作系统。主要用于工业控制、 军事设备、 航空航天等领域对系统的响应时间有苛刻的要求，这就需要使用实时系统。又可分为软实时和硬实时两种，而android是基于linux内核的，因此属于软实时。

**ViewFlipper和ViewPager的区别**

1.ViewFilpper类继承于ViewAnimator类。而ViewAnimator类继承于FrameLayout。

查看ViewAnimator类的源码可以看出此类的作用主要是为其中的View切换提供动画效果。该类有如下几个和动画相关的方法。

2.ViewPager用于实现多页面的切换效果，该类存在于Google的兼容包里面，所以在引用时记得在BuilldPath中加入“android-support-v4.jar”

**定义数组的几种方式**

nt a[]; //声明未初始化  
a = new int [10]; // 定义占用空间大小（10个int）  
int a[] = new int [10]; //声明并定义大小（即分配了指定大小的空间）  
int a[] = {1,2,3}; // 声明并初始化，占用空间大小是3个int。

**知道哪几种排序方式**

冒泡

public static int[] doMaoPao(int[] in){

int temp;

int length = in.length;

for(int i = 0;i<length; i++){

for(int j=0;j<length-1-i;j++){

if(in[j]<in[j+1]){

temp = in[j+1];

in[j+1] = in[j];

in[j] = temp;

}

System.out.print("输出j="+in[j]+"");

System.out.println("");

System.out.print("输出j+1= "+in[j+1]);

}

// System.out.print("i="+in[i]);

}

return in;

}

选择

public static int[] doPaiXv(int[] in){

for(int i=0;i<in.length;i++){

int temp = 0;

for(int j=i+1;j<in.length;j++){

if(in[i]>in[j]){

temp = in[i];

in[i] = in[j];

in[j] = temp;

}

System.out.print("输出j="+in[j]+"");

System.out.println("");

System.out.print("输出i= "+in[i]);

}

}

return in;

}

二分

public static void binary\_sort(int a[]){

int i,j,temp;

int left,right,mid;

for(i = 1; i<a.length; i++){

left = 0;

right = i-1;

temp = a[i];

while(left <= right){

mid = (left + right) /2;

if(a[mid] > temp){

right = mid - 1;

}else{

left = mid + 1;

}

}

for(j=i-1; j>right; j--){

a[j+1] = a[j];

a[right+1] = temp;

}

}

}

public static void main(String[] args) {

int a[] = {2,6,9,8,4,7,3,1,0,5};

binary\_sort(a);

for(int i=0;i<a.length;i++)

System.out.print(a[i]+ "");

}

**Doget和doPost的区别**

通过get方式提交的数据有大小的限制，通常在1024字节左右。也就是说如果提交的数据很大，用get方法就可需要小心；而post方式没有数据大小的限制，理论上传送多少数据都可以。

通过get传递数据，实际上是将传递的数据按照”key，value”的方式跟在URL的后面来达到传送的目的的；而post传递数据是通过http请求的附件进行的，在URL中并没有明文显示

**什么是API**

API（Application Programming Interface,应用程序编程接口）是一些预先定义的函数，目的是提供应用程序与开发人员基于某软件或硬件的以访问一组例程的能力，而又无需访问源码，或理解内部工作机制的细节.  
简单来说,就是一套可供调用的系统接口.

**常遇到的异常、如何解决**

异常的继承结构：

基类为Throwable，Error和Exception继承Throwable，RuntimeException和IOException等继承Exception，具体的RuntimeException继承RuntimeException。

Error和RuntimeException及其子类成为未检查异常（unchecked），其它异常成为已检查异常（checked）。

1、 SQLException：操作数据库异常类。

问题：当Java应用程序与数据库进行沟通时如果产生了错误，就会触发这个类。同时会将数据库的错误信息通过这个类显示给用户。当用户插入数据的时候，如果这个字段的值跟现有的纪录重复了，违反了数据库的唯一性约束，此时数据库就会跑出一个异常信息。这个信息一般用户可能看不到，因为其发生在数据库层面的。此时这个操作数据库异常类就会捕捉到数据库的这个异常信息，并将这个异常信息传递到前台。如此的话，前台用户就可以根据这个异常信息来分析发生错误的原因。

解决方案：检查插入的数据是否重复。  
 2、 ClassCastException：数据类型转换异常。

问题：在Java应用程序中，有时候需要对数据类型进行转换。这个转换包括显示的转换与隐式的转换。不过无论怎么转换，都必须要符合一个前提的条件，即数据类型的兼容性。在数据类型进行转换之前，就保证数据类型的兼容性。如此的话，就不容易造成数据类型的转换异常。如在只允许数值类型的字段中，可以设置不允许用户输入数值以外的字符。

解决方案：注意相应的组件在初始化的时候，是否相对应。  
 3、 NumberFormatException：字符串转换为数字类型时抛出的异常。  
  
在数据类型转换过程中，如果是字符型转换为数字型过程中出现的问题，对于这个异常在Java程序中采用了一个独立的异常，即NumberFormatException.如现在讲字符型的数据“123456”转换为数值型数据时，是允许的。但是如果字符型数据中包含了非数字型的字符，如123#56，此时转换为数值型时就会出现异常。系统就会捕捉到这个异常，并进行处理。  
解决方案：检查数据是否正确。

4. java.lang.NullPointerException  
(1).如果使用第三包的jar包，jar包中有错误

(2).没有实例化组件

(3).没有把Adapter和值绑定；

(4)．当页面中某个组件赋值为null时activity会出现错误，程序崩溃后其他页面的数据就会被重新初始化

(5).用Intent传值时，获取的key不存在

(6).没有赋初始值

5. java.lang.ClassNotFoundException　　异常的解释是"指定的类不存在"。  
6. java.lang.ArithmeticException　　这个异常的解释是"数学运算异常"，比如程序中出现了除以零这样的运算就会出这样的异常。  
7. java.lang.ArrayIndexOutOfBoundsException  
　　异常的解释是"数组下标越界"，现在程序中大多都有对数组的操作，因此在调用数组的时候一定要认真检查，看自己调用的下标是不是超出了数组的范围，一般来说，显示（即直接用常数当下标）调用不太容易出这样的错，但隐式（即用变量表示下标）调用就经常出错了，还有一种情况，是程序中定义的数组的长度是通过某些特定方法决定的，不是事先声明的，这个时候，最好先查看一下数组的length，以免出现这个异常。  
8. java.lang.IllegalArgumentException

这个异常的解释是"方法的参数错误"，比如g.setColor(int red,int green,int blue)这个方法中的三个值，如果有超过２５５的也会出现这个异常，因此一旦发现这个异常，我们要做的，就是赶紧去检查一下方法调用中的参数传递是不是出现了错误。  
9. java.lang.IllegalAccessException  
　　这个异常的解释是"没有访问权限"，当应用程序要调用一个类，但当前的方法即没有对该类的访问权限便会出现这个异常。对程序中用了Package的情况下要注意这个异常

10.安全异常：产生的原因是由于当前的设备不支持当前程序，这个是由于机型产生的问题，我们应该换一个设备进行测试，检测当前程序存在什么样的异常；另一种原因是获取激活的网络信息实体类，需要添加权限，否则会出现此异常

解决办法：在配置文件中，添加android\_permission\_Access\_NetWork\_State的权限

11. RejectExcuteException：一个异步任务只能执行一次，否则会报异常

一个线程最多能执行5个线程，超出则等待，否则会报拒绝执行的异常

解决办法：使用子线程+handler来代替AsyncTask，或者一个线程中少添加几个异步操作。

12. Out of memory：内存溢出是存放的数据过多导致内存不足的溢出事件

13. Memory leak：内存泄露是本应该释放的资源没有充分得到释放，就继续添加数据导致内存泄露事件；

Java应用程序中常见的异常类还有很多。如未找到相应类异常、不允许访问某些类异常、文件已经结束异常、文件未找到异常、字段未找到异常等等。一般系统开发人员都可以根据这个异常名来判断当前异常的类型。

ArithmeticException（除数为0的异常）, BufferOverflowException（缓冲区上溢异常）, BufferUnderflowException（缓冲区下溢异常）, IndexOutOfBoundsException（出界异常）, NullPointerException（空指针异常）, EmptyStackException（空栈异常）, IllegalArgumentException（不合法的参数异常）, NegativeArraySizeException, NoSuchElementException, SecurityException, SystemException, UndeclaredThrowableException

**常用的推送，并简单介绍**

并行的都包括：即时信息和空间协议(IMPP)、空间和即时信息协议(PRIM)；

推送的实现原理：

XMPP是一种基于XML的协议，它继承了在XML环境中灵活的发展性。因此，基于XMPP的应用具有超强的可扩展性。XMPP的前身是Jabber，一个开源形式组织产生的网络即时通信协议。所有从一个client到另一个client的jabber消息和数据都要xmpp server，client连接到server，server利用本地目录系统的证书对其认证，client制定目标地址，让server告知目标状态，server查找，连接并进行相互认证，client间进行交互。

XMPP系统特点：

1)客户机/服务器通信模式；(2)分布式网络；(3)简单的客户端；(4)XML的数据格式。

通俗解释：

其实XMPP 是一种很类似于http协议的一种数据传输协议，它的过程就如同“解包装--〉包

装”的过程，用户只需要明白它接受的类型，并理解它返回的类型，就可以很好的利用xmpp来进行数据通讯。

**服务器提供数据前段实现容易出现什么问题**

接口不可用，接口提供的数据格式不是我们想要的。

**133．如何判断那种网络**

1.网络是否连接（包括Wifi和移动网络）

// 是否有可用网络

**private** **boolean** isNetworkConnected() {

        ConnectivityManager cm =

                (ConnectivityManager) mContext.getSystemService(Context.CONNECTIVITY\_SERVICE);

        NetworkInfo network = cm.getActiveNetworkInfo();

**if** (network != **null**) {

**return** network.isAvailable();

        }

**return** **false**;

    }

2.wifi是否可用

// Wifi是否可用

**private** **boolean** isWifiEnable() {

        WifiManager wifiManager = (WifiManager) mContext

                .getSystemService(Context.WIFI\_SERVICE);

**return** wifiManager.isWifiEnabled();

    }

**GPS是否可用**

**[javascript]** [view plain](http://blog.csdn.net/sky837/article/details/7867601" \o "view plain)[copy](http://blog.csdn.net/sky837/article/details/7867601" \o "copy)[print](http://blog.csdn.net/sky837/article/details/7867601" \o "print)[?](http://blog.csdn.net/sky837/article/details/7867601" \o "?)

// Gps是否可用

**private** **boolean** isGpsEnable() {

        LocationManager locationManager =

                ((LocationManager) mContext.getSystemService(Context.LOCATION\_SERVICE));

**return** locationManager.isProviderEnabled(LocationManager.GPS\_PROVIDER);

}

**抽象类和接口的区别**

abstract可以修饰抽象方法，而一个类只要有一个抽象方法，就必须用abstract定义该类，即抽象类。抽象类，被继承，实现它里面的方法，可以只有普通方法

用interface修饰的类，里面的方法都是抽象方法（不能实例化），因此在定义接口的时候，可以直接不加那些修饰，系统会默认的添上去。接口里面的字段都是公有常量，即public static final修

饰的字段。可以没有函数原型，里面可以是空的。

**135. Android中版本代号**

Android 1.0（没有开发代号）

Android 1.1 - Petit Four

Android 1.5 - Cupcake

Android 1.6 - Donut

Android 2.0/2.1 - Éclair

Android 2.2 - Froyo

Android 2.3 - Gingerbread

Android 3.0/3.1/3.2 - Honeycomb

Android 4.0 - Ice Cream Sandwich

Android 4.1/4.2/4.3 - Jelly Bean

Android 4.4 - KitKat

Android 5.0/5.1 – Lollipop

**136. Final和Finally的区别**

final用于声明属性，方法和类，分别表示属性不可变，方法不可覆盖，类不可继承。

finally是异常处理语句结构的一部分，表示总是执行。

finalize是Object类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法，可以覆盖此方法提供垃圾收集时的其他资源回收，例如关闭文件等。

**136. 分享平台有哪些**

ShardSdk、友盟等。

**137. 后台Activity被回收如何恢复**

重写onSaveInstanceState()方法，在此方法中保存需要保存的数据，该方法将会在activity被回收之前调用。通过重写onRestoreInstanceState()方法可以从中提取保存好的数据

**138.Android事件分发的理解**

**事件分发：public boolean dispatchTouchEvent(MotionEvent ev)**

Touch 事件发生时 Activity 的 dispatchTouchEvent(MotionEvent ev) 方法会以隧道方式（从根元素依次往下传递直到最内层子元素或在中间某一元素中由于某一条件停止传递）将事件传递给最外层 View 的 dispatchTouchEvent(MotionEvent ev) 方法，并由该 View 的 dispatchTouchEvent(MotionEvent ev) 方法对事件进行分发。dispatchTouchEvent 的事件分发逻辑如下：

如果 **return true**，事件会分发给当前 View 并由 dispatchTouchEvent 方法进行消费，同时事件会停止向下传递；

如果 **return false**，事件分发分为两种情况：

如果当前 View 获取的事件直接来自 Activity，则会将事件返回给 Activity 的 onTouchEvent 进行消费；

如果当前 View 获取的事件来自外层父控件，则会将事件返回给父 View 的  onTouchEvent 进行消费。

如果返回系统默认的 **super**.dispatchTouchEvent(ev)，事件会自动的分发给当前 View 的 onInterceptTouchEvent 方法。

**▐ 事件拦截：public boolean onInterceptTouchEvent(MotionEvent ev)**

在外层 View 的 dispatchTouchEvent(MotionEvent ev) 方法返回系统默认的 super.dispatchTouchEvent(ev) 情况下，事件会自动的分发给当前 View 的 onInterceptTouchEvent 方法。onInterceptTouchEvent 的事件拦截逻辑如下：

如果 onInterceptTouchEvent 返回 **true**，则表示将事件进行拦截，并将拦截到的事件交由当前 View 的 onTouchEvent 进行处理；

如果 onInterceptTouchEvent 返回 **false**，则表示将事件放行，当前 View 上的事件会被传递到子 View 上，再由子 View 的 dispatchTouchEvent 来开始这个事件的分发；

如果 onInterceptTouchEvent 返回 **super**.onInterceptTouchEvent(ev)，事件默认会被拦截，并将拦截到的事件交由当前 View 的 onTouchEvent 进行处理。

**▐ 事件响应：public boolean onTouchEvent(MotionEvent ev)**

在 dispatchTouchEvent 返回 super.dispatchTouchEvent(ev) 并且 onInterceptTouchEvent 返回 true 或返回 super.onInterceptTouchEvent(ev) 的情况下 onTouchEvent 会被调用。onTouchEvent 的事件响应逻辑如下：

如果事件传递到当前 View 的 onTouchEvent 方法，而该方法返回了 false，那么这个事件会从当前 View 向上传递，并且都是由上层 View 的 onTouchEvent 来接收，如果传递到上面的 onTouchEvent 也返回 false，这个事件就会“消失”，而且接收不到下一次事件。

如果返回了 true 则会接收并消费该事件。

如果返回 super.onTouchEvent(ev) 默认处理事件的逻辑和返回 false 时相同。

到这里，与 Touch 事件相关的三个方法就分析完毕了。

**139. 线程的几种状态**

1)新建状态(New)：新创建了一个线程对象。

　2)就绪状态(Runnable)：线程对象创建后，其他线程调用了该对象的start()方法。该状态的线程位于可运行线程池中，变得可运行，等待获取[CPU](http://product.it168.com/list/b/0217_1.shtml)的使用权。

　　3)运行状态(Running)：就绪状态的线程获取了CPU，执行run()方法。

　　4)阻塞状态(Blocked)：阻塞状态是线程因为某种原因放弃CPU使用权，暂时停止运行。直到线程进入就绪状态，才有机会转到运行状态。阻塞的情况分三种：

等待阻塞：运行的线程执行wait()方法，JVM会把该线程放入等待池中。

同步阻塞：运行的线程在获取对象的同步锁时，若该同步锁被别的线程占用，则JVM会把该线程放入锁池中。

其他阻塞：运行的线程执行sleep()或join()方法，或者发出了I/O请求时，JVM会把该线程置为阻塞状态。当sleep()状态超时、join()等待线程终止或者超时、或者I/O处理完毕时，线程重新转入就绪状态。

5)死亡状态(Dead)：线程执行完了或者因异常退出了run()方法，该线程结束生命周期。

当调用start方法的时候，该线程就进入就绪状态。等待CPU进行调度执行，此时还没有真正执行线程。

**140. Ddms和Traceview的区别**

DDMS是一个程序执行查看器，在里面可以看见线程和堆栈等信息，TraceView是程序性能分析器。

**150. PathImage是什么**

.9.PNG是安卓开发里面的一种特殊的图片，这种格式的图片通过ADT自带的编辑工具生成，使用九宫格切分的方法，使图片支持在android 环境下的自适应展示。

**151. Android项目的目录结构**

1)src目录：src（即SourceCode)，src目录用来存放应用程序中所有的源代码，代码的源文件一般存放在该目录下的相应的包下。

2)gen目录：gen目录，该目录下一般只有一个文件，即R文件。该文件夹下面有个R.java文件，R.java是在建立项目时自动生成的，定义该项目所有资源的索引。

3)Android4.1目录：该文件夹下包含android.jar文件，这是一个Java归档文件，其中包含构建应用程序所需的所有的AndroidSDK库（如Views、Controls）和APIs。

通过android.jar将自己的应用程序绑定到AndroidSDK和AndroidEmulator，这允许你使用所有Android的库和包，且使你的应用程序在适当的环境中调试。

4)assets目录：该目录存放应用程序中使用的外部资源文件，如声音视频等，在Android程序中可以通过输入/输出流对该目录中的文件进行读写。

5)res目录：该目录用来存放程序这用到的图片、界面布局文件及XML格式的描述文件。该目录下有多个目录。新建一Android项目，在res目录下一般会有drawable,layout,valuse三个目录。drawable用来存放图片资源,目录下一般会有三个目录分别存放高中低三种不同分辨率的图片，layout目录用来存放应用程序的界面布局文件，values目录下存放字符串资源，颜色资源，数组资源等XML文件。

6)AndroidManifest.xml

**152. Intentservice的用法**

androidservicethreadstringclassnullIntentService实现原理及内部代码：

Android中除了Service还有一个IntentService，他们之间到底有哪些区别呢?在继承关系上而言IntentService是Service的子类，内部实现的代码中涉及到一些Android入门开发者不了解的Looper，Android123在早期的文章中已经说明他们的用法，这里不再赘述，有关原理大家可以看源码实现如下:

1. publicabstractclassIntentServiceextendsService{
2. privatevolatileLoopermServiceLooper;
3. privatevolatileServiceHandlermServiceHandler;//一个Handler封装了Looper对象
4. privateStringmName;
5. privatebooleanmRedelivery;
6. privatefinalclassServiceHandlerextendsHandler{
7. publicServiceHandler(Looperlooper){
8. super(looper);
9. }
10. @Override
11. publicvoidhandleMessage(Messagemsg){
12. onHandleIntent((Intent)msg.obj);
13. stopSelf(msg.arg1);
14. }
15. }
16. publicIntentService(Stringname){//构造方法，需要提供一个name作为标识
17. super();
18. mName=name;
19. }
20. 对于下面的setIntentRedelivery的参数，如果为真时
21. onStartCommand(Intent,int,int)}willreturn
22. Service#START\_REDELIVER\_INTENT},soifthisprocessdiesbefore
23. onHandleIntent(Intent)}returns,theprocesswillberestarted
24. 如果为假时
25. onStartCommand(Intent,int,int)}willreturn
26. Service#START\_NOT\_STICKY},andiftheprocessdies
27. publicvoidsetIntentRedelivery(booleanenabled){
28. mRedelivery=enabled;
29. }
30. @Override
31. publicvoidonCreate(){//这里重写父类Service的创建，主要是构造一个线程
32. super.onCreate();
33. HandlerThreadthread=newHandlerThread("IntentService["+mName+"]");
34. thread.start();
35. mServiceLooper=thread.getLooper();
36. mServiceHandler=newServiceHandler(mServiceLooper);
37. }
38. @Override
39. publicvoidonStart(Intentintent,intstartId){//Android2.0以前的Service启动参数控制
40. Messagemsg=mServiceHandler.obtainMessage();
41. msg.arg1=startId;
42. msg.obj=intent;
43. mServiceHandler.sendMessage(msg);
44. }
45. @Override
46. publicintonStartCommand(Intentintent,intflags,intstartId){//Android2.0以后的服务启动参数
47. onStart(intent,startId);
48. returnmRedelivery?START\_REDELIVER\_INTENT:START\_NOT\_STICKY;
49. }
50. @Override
51. publicvoidonDestroy(){//服务摧毁的时候Looper一定要释放掉，这点很重要。
52. mServiceLooper.quit();
53. }
54. @Override
55. publicIBinderonBind(Intentintent){
56. returnnull;
57. }
58. protectedabstractvoidonHandleIntent(Intentintent);
59. }
60. 从上面的代码相信大家可以看出IntentService和Service的不同了，通过Looper和Thread来解决标准Service中处理逻辑的阻塞问题，毕竟Android的Service也是会阻塞的。
61. 关于使用
62. IntentService是Service类的子类，用来处理异步请求。客户端可以通过startService(Intent)方法传递请求给IntentService，IntentService通过workerthread处理每个Intent对象，执行完所有的工作之后自动停止Service。
63. 说明：workerthread处理所有通过传递过来的请求，创建一个workerqueue，一次只传递一个intent到onHandleIntent中，从而不必担心多线程带来的问题。处理完毕之后自动调用stopSelf()方法；默认实现了Onbind（）方法，返回值为null；
64. 模式实现了哦你StartCommand()方法，这个方法会放到workerqueue中，然后在onHandleIntent()中执行0
65. 使用IntentService需要两个步骤：
66. 1、写构造函数
67. 2、复写onHandleIntent()方法
68. 好处：处理异步请求的时候可以减少写代码的工作量，比较轻松地实现项目的需求
69. 用Service来处理后台耗时操作，却很少注意到还有个IntentService
70. 首先IntentService是继承自Service的，那我们先看看Service的官方介绍，这里列出两点比较重要的地方：
71. 1.AServiceisnotaseparateprocess.TheServiceobjectitselfdoesnotimplyitisrunninginitsownprocess;unlessotherwisespecified,itrunsinthesameprocessastheapplicationitispartof.
72. 2.AServiceisnotathread.Itisnotameansitselftodoworkoffofthemainthread(toavoidApplicationNotRespondingerrors).
73. 稍微翻一下(英文水平一般)
74. 1.Service不是一个单独的进程，它和应用程序在同一个进程中。
75. 2.Service不是一个线程，所以我们应该避免在Service里面进行耗时的操作
76. 关于第二点我想说下，不知道很多网上的文章都把耗时的操作直接放在Service的onStart方法中，而且没有强调这样会出现ApplicationNotResponding！希望我的文章能帮大家认清这个误区(Service不是一个线程，不能直接处理耗时的操作)。
77. 有人肯定会问，那么为什么我不直接用Thread而要用Service呢？关于这个，大家可以网上搜搜，这里不过多解释。有一点需要强调，如果有耗时操作在Service里，就必须开启一个单独的线程来处理！！！这点一定要铭记在心。
78. IntentService相对于Service来说，有几个非常有用的优点，首先我们看看官方文档的说明：
79. IntentServiceisabaseclassforServicesthathandleasynchronousrequests(expressedasIntents)ondemand.ClientssendrequeststhroughstartService(Intent)calls;theserviceisstartedasneeded,handleseachIntentinturnusingaworkerthread,andstopsitselfwhenitrunsoutofwork.
80. This"workqueueprocessor"patterniscommonlyusedtooffloadtasksfromanapplication'smainthread.TheIntentServiceclassexiststosimplifythispatternandtakecareofthemechanics.Touseit,extendIntentServiceandimplementonHandleIntent(Intent).IntentServicewillreceivetheIntents,launchaworkerthread,andstoptheserviceasappropriate.
81. Allrequestsarehandledonasingleworkerthread--theymaytakeaslongasnecessary(andwillnotblocktheapplication'smainloop),butonlyonerequestwillbeprocessedatatime.
82. 稍微翻译理一理，这里主要是说IntentService使用队列的方式将请求的Intent加入队列，然后开启一个workerthread(线程)来处理队列中的Intent，对于异步的startService请求，IntentService会处理完成一个之后再处理第二个，每一个请求都会在一个单独的workerthread中处理，不会阻塞应用程序的主线程，这里就给我们提供了一个思路，如果有耗时的操作与其在Service里面开启新线程还不如使用IntentService来处理耗时操作。下面给一个小例子：
83. 1Service：
84. packagecom.zhf.service;
85. importAndroid.app.Service;
86. importAndroid.content.Intent;
87. importAndroid.os.IBinder;
88. publicclassMyServiceextendsService{
89. @Override
90. publicvoidonCreate(){
91. super.onCreate();
92. }
93. @Override
94. publicvoidonStart(Intentintent,intstartId){
95. super.onStart(intent,startId);
96. //经测试，Service里面是不能进行耗时的操作的，必须要手动开启一个工作线程来处理耗时操作
97. System.out.println("onStart");
98. try{
99. Thread.sleep(20000);
100. }catch(InterruptedExceptione){
101. e.printStackTrace();
102. }
103. System.out.println("睡眠结束");
104. }
105. @Override
106. publicIBinderonBind(Intentintent){
107. returnnull;
108. }
109. }
110. 2.IntentService：
111. packagecom.zhf.service;
112. importAndroid.app.IntentService;
113. importAndroid.content.Intent;
114. publicclassMyIntentServiceextendsIntentService{
115. publicMyIntentService(){
116. super("yyyyyyyyyyy");
117. }
118. @Override
119. protectedvoidonHandleIntent(Intentintent){
120. //经测试，IntentService里面是可以进行耗时的操作的
121. //IntentService使用队列的方式将请求的Intent加入队列，然后开启一个workerthread(线程)来处理队列中的Intent
122. //对于异步的startService请求，IntentService会处理完成一个之后再处理第二个
123. System.out.println("onStart");
124. try{
125. Thread.sleep(20000);
126. }catch(InterruptedExceptione){
127. e.printStackTrace();
128. }
129. System.out.println("睡眠结束");
130. }
131. }
132. 测试主程序：
133. packagecom.zhf.service;
134. importAndroid.app.Activity;
135. importAndroid.content.Intent;
136. importAndroid.os.Bundle;
137. publicclassServiceDemoActivityextendsActivity{
138. /\*\*Calledwhentheactivityisfirstcreated.\*/
139. @Override
140. publicvoidonCreate(BundlesavedInstanceState){
141. super.onCreate(savedInstanceState);
142. setContentView(R.layout.main);
143. startService(newIntent(this,MyService.class));//主界面阻塞，最终会出现Applicationnotresponding
144. //连续两次启动IntentService，会发现应用程序不会阻塞，而且最重的是第二次的请求会再第一个请求结束之后运行(这个证实了IntentService采用单独的线程每次只从队列中拿出一个请求进行处理)
145. startService(newIntent(this,MyIntentService.class));
146. startService(newIntent(this,MyIntentService.class));
147. }
148. }

**153. IntentService有何优点?**

答案：IntentService使用队列的方式将请求的Intent加入队列，然后开启一个worker thread(线程)来处理队列中的Intent，对于异步的startService请求，IntentService会处理完成一个之后再处理第二个，每一个请求都会在一个单独的worker thread中处理，不会阻塞应用程序的主线程，这里就给我们提供了一个思路，如果有耗时的操作与其在Service里面开启新线程还不如使用IntentService来处理耗时操作。

**154. 如何启用service，如何停用service？**

答案：①：通过调用Context.startservice()启动，调用Context.stopservice()结束，startservice()可以传递参数给service.

②：通过调用context.bindservice()启动，调用Context.unbindservice()结束，还可以通过serviceConnection访问service。

**155. 解释下单线程中message、handler、Message Queue、Looper之间的关系。**

答案：简单来说，Handler获取当前线程中的looper对象，looper用来从存放Message说的MessageQueue中获取Message，再有Handler进行Message的分发和处理。

**156. AIDL全称是什么？能处理哪些数据？**

答案：全称是：Android Interface Definition Landuage是一种街口描述语言；编译器可以通过aidl文件生成一段代码，通过预先定义的接口达到两个进程内部通信进程跨界对象访问的目的，aidl的IPC的机制和COM或CORBA类似，是基于接口的，是轻量级的。使用代理类在客户端和实现层间接传递值。

**157. Activity的启动模式有哪些？**

答案：①、standard，每次激活activity时，都创建activity实例，并放入任务栈。

②、singleTop，如果activity自己激活自己，即任务栈栈顶就是该activity，则不需要创建，其余情况都要创建activity实例。

③、singleTask，如果要激活的那个activity在任务栈中存在该实例，则不需要创建，只需要把此activity放入栈顶，并把该activity以上的activity实例都pop；

④、singleInstance，如果应用1的任务栈中创建了MainActivity实例，如果应用2也要激活MainActivity，则不需要创建，两应用共享该Activity实例；

**158. 如何评价Android？**

Android平台的五大优势：

开放性、挣脱运营商的束缚、丰富的硬件选择、不受任何限制的开发商、无缝结合的Google应用

Android平台的五大不足：

1. 安全和隐私
2. 首先开卖Android手机的不是最大运营商
3. 运营商仍然能够影响到Android手机
4. 同类机型用户减少
5. 过分依赖开发商缺少标准配置

**159. 断点续传**

断点续传下载，其实就是记录下原来下载的位置，然后从那个位置开始下载就是了。

**160. Dvm是什么**

Android 运行环境主要指的虚拟机技术——Dalvik。Android中的所有Java程序都是运行在Dalvik VM上的。Android上的每个程序都有自己的线程，DVM只执行.dex的Dalvik executable 文件。每个Android应用在底层都对应有一个独立的DVM实例并在其解释下执行。

**170. 图片三级缓存的机制**

实现图片缓存也不难，需要有相应的cache策略。这里我采用 内存-文件-网络 三层cache机制，其中内存缓存包括强引用缓存和软引用缓存（SoftReference），其实网络不算cache，这里姑且也把它划到缓存的层次结构中。当根据url向网络拉取图片的时候，先从内存中找，如果内存中没有，再从缓存文件中查找，如果缓存文件中也没有，再从网络上通过http请求拉取图片。在键值对（key-value）中，这个图片缓存的key是图片url的hash值，value就是bitmap。所以，按照这个逻辑，只要一个url被下载过，其图片就被缓存起来了。

**171. 你看好的行业领域**

穿戴

**172. 如何控制耗电量**

在以下情况下耗电比较多：

1、 大数据量的传输。

2、 不停的在网络间切换。

3、 解析大量的文本数据。

**1、 在需要网络连接的程序中，首先检查网络连接是否正常，如果没有网络连接，那么就不需要执行相应的程序。**

**2、 使用效率高的数据格式和解析方法。(fastJson,LoganSquare-master)**

3、使用GZIP压缩方式下载数据，能减少网络流量

4、 其它一些优化方法：  
回收java对象，特别是较大的java对像

对定位要求不是太高的话尽量不要使用GPS定位，可能使用wifi和移动网络cell定位即可。GPS定位消耗的电量远远高于移动网络定位。

尽量不要使用浮点运算。

获取屏幕尺寸等信息可以使用缓存技术，不需要进行多次请求。

**173. Service用在哪个线程**

默认情况下Service是运行在启动该Service的应用主线程的，如果Service中的操作占用大量的CPU资源或有阻断操作（比如播放MP3或者访问网络）会影响应用主线程的响应性能，甚至会造成“应用程序无响应(ANR)”问题。

**174. ContentProvider和SQL的区别**

Sql只能在该工程的内部共享数据，ContentProvider能在工程之间实现数据共享。

**175.Android如何访问自定义ContentProvider**

1.第一：得到ContentResolver类对象：ContentResolver cr = getContentResolver（）；

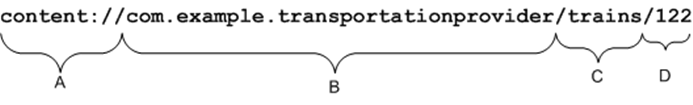
2. 第二：定义要查询的字段String数组。

3. 第三：使用cr.query();返回一个Cursor对象。

4. 第四：使用while循环得到Cursor里面的内容。

**176.Uri有什么组成**

答：URI由是部分组成：



A，标准的前缀: ContentProvider（内容提供者）的scheme已经由Android所规定， scheme为：content://

B，唯一标识整个Content Provider: 主机名（或叫Authority）用于唯一标识这个ContentProvider，外部调用者可以根据这个标识来找到它。

C，描述了数据的路径，确定返回哪类数据: 路径（path）可以用来表示我们要操作的数据，路径的构建应根据业务而定

1. ID唯一标注请求的数据: 我们要请求数据的ID

**177. 对android系统的理解**

5大优势：

1. 开放性
2. 挣脱运营商的束缚
3. 丰富的硬件选择
4. 不受任何限制的开发商
5. 无缝结合的Google应用

Android的5大不足

1. 安全和隐私
2. 首先开卖Android手机的不是最大运营商
3. 运营商仍然能够影响到Android手机
4. 同类机型用户减少
5. 过分依赖开发商缺少标准配置

**178. 简单介绍服务**

服务是没有界面的长生命周期的代码。一个很好的例子是媒体播放器从列表中播放歌曲。在一个媒体播放器程序中，大概要有一个或多个活动（activity）来供用户选择歌曲并播放它。然而，音乐的回放就不能使用活动（activity）了，因为用户希望他导航到其他界面时音乐继续播放。这种情况下，媒体播放器活动（activity）要用Context.startService()启动一个服务来在后台运行保持音乐的播放。系统将保持这个音乐回放服务的运行直到它结束。注意一下，你要用Context.bindService()方法连接服务（如果它没有运行，要先启动它）。当连接到服务后，你可以通过服务暴露的一个接口和它通信。对于音乐服务，它允许你暂停、倒带，等等。

**179. 一条最长短信约占多少byte**

160

**180. 上拉加载下拉刷新如何实现**

答：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，调用 addFooterView，添加记录到adapter, adapter调用 notifyDataSetChanged 更新数据;如果没有记录了，把自定义的mFooterView去掉。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

**181.手机适配如何做，自适应**

答:

在布局中设置控件尺寸是尽量使用fill-parent,warp-content,dp等值，文字使用sp

尽量使用Relativelayout 。

对不同的屏幕提供不同大小的图片，

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。

**182. Manifest中放什么东西**

AndroidManifest.xml是每个android程序中必须的文件。它位于整个项目的根目录，描述了package中暴露的组件（activities,services,等等），他们各自的实现类，各种能被处理的数据和启动位置。

除了能声明程序中的Activities,ContentProviders,Services,和IntentReceivers,还能指定

permissions和instrumentation（安全控制和测试）

**183. 自定义组件的实现思路**

答: 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. 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

**184. 项目中的混淆打包**

a在项目的project.properties文件中加入如下代码：

proguard.config=proguard.cfg即可

好处：1.隐藏代码，避免被反编译2.减少代码提交，提高效率4.android中四大组件不能被混淆，所用在清单文件中声明的都是被混淆5.native不能被混淆

**185. 什么是020，b2b，b2c，**

http://blog.sina.com.cn/s/blog\_64e090b001016843.html

**186.如何进行手势判断**

用onTouchEvent(ev)判断手势

**187. Xutil的用法**

http://www.csdn123.com/html/topnews201408/82/6182.htm

**188. ContentProvider Uri 的组成**



**189. 数据库如何写sql语句**

公司框架中中有对数据库的增删改查，有时候自己也写一些复杂的sql语句】

**190. 电商领域的核心**

订单，支付，购物，和信息安全

**191. Asynctask的原理**

AsyncTask的本质是一个线程池，所有提交的异步任务都会在这个线程池中的工作线程内执行，当工作线程需要跟UI线程交互时，工作线程会通过向在UI线程创建的Handler传递消息的方式，调用相关的回调函数，从而实现UI界面的更新。

**192.asset和res的区别**

答：res/raw和assets的相同点：

两者目录下的文件在打包后会原封不动的保存在apk包中，不会被编译成二进制。

res/raw和assets的不同点：

1）res/raw中的文件会被映射到R.java文件中，访问的时候直接使用资源ID即 R.raw.filename；assets文件夹下的文件不会被映射到R.java中，访问的时候需要AssetManager类。

2）res/raw不可以有目录结构，而assets则可以有目录结构，也就是assets目录下可以再建立文件夹

3）读取文件资源举例：

读取res/raw下的文件资源，通过以下方式获取输入流来进行写操作

InputStream is = getResources().openRawResource(R.raw.filename);

读取assets下的文件资源，通过以下方式获取输入流来进行写操作

AssetManager am = null;

am = getAssets();

InputStream is = am.open("filename");

**193. 瀑布流如何实现**

就是异步下载图片的时候把图片添加到scrollview中，

你需要几列就在scrollview中放置几个linearlayout，在scrollview滑动的的时候动态的测量linearlayout随机生成linearlayout的高度

**194. 图片的内存不足**

可能是你加载的当前的图片占用内存大于当前剩余内存，解决方法就是及时释放内存，不显示的图片要释放掉。也可能是你加载的图片是空的，导致内存不足。android系统给图片分配的内存只有8M,当加载大量图片时往往会出现OOM。

Android加载大量图片内存溢出解决方案：

1）尽量不要使用setImageBitmap或setImageResource或BitmapFactory.decodeResource来设置一张大图，因为这些函数在完成decode后，最终都是通过java层的createBitmap来完成的，需要消耗更多内存，可以通过BitmapFactory.decodeStream方法，创建出一个bitmap，再将其设为ImageView的 source

2）使用BitmapFactory.Options对图片进行压缩

InputStream is = this.getResources().openRawResource(R.drawable.pic1);  
 BitmapFactory.Options options=new BitmapFactory.Options();  
 options.inJustDecodeBounds = false;  
 options.inSampleSize = 10; //width，hight设为原来的十分一  
 Bitmap btp =BitmapFactory.decodeStream(is,null,options);   
3）运用Java软引用，进行图片缓存，将需要经常加载的图片放进缓存里，避免反复加载

及时销毁不再使用的Bitmap对象

if(!bmp.isRecycle() ){  
 bmp.recycle() //回收图片所占的内存  
 system.gc() //提醒系统及时回收  
}

**195. Http和Https有什么区别**

http和https使用的是完全不同的连接方式,用的端口也不一样,前者是80,后者是443。http的连接很简单,是无状态的,... HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议 要比http协议安全

**196. 相册缩略图如何实现**

public class Thumbnail extends Activity {

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

try {

saveMyBitmap("ooo");

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public void saveMyBitmap(String bitName) throws IOException {

File originalFile = new File("sdcard/pic/ll.jpg");

Bitmap bmp = decodeFile(originalFile);

File f = new File("/sdcard/" + bitName + ".jpg");

f.createNewFile();

FileOutputStream fOut = null;

try {

fOut = new FileOutputStream(f);

} catch (FileNotFoundException e) {

e.printStackTrace();

}

//BitmapFactory.Options options=new BitmapFactory.Options();

        //options.inSampleSize = 10;

//options.inTempStorage = new byte[16\*1024];

//Bitmap bmp = BitmapFactory.decodeFile("/sdcard/pic/sd.jpg");

//Bitmap bmp = BitmapFactory.decodeFile("/sdcard/pic/ll.jpg", options);

//bmp = Bitmap.createScaledBitmap(bmp, 800, 480, true);

bmp.compress(Bitmap.CompressFormat.JPEG, 30, fOut);

try {

fOut.flush();

} catch (IOException e) {

e.printStackTrace();

}

try {

fOut.close();

} catch (IOException e) {

e.printStackTrace();

}

}

//decodes image and scales it to reduce memory consumption

private Bitmap decodeFile(File f){

   try {

       //Decode image size

       BitmapFactory.Options o = new BitmapFactory.Options();

       o.inJustDecodeBounds = true;

       BitmapFactory.decodeStream(new FileInputStream(f),null,o);

       //The new size we want to scale to

       final int REQUIRED\_HEIGHT=800;

       final int REQUIRED\_WIDTH=480;

       //Find the correct scale value. It should be the power of 2.

       int width\_tmp=o.outWidth, height\_tmp=o.outHeight;

       System.out.println(width\_tmp+"  "+height\_tmp);

       Log.w("===", (width\_tmp+"  "+height\_tmp));

       int scale=1;

       while(true){

           if(width\_tmp/2<REQUIRED\_WIDTH && height\_tmp/2<REQUIRED\_HEIGHT)

               break;

           width\_tmp/=2;

           height\_tmp/=2;

           scale++;

           Log.w("===", scale+"''"+width\_tmp+"  "+height\_tmp);

       }

       //Decode with inSampleSize

       BitmapFactory.Options o2 = new BitmapFactory.Options();

       o2.inSampleSize=scale;

       return BitmapFactory.decodeStream(new FileInputStream(f), null, o2);

   } catch (FileNotFoundException e) {}

   return null;

}

}

**197.View更新页面的五种方式**

Activity.runOnUiThread(Runnable)

View.post(Runnable)

View.postDelayed(Runnable,long);

View.postInvalidate();

View.Invalidate();

**198.Android程序的入口**

action节点中的android.intent.action.MAIN表明它所在的Activity是整个应用程序的入口点

**199. 高德地图了解**

地图渲染技术：性能提升10倍，所占空间降低80%，比传统地图软件节省流量超过90%

专业在线导航功能：覆盖全国364个城市、全国道路里程352万公里

在线[导航](http://baike.baidu.com/view/425408.htm" \t "http://baike.baidu.com/_blank)功能：最新[高德](http://baike.baidu.com/view/876937.htm" \t "http://baike.baidu.com/_blank)在线导航引擎，全程语音指引提示，完善偏航判定和偏航重导功能。

AR虚拟实景：AR功能结合手机摄像头和用户位置、方向等信息，将信息点以更直观的方式展现给用户，为发现和目标地点指引

丰富的出行查询功能：地名信息查询、分类信息查询、公交换乘、驾车路线规划、公交线路查询、位置收藏夹等丰富的基础地理信息查询工具。

锁屏语音提示，即使你的iPhone在锁屏状态也能听到高德导航的语音提示，不用担心一直开着iPhone屏幕费电了。

夜间导航HUD抬头提示，打开高德导航并开启[HUD](http://baike.baidu.com/view/1122312.htm" \t "http://baike.baidu.com/_blank)，把iPhone放到汽车挡风玻璃下，高德导航会把路线提示倒映到汽车挡风玻璃上，看起来特别方便，不用担心低头看手机影响驾驶了。

**动态导航**

交通路况实时播报

智能计算到达目的地所需的时间

避堵路线方案规划

摄像头提醒

**离线下载**

2D、3D离线地图

分地区下载地图包

全国地图包、全国概要图

**地图搜索**

热门地点、线路搜索

公交、自驾出行线路规划

公交、火车、天气查询服务

**全新引擎**

最新3D版本，360度旋转视角

矢量数据传送

观看更流畅、更清晰

**兴趣点**

餐饮、住宿、优惠、

演出、团购全覆盖

海量兴趣点随意搜

**叫车服务**

2013年7月,高德地图更新，新增打车功能，综合快的打车、嘀嘀打车两家资源，覆盖80%市场份额。与阿里合作后，高德地图凭借高市场占有率，逐渐从单一地图向LBS工具箱转变。新版高德地图不仅增加了可预约的出租车数量，提高了打车效率，覆盖城市也扩展到26个

**200. Activity中 this和getActivity区别**

This指的是Activity当前的上下文

getActivity()指的是在fragment当中调用得到他所在的Activity

**201. 单例模式和抽象工厂各自的优点**

单例模式：减少创建对象的次数，对内存的管理更方便，占用内存更少

抽象工厂：把公共的部分抽取出，提高了代码的复用性，也减少了代码量，便于管理。

**202. Android的几种缓存方式**

内存缓存，文件缓存，网络缓存

**203. 简单讲解一下递归**

[递归算法](http://baike.baidu.com/view/1733593.htm" \t "http://baike.baidu.com/_blank)一般用于解决三类问题：

(1)数据的定义是按递归定义的。（Fibonacci[函数](http://baike.baidu.com/view/15061.htm" \t "http://baike.baidu.com/_blank)）

(2)问题解法按[递归算法](http://baike.baidu.com/view/1733593.htm" \t "http://baike.baidu.com/_blank)实现。

这类问题虽则本身没有明显的递归结构，但用递归求解比迭代求解更简单，如Hanoi问题。

(3)数据的结构形式是按递归定义的。

如二叉树、广义表等，由于结构本身固有的递归特性，则它们的操作可递归地描述。

递归的缺点：

[递归算法](http://baike.baidu.com/view/1733593.htm" \t "http://baike.baidu.com/_blank)解题相对常用的算法如普通循环等，运行效率较低。因此，应该尽量避免使用递归，除非没有更好的[算法](http://baike.baidu.com/view/7420.htm" \t "http://baike.baidu.com/_blank)或者某种特定情况，递归更为适合的时候。在[递归调用](http://baike.baidu.com/view/1265506.htm" \t "http://baike.baidu.com/_blank)的过程当中系统为每一层的返回点、局部量等开辟了栈来存储。递归次数过多容易造成[栈溢出](http://baike.baidu.com/view/2061755.htm" \t "http://baike.baidu.com/_blank)等。

递归典型问题： 梵塔问题（汉诺塔问题）

已知有三根针分别用A, B, C表示，在A中从上到下依次放n个从小到大的盘子，现要求把所有的盘子

从A针全部移到B针，移动规则是：可以使用C临时存放盘子，每次只能移动一块盘子，而且每根针上

不能出现大盘压小盘，找出移动次数最小的方案.

**204. 如何根据权重实现布局**

//权重和父容器orientation有关

horizontal 指水平方向权重  android:layout\_width

vertical  指垂直方向权重   android:layout\_height

Layout\_weight是线性布局，也就是LinearLayout里面用到的

**在layout\_width设置为fill\_parent的时候，layout\_weight代表的是你的控件要优先尽可能的大,但尽可能大是有限度的，即fill\_parent**.

**在layout\_width设置为wrap\_content的时候，layout\_weight代表的是你的控件要优先尽可能的小,但这个小是有限度的，即wrap\_content.**

**205.在Activity中This和GetApplictionContext的区别**

getApplicationContext() 返回应用的上下文，生命周期是整个应用，应用摧毁它才摧毁  
Activity.this的context 返回当前activity的上下文，属于activity ，activity 摧毁他就摧毁  
getBaseContext() 返回由构造函数指定或setBaseContext()设置的上下文  
this.getApplicationContext（）取的是这个应 用程序的Context，Activity.this取的是这个Activity的Context，这两者的生命周期是不同 的，前者的生命周期是整个应用，后者的生命周期只是它所在的Activity。

1. **Android中项目升级注意的问题**

在清单文件里面的android:versionCode和android:versionName有版本文件, 客户端同时存储该应用当前版本号 (SharedPreferences/Sqlite), 每次打开应用,去检测服务器版本号与本地版本号是否一致,如果不一 致,则自定义对话框提示是否下载更新

1. **Xml解析有几种，分别是什么**

三种分别为：sax,pull,dom

dom解析：解析器读入整个文档，然后构建一个驻留内存的树结构，然后就可以使用 DOM 接口来操作这个树结构。优点是对文档增删改查比较方便，缺点占用内存比较大。

sax解析：基于事件驱动型,优点占用内存少，解析速度快，缺点是只适合做文档的读取，不适合做文档的增删改，不能中途停止。

pull解析：同样基于事件驱动型,android 官方API提供,可随时终止,调用next() 方法提取它们

1. **播放视频的实现方式**

1）使用系统自带的播放器来播放，指定Action为ACTION\_VIEW,Data为Uri，Type为其MIME类型。

//调用系统自带的播放器

Intent intent = new Intent(Intent.ACTION\_VIEW);

intent.setDataAndType(uri, "video/mp4");

startActivity(intent);

2)使用VideoView组件来播放, 可以结合MediaController来实现播控, 只是不能随意更改视频的大小及位置。

3)使用MediaPlayer和SurfaceView来实现，这种方式很灵活,可以自定义视频播放的大小和位置。

1. **IPC是什么**

IPC（Inter-Process Communication，进程间通信）,aidl是 Android Interface definition language的缩写，它是一种android内部进程通信接口的描述语言，通过它我们可以定义进程间的通信接口.编译器可以通过扩展名为aidl的文件生成一段代码，通过预先定义的接口达到两个进程内部通信进程的目的.

BroadcastReceiver也可以实现进程间通信

ContentProvider 提供进程间数据共享

1. **Viewpager是如何使用**

我们可以做很多事情，从最简单的导航，到页面菜单等等。那如何使用它呢，与LisstView类似，我们也需要一个适配器，他就是PagerAdapter，也可以与fragment结合使用

1. **如何自定义一个Adapter**  
   继承自BaseAdapter实现里面的方法，listView在开始绘制的时候，系统首先调用getCount（）函数，根据他的返回值得到listView的长度，然后根据这个长度，调用getView（）逐一绘制每一行。如果你的getCount（）返回值是0的话，列表将不显示同样return 1，就只显示一行。系统显示列表时，首先实例化一个适配器（这里将实例化自定义的适配器）。当手动完成适配时，必 须手动映射数据，这需要重写getView（）方法。系统在绘制列表的每一行的时候将调用此方法。getView()有三个参数，position表示将显示的是第几行，covertView是从布局文件中inflate来的 布局。我们用LayoutInflater的方法将定义好的main.xml文件提取成View实例用来显示。然后 将xml文件中的各个组件实例化（简单的findViewById()方法）。这样便可以将数据对应到各个组件上了。但是按钮为了响应点击事件，需要为它添加点击监听器，这样就能捕获点击事件。至此一个自定 义的listView就完成了，现在让我们回过头从新审视这个过程。系统要绘制ListView了，他首先获得 要绘制的这个列表的长度，然后开始绘制第一行，怎么绘制呢？调用getView()函数。在这个函数里面 首先获得一个View（实际上是一个ViewGroup），然后再实例并设置各个组件，显示之。好了，绘制完这一行了。那 再绘制下一行，直到绘完为止。在实际的运行过程中会发现listView的每一行没有焦点了，这是因为Button抢夺了listView的焦点，只要布局文件中将Button设置为没有焦点就OK了。