目录

1、方法重载与覆盖的区别?(Overload 与 Override 的区别)

答: 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。

2、String 和 StringBuffer 的区别

答：STRING 的长度是不可变的，STRINGBUFFER 的长度是可变的。如果你对字符串中的内容经常进行操作,特别是内容要修改时,那么使用 StringBuffer,如果最后需要 String，那么使用 StringBuffer 的 toString()方法。

3、字符串“abcde”通过写一个函数不让调用第三方的字符串，实现一个字符串倒序，比如字符串“abcde” 变成“edcba”

答： String src = ″ABCDEF ″;

String dst = new StringBuffer(src).reverse().toString();

4、抽象类与接口的区别（abstract 与 interface 的区别）

答：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。

5、集合的实现类与区别?

答:Collection 接口，集合结构总的父接口，有两个子接口 list 和 set

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

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，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

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

6、线程有几种状态,分别是哪些?（调用 run()和调用 start()的区别）

答：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 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：

(一)、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。

(二)、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。

(三)、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。

当调用 run 方法的时候，是已经被 CPU 进行调度，执行线程的主要任务。

7、线程的实现方式

答：线程的实现有两种方式，一是继承 Thread 类，二是实现 Runnable 接口

8、sleep() 与 wait()的区别

答：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

9、线程中 wait，join，sleep，yield, notify，notifyall，synchronized，区别及联系

答：1).sleep()方法

在指定时间内让当前正在执行的线程暂停执行，但不会释放“锁标志”。不推荐使用。sleep()使当前线程进入阻塞状态，在指定时间内不会执行。

2).wait()方法

在其他线程调用对象的 notify 或 notifyAll 方法前，导致当前线程等待。线程会释放掉它所占有的“锁标志”，从而使别的线程有机会抢占该锁。

唤醒当前对象锁的等待线程使用 notify 或 notifyAll 方法,waite() 和 notify()必须在 synchronized 函数或 synchronized block 中进行调用。3.yield 方法

暂停当前正在执行的线程对象。yield()只是使当前线程重新回到可执行状态，所以执行 3)yield()的线程有可能在进入到可执行状态后马上又被执行。yield()只能使同优先级或更高优先级的线程有执行的机会。

4).join 方法

等待该线程终止。等待调用 join 方法的线程结束，再继续执行。如：t.join();//主要用于等待

t 线程运行结束，若无此句，main 则会执行完毕，导致结果不可预测。

10、Final、finally、finanlize()的区别

答：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。

11、常用设计模式及应用场景，用两种方式实现单例模式，要求线程安全？

答: 常用设计模式:

单例模式: Calendar 实例的获取

适配器模式: Adapter 为 ListView GridView 等添加数据

工厂模式: Spring IOC 反转控制

代理模式: Spring AOP 面向切面编程

观察者模式: ContentObserver 监听内容改变

（懒汉式）程序执行过程中需要这个类的对象时再实例化该类的对象

步骤 1.定义静态私有对象

2. 构造方法私有化保证在类的外部无法实例化该类的对象

3. 定义对外开放的静态方法在调用方法是判断对象是否为空，为空再创建对象返回

```
public class Singleton {
  private static Singleton singleton;
  // 构造方法私有化，保证在类的外部无法实例化该类的对象
  private Singleton() {
  }
  public static synchronized Singleton getSingletonInstance() {
    if (singleton == null) {
      singleton = new Singleton();
    }
    return singleton;
  }

}
```

（饿汉式）类加载的时候就实例化该类的对象

```
public class Singleton {
  private static Singleton    singleton = new Singleton();

  // 构造方法私有化，保证在类的外部无法实例化该类的对象
  private Singleton() {
  }
  public static Singleton getSingletonInstance() {
    return singleton;
  }
}
```

12、常用排序算法,时间复杂度,实现思路

答：冒泡排序 O(n^2）：冒泡排序也是最简单最基本的排序方法之一。冒泡排序的思想很

简单，就是以此比较相邻的元素大小，将小的前移，大的后移，就像水中的气泡一样，最小的元　素经过几次移动，会最终浮到水面上。

```
for (int i = 0; i < num.length; i++) {
// 内循环控制比较后移位
for (int j = num.length-1; j > i ; j--) {
                    if (num[j-1]>num[j]) {
                        temp = num[j-1];
                        num[j-1] = num[j];
                        num[j] = temp;
                    }
                }
```

快速排序 O(n log n) ：快速排序采用的思想是分治思想。快速排序算法的核心算法是分区操作，即如何调整基准的位置以及调整返回基准的最终位置以便分治递归。

插入排序 O(n^2）:将新来的元素按顺序放入一个已有的有序序列当中。

选择排序 O(n^2）:第 i 趟简单选择排序是指通过 n-i 次关键字的比较，从 n-i+1个记录中选出关键字最小的记录，并和第 i 个记录进行交换。共需进行 i-1趟比较，直到所有记录排序完成为止。

13、android 系统架构？

答：1）应用程序层　java 语言　应用程序开发

2）应用程序框架层　java 语言　OS 定制　framework 层开发

3）系统运行库层　C C++ 实现　so 库

4）Linux 内核层

14、Activity 生命周期？保存 activity 的一些信息在哪个生命周期方法中？

答：共有七个周期函数：

void onCreate(Bundle savedInstanceState) 第一次创建时调用

void onStart()　被用户可见时调用

void onRestart()　当 Activity 处于 stop 状态又被重新启动时调用

void onResume() 当获得焦点即可与用户交互时调用

void onPause()　当失去焦点时调用

void onStop()　当不可见时调用

void onDestroy()　当销毁时调用

15、Activity 的 onSaveInstanceState() 和 onRestoreInstanceState()

答: Activity 的 onSaveInstanceState() 和 onRestoreInstanceState()并不是生命周期方法，它们不同于 onCreate()、onPause()等生命周期方法，它们并不一定会被触发。当应用遇到意外情况（如：内存不足、用户直接按 Home 键）由系统销毁一个 Activity 时，onSaveInstanceState() 会被调用。但是当用户主动去销毁一个 Activity 时，例如在应用中按返回键，onSaveInstanceState()就不会被调用。因为在这种情况下，用户的行为决定了不需要保存 Activity 的状态。通常 onSaveInstanceState()只适合用于保存一些临时性的状态，而 onPause()适合用于数据的持久化保存。

另外，当屏幕的方向发生了改变，Activity 会被摧毁并且被重新创建，如果你想在 Activity 被摧毁前缓存一些数据，并且在 Activity 被重新创建后恢复缓存的数据。可以重写 Activity 的 onSaveInstanceState() 和 onRestoreInstanceState()方法。

16、Android 的四大组件是什么？它们的作用是什么？

答:Android 有四大组件：Activity、Service、Broadcast Receiver、Content Provider。

Activity :应用程序中，一个 Activity 通常就是一个单独的屏幕，它上面可以显示一些控件也可以监听并处理用户的事件做出响应。Activity 之间通过 Intent 进行通信。
Service 服务:一个 Service 是一段长生命周期的，没有用户界面的程序，可以用来开发如监控类程序。
BroadcastReceive 广播接收器:你的应用可以使用它对外部事件进行过滤只对感兴趣的外部事件(如当电话呼入时，或者数据网络可用时)进行接收并做出响应。广播接收器没有用户界面。然而，它们可以启动一个 activity 或 serice 来响应它们收到的信息。
Content Provider 内容提供者 :主要用于多个应用间数据共享。这些数据可以存储在文件系统中或 SQLite 数据库。

17、广播如何调用，有什么方式，各自的区别？

答：程序中发送广播通过 sendBroadcastReceiver（）实现
接收广播通过定义一个类继承 BroadcastReceiver 并重写 onReceive（）方法实现
注册广播有两种方式：
第一种静态方式:在清单文件中通过<receive>标签声明
第二种代码动态方式:

```
IntentFilter filter = new IntentFilter("android.provider.Telephony.SMS_RECEIVED");
IncomingSMSReceiver receiver = new IncomgSMSReceiver();
registerReceiver(receiver.filter);
```

1)第一种不是常驻型广播，也就是说广播跟随 activity 的生命周期。注意: 在 activity 结束前，移除广播接收器。
2)第二种是常驻型，也就是说当应用程序关闭后，如果有信息广播来，程序也会被系统调用自动运行。

18、Android 中 asset 文件夹和 raw 文件夹区别？

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");
```

19、Android 中的五种存储方式及其应用场景

答：1)SharedPreferences
存储路径:(data/data/packagename/shares_prefs)， 轻量级存储,以键值对的形式存储在 xml 中,一般用来保存应用中的设置属性
2)文件存储 SD 卡存储多媒体文件， 文件缓存

3) Sqlite 数据库　　存储路径:(data/data/packagename/databases), 一种嵌入式数据库,支持 sql 语言,存储大量结构性数据

4)ContentProvider　进程(应用程序)间数据共享,数据源可以是 sqlite,也可以是 xml,相关类：ContentResolver(内容解析器), ContentObserver(数据观察者)

5) 网络存储　　天气数据的 xml,json 格式等等,通过 HttpUrlConnection,HttpClient,或者 SOAP 协议获取数据

20、什么是 ANR 如何避免它？（Android 线程间的通信方式？）

答：ANR：Application Not Responding(应用程序无响应).当出现下列情况时，Android 就会显示 ANR 对话框了：　　对输入事件（如按键、触摸屏事件）的响应超过5秒　意向接受器（intentReceiver）超过10秒钟仍未执行完毕 Android 应用程序完全运行在一个独立的线程中（例如 main）。这就意味着，任何在主线程中运行的，需要消耗大量时间的操作都会引发 ANR。

解决方案有两种:

1. AsyncTask 异步任务中，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，如移动护理系统中的网络连接、解析 XML 等操作。该方法必须重载。

onPostExecute(Result) 这个方法也运行于 UI 线程，在 doInBackground(Params…)方法执行后调用，该方法用于处理后台任务执行后返回的结果。

2.　子 thread + handler

21、Handler 的运行机制(运行原理)(Handler,Looper,MessageQueue,Message 之间的关系)

一个 Handler 允许你发送和处理 Message 和 Runable 对象，每个线程都有自己的 Looper，每个 Looper 中封装着 MessageQueue。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。每个 handler 也和线程关联，Handler 负责把 Message 和 Runable 对象传递给 MessageQueue（用到 post ，sendMessage 等方法），而且在这些对象离开 MessageQueue 时，Handler 负责执行他们（用到 handleMessage 方法）。

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。

22、listview 优化策略？

23、

答：1)、对 convetView 进行判空，是当 convertView 不为空的时候直接重新使用 convertView 从而减少了很多不必要的 View 的创建

2)定义一个 ViewHolder，将 convetView 的 tag 设置为 ViewHolder,不为空时重新使用即可

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、ListView 分页加载实现思路？

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，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

25、ListView 图片异步加载实现思路？

1.先从内存缓存中获取图片显示（内存缓冲）

2.获取不到的话从 SD 卡里获取（SD 卡缓冲，，从 SD 卡获取图片是放在子线程里执行的，否则快速滑屏的话会不够流畅）

3.都获取不到的话从网络下载图片并保存到 SD 卡同时加入内存并显示（视情况看是否要显示）

26、Intent 的原理，作用，可以传递哪些类型的参数？

答：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。

27、如何实现屏幕分辨率的自适应？

答: 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

28、简述 Android 中的 IPC 机制

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

29、Android 程序入口如何判断?

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

30、android 哪几种方式访问网络？

HttpURLConnection

HttpClient 方式（HttpGet 和 HttpPost 类）

31、说说 HttpClient 的通信过程

1.生成请求对象（HttpGet get，HttpPost post）

2.生成客户端对象 HttpClient client

3.执行请求接收相应 HttpResponse response = client.execute(post)

HttpEntity entity = response.getEntity()

4.得到数据流

InputStream inputStream = entity.getContent();

5.最后关闭过期连接

32、移动互联数据交互格式有哪些及其区别？（Json 与 xml 的区别？）

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

33、XML 解析有哪几种？各自优缺点，官方推荐使用哪种？

基本的解析方式有三种: DOM,SAX,Pull

1.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，然后代码就可以使用 DOM 接口来操作这个树结构的优点是对文档增删改查比较方便，缺点占用内存比较大。

2.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查。

3.pull 解析同样基于事件驱动型,android 官方 API 提供,可随时终止

34、百度地图核心类，及实现的功能？

BMapManager:地图引擎管理类,负责初始化，开启地图 API，终止百度地图 API 等工作

MKSearch：搜索服务.用于位置检索、周边检索、范围检索、公交检索、驾乘检索、步行检索

MKSearchListener 搜索结果通知接口。该接口返回 poi 搜索,公交搜索,驾乘路线,步行路线结果

MapView:显示地图的 View

MyLocationOverlay:一个负责显示用户当前位置的 Overlay。

Overlay:Overlay 是一个基类，它表示可以显示在地图上方的覆盖物。

35、GC 内存泄露在什么情况下回出现？怎么解决？

(一) 查询数据库没有关闭游标

(二) 构造 Adapter 时，没有使用缓存的 convertView

(三) Bitmap 对象不在使用时调用 recycle()释放内存

(四) 不用的对象没有及时释放对象的引用

36、android 内存的优化

答：android 内存泄露容易导致内存溢出，又称为 OOM。

Android 内存优化策略：

1）在循环内尽量不要使用局部变量

2）不用的对象即时释放，即指向 NULL

3）数据库的 cursor 即时关闭。

4）构造 adapter 时使用缓存 contentview

5）调用 registerReceiver()后在对应的生命周期方法中调用 unregisterReceiver()

6）即时关闭 InputStream/OutputStream。

7）android 系统给图片分配的内存只有 8M, 图片尽量使用软引用, 较大图片可通过 BitmapFactory 缩放后再使用,并及时 recycle

8）尽量避免 static 成员变量引用资源耗费过多的实例。

37、加载大图片的时候如何防止内存溢出

答: 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
}
```

38、Android 缓存机制

答: 客户端缓存机制是 android 应用开发中非常重要的一项工作,使用缓存机制不仅仅可以为用户节省 3G 流量,同时在用户体验方面也是非常好的选择，比如有些新闻客户端支持离线模式，也是通过缓存机制实现的.缓存机制分为两部分,一部分是文字缓存,另一部分是多媒体文件缓存.

文字缓存有两种实现：

1）可以将与服务器交互得到的 json 数据或者 xml 数据存入 sd 卡中，并在数据库添加该数据的记录.添加数据库记录时,提供两个关键字段,一个是请求的 URL,另一个则是本地保存后的文件地址，每次加载数据之前都会根据 URL 在数据库中检索

2）将 JSON 数据解析后装入 List<Map>对象中,然后遍历 List,将数据统统写入相应的数据库表结构中,以后每次向服务器发起请求之前可以先在数据库中检索,如果有直接返回.

多媒体文件缓存：主要指图片缓存

图片的缓存可以根据当前日期,时间为名字缓存到 SD 卡中的指定图片缓存目录,同时数据库中做相应记录,记录办法可以采用两个关键字段控制,一个字段是该图片的 URL 地址,另一个字段是该图片的本机地址.取图片时根据 URL 在数据中检索,如果没有则连接服务器下载,下载之后再服务器中作出相应记录

缓存文件删除策略:

1. 每一个模块在每次客户端自动或者用户手动更新的时候删除相应模块的缓存文件,并重新下载新的缓存文件.

2. 在设置界面中提供删除缓存的功能,点击后删除本机所有缓存.

39、如何实现消息推送，有哪些方式，各自优缺点，最常使用哪种？

答：实现消息推送的方式有五种，分别是轮询，SMS,C2DM,MQTT,XMPP 最常使用的是 XMPP,我们做项目时采用的是 XMPP 协议

1.XMPP 协议，它是一种基于 XML 的传递协议，具有很强的灵活性和可扩展性。它的特

点是将复杂性从客户端转移到了服务器端。GTalk、QQ、IM 等都用这个协议。

2.轮询:客户端定时去服务端取或者保持一个长 Socket，从本质讲这个不叫推送，而是去服务端拽数据。但是实现简单，主要缺点：耗电,浪费用户流量等

3.Google 的 C2DM，具体不细说，缺点，服务器在国外，不是很稳定。

4.通过短信方式, 但是很难找到免费短信平台

5. MQTT 协议, IBM 提供的一种推送服务,不太灵活

40、MVC 在 Android 中的应用

答: Android 中界面部分也采用了当前比较流行的 MVC 框架，在 Android 中：

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。也可以使用 JavaScript+HTML 等的方式作为 View 层，通过 WebView 组件加载,同时可以实现 Java 和 JavaScript 之间的通信。

2) 控制层（Controller）：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，Android 的控制层的重任通常落在了众多的 Acitvity 的肩上，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。

在 Android SDK 中的数据绑定，也都是采用了与 MVC 框架类似的方法来显示数据。在控制层上将数据按照视图模型的要求（也就是 Android SDK 中的 Adapter）封装就可以直接在视图模型上显示了，从而实现了数据绑定。比如显示 Cursor 中所有数据的 ListActivity，其视图层就是一个 ListView，将数据封装为 ListAdapter，并传递给 ListView，数据就在 ListView 中显示。

41、Android 自定义组件实现思路

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

42、版本更新的实现思路

答：在服务器相应 URL 上有版本文件，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

43、播放视频有哪些实现方式?

答：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。

44、NDK 开发流程？（JNI 运行原理）

答: NDK应用的开发流程(在应用中定义本地接口(native), 编译成.h头文件,交由C程序员实现,将.c实现通过NDK编译成.so动态链接库,导入项目中libs/armeabi,代码中调用该本地接口)
应用场景: 音频,视频解码,拍摄车牌号,识别车牌号

45、如何实现一键退出？

答：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。

46、AndroidManifest.xml 清单文件<Activity>标签中属性 android:excludeFromRecents="true" android:screenOrientation="portrait" android:configChanges="orientation|locale"的含义

答: android:excludeFromRecents 表示是否可被显示在最近打开的 activity 列表里,true 表示否,false 表示是
android:screenOrientation 表示 activity 显示的模式, 一般用来设置 activity 横屏显示(horizontal)或竖屏显示(portrait)
android:configChanges=[oneormoreof:"mcc""mnc""locale""touchscreen""keyboard""keyboardHidden""navigation""orientation""fontScale"]
是当所指定属性(Configuration Changes)发生改变时，通知程序调用 onConfigurationChanged()函数,比如 orientation 屏幕方向发生改变,locale 语言环境发生改变时

47、如何将一个 Activity 设置成窗口的样式

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

48、谈谈 UI 中， Padding 和 Margin 有什么区别,gravity 与 layout_gravity 的区别

答：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
Gravity 用来指定组件内的内容相对于组件本身的位置
Layout_gravity 用来指定组件相对于其父组件的位置

49、哪个组件可以实现手风琴效果，用来实现设置界面的类，实现抽屉效果，悬浮窗口？

答：实现手风琴效果（ExpandableListView）
设置界面的类（preferenceActivity）保存到 sharedpreference 中
抽屉效果（slidingDrawer）组件
悬浮窗口: PopWindow,可以实现类似 Dialog 和菜单的效果

50、Android SDK 3.0(HoneyComb)及 4.0(Ice Cream)新特性

答：新版 SDK 发布的同时也发布了一个扩展包 android-support-v4，把部分特性单独的抽出来，使低版本的 SDK 也可以使用这些特性，主要支持以下特性：
Fragment: 3.0 引入,碎片管理,可以局部刷新 UI,它设计的功能和 Activity 一样强大，包括生命周期、导航等，Fragment 的每次导航都可以记录下来用于返回。
ViewPager: 提供了多界面切换的新效果
GridLayout: 4.0 引入, 网格布局, android 第六大布局
Loader: 装载器从 android3.0 开始引进。它使得在 activity 或 fragment 中异步加载数据变得简单

51、android 客户端如何实现自动登录

答: 通过 SharedPreferences 存储用户名,密码,当存储不为空时实现自动登录功能