1. **Android的四大组件**

activity是用户和应用程序交互的窗口，一个相当于我们实际中的一个网页，当打开一个屏幕时，之前的那一个屏幕会被置为暂停状态，并且压入历史堆栈中，用户可以通过回退操作返回到以前打开过的屏幕。Activity的生命周期：即“产生、运行、销毁”，但是这其中会调用许多方法onCreate、onStart、onResume、onPause、

onStop、onDestroy、onDestroy 、onRestart等。

Service是一种程序，它可以运行很长的时间，相当于后台的一个服务，通过startService(Intent service)可以启动一个Service，通过Context.bindService()可以绑定一个Service

BroadCast Recevicer：接受一种或者多种Intent作触发事件，接受相关消息，做一些简单处理，转换成一条Notification，统一了Android的事件广播模型。可以使用BroadcastReceiver来让应用对一个外部的事件作出响应。Broadcast Receiver通过NotificationManager来通知用户这些事情发生了，BroadcastReceiver

注册的有两种方式，一种是可以在AndroidManifest.xml中注册，另一种可以在运行时的代码中使用Context.registerReceiver()进行注册。用户还可以通过Context.sendBroadcast()将他们自己的intent broadcasts广播给其他的应用程序。

Content provider：内容提供者，可通过它来共享自己的数据给外部调用，给第三方应用提供数据访问的接口。

1. **Android中Intent作用**

在一个Android应用中，主要是由四种组件组成的。 而这四种组件是独立的，它们之间可以互相调用，协调工作，最终组成一个真正的Android应用。

在这些组件之间的通讯中，主要是由Intent协助完成的。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。

因此，Intent在这里起着一个媒体中介的作用，专门提供组件互相调用的相关信息，实现调用者与被调用者之间的解耦。

1. **Android项目工程下面的assets目录的作用是什么**

用于存放应用程序中使用的外部资源文件，程序可以通过I/O流对目录中的文件进行读写，存放在此目录下的文件

都会被打包到发布包中

assets和res/raw工程目录下都可以放一些小于1M的文件（2.3版本以前要求，否则将不能读出数据。），这些文

件将被打包到APK中应用使用。assets目录下的文件将不做任何处理被打包，而res/raw目录下的文件，有些文

档说会编译为二进制，有些文档说不会。一个原则那就是最好不要将过大的文件打包到APK中，如果你的资源很大，例如视频等等，应该单独存储在文件系统中。

这两个目录的区别主要是assets目录下的文件不会被映射到R.java中，即系统不会自动为其生成一个资源ID.另一方面，由于系统会为res目录下的所有资源生成相应的资源ID,所以raw中的文件也不例外。综上，我们可以通过ID去访问res/raw目录中的任何文件，而assets目录中的文件就需要借助AssetManager去访问了。最后二者还有一个区别，assets目录允许下面有多级子目录，而res/raw下不允许存在目录结构。

读取文件举例 :  
1.assets目录

AssetManager a = getAssets()  ;  
//fileName为assets目录下需要访问的文件的名称  
InputStream is = a.open(fileName) ;  
//然后就可以通过输入流来读取fileName的内容了。

另外，我们可以通过a.openFd(fileName)得到的AssetFileDescriptor对象来获得fileName文件的信息，例如长度等。我们也可以通过a.list(“”)得到assets目录下的所有文件和子目录名称的数组，通过a.list(SubFolderName), 得到assets/SubFolderName下所有的文件和子目录名称的数组。

2.res/raw目录

InputStream is = getResources().openRawResource(R.id.fileNameID) ;  
//R.id.fileNameID为**需要访问的文件对应的资源ID.接着我们就可以通过输入流**来读取相应文件的内容了。

1. **Android中service启动实现方法**

1.Context.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。

1. **Android的数据存储方式**

1.SharedPreferences共享偏好   
2.Internal Storage内部存储空间   
3.External Storage外部存储空间   
4.SQLite Database数据库   
5.Internet网络

1. **注册广播有几种方式，这些方式有何优缺点？**

有两种方法，一种是代码动态注册：

//生成广播处理    
smsBroadCastReceiver = new SmsBroadCastReceiver();    
//实例化过滤器并设置要过滤的广播

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

BroadCastReceiverActivity.this.registerReceiver(smsBroadCastReceiver, intentFilter);

第二种是在AndroidManifest.xml中配置广播

<receiver android:name=".SmsBroadCastReceiver">     
            <intent-filter android:priority="20">     
                <action android:name="android.provider.Telephony.SMS\_RECEIVED"/>     
            </intent-filter>     
        </receiver>

两种注册类型的区别是：

1)第一种不是常驻型广播，也就是说广播跟随程序的生命周期。

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](http://www.2cto.com/os/)调用自动运行。

1. **请解释下载单线程模式中Message、Handler、MessageQueue、Looper之间的关系**。
2. **AndroidManifest.xml文件中主要包括哪些信息**？
3. 包名(package):指定本应用内java主程序包的包名。当没有指定apk的文件名时，编译后产生程序包将以此命名。本包名应当在Android系统运行时唯一。
4. 认证(certificate):指定本应用程序所授予的信任级别，目前有的认证级别有platform(system)、shared、media以及应用自定义的认证。不同的认证可以享受不同的权限。
5. 权限组(permission-group):权限组的定义是为了描述一组具有共同特性的权限。
6. 权限(permission):权限用来描述是否拥有做某件事的权力。Android系统中权限是分级的，前分为普通级别(Normal)，危险级别(dangerous)，签名级别(signature)和系统/签名级别(signature or system)。系统中所有预定义的权限根据作用的不同，分别属于不同的级别。对于普通和危险级别的权限，我们称之为低级权限，应用申请即授予。其他两级权限，我们称之为高级权限或系统权限，应用拥有platform级别的认证才能申请。当应用试图在没有权限的情况下做受限操作，应用将被系统杀掉以警示。系统应用可以使用任何权限。权限的声明者可无条件使用该权限。
7. 权限树(permission-tree)权限树的设置是为了统一管理一组权限，声明于该树下的权限所有者归属该应用。系统提供了API，应用可以在运行时动态添加。 PackageManager.addPermission()
8. 使用权限(uses-permission):应用需要的权限应当在此处申请，所申请的权限应当被系统或某个应用所定义，否则视为无效申请。同时，使用权限的申请需要遵循权限授予条件，非platform认证的应用无法申请高级权限。
9. SDK（uses-sdk）:标识本应用运行的SDK版本。高兼容性的应用可以忽略此项。

application:application是Android应用内最高级别(top level)的模块，每个应用内最多只能有一个application，如果应用没有指定该模块，一个默认的application将被启用。application将在应用启动时最先被加载，并存活在应用的整个运行时生命周期。因此一些初始化的工作适合在本模块完成. Application元素有许多属性，其中：“persistent”表示本应用是否为常驻内存，“enable”表示本应用当前是否应当被加载。

1. activity:activity是application模块的运行时子元素，标识了一个UI。除了application，一个应用可以声明并实现零至多个其它运行时模块，activity也同样。activity也包含了许多定义它工作状态的属性，其中：“name”是必须的，它指定了该activity所在的文件名，如果该文件所属包不同于该应用的包名（即本描述文件的最开始处），那么名字前面需要加入所在包名。activity通过增加intent-fliter来标识哪些intent可以被处理，同时intent也是调度activity的主要参数。
2. receiver:receiver也是application的运行时子元素。receiver通过增加intent-fliter来标识它需要接受哪些intent。当收到intent后，receiver将根据不同的intent进行不同的处理。当一个Intent发出后，所有注册了该intent的receiver都将会收到，系统会根据receiver在系统中的注册次序顺序发送。当一个receiver处理完该Intent后，系统才会向下一个receiver发送。当一个receiver有多个未接收的intent时，将按照intent发送的次序顺序接收。
3. service:service也是application的运行时子元素。Service属于后台模块，启动后将长时间

运行，除非停止该service或所在应用进程被杀死。

1. provider:provider也是application的运行时子元素。它继承于ContentProvider，是对该应

用管理的用户数据的结构化接入，是基于数据库操作方式的封装。如果应用允许外部应用访问／管理它的用户数据，provider是Android平台提供的最佳方式。

1. activity-alias:顾名思义，是已有activity的别名。
2. uses-library:标识应用启动所必须的共享库。
3. **根据自己的理解描述下Android数字签名**。

Android系统要求每一个Android应用程序必须要经过数字签名才能够安装到系统中，也就是说如果一个Android应用程序没有经过数字签名，是没有办法安装到系统中的！Android通过数字签名来标识应用程序的作者和在应用程序之间建立信任关系，不是用来决定最终用户可以安装哪些应用程序。这个数字签名由应用程序的作者完成，并不需要权威的数字证书签名机构认证，它只是用来让应用程序包自我认证的。

1. **如何打开res raw目录中的文件？**

在Android中不能直接打开res ｒaw目录中的数据库文件，而需要在程序第一次启动时将该文件复制到手机内存或SD卡的某个目录中，然后再打开该数据库文件。复制的基本方法是使用getResources().openRawResource方法获得res ｒaw目录中资源的 InputStream对象，然后将该InputStream对象中的数据写入其他的目录中相应文件中。在Android SDK中可以使用SQLiteDatabase.openOrCreateDatabase方法来打开任意目录中的SQLite数据库文件

1. **Android系统的架构，采用了分层的架构，从高层到底层分别是：**

1.Android系统架构之应用程序

Android会同一系列核心应用程序包一起发布，该应用程序包包括email客户端，SMS短消息程序，日历，地图，浏览器，联系人管理程序等。所有的应用程序都是使用JAVA语言编写的。

2.Android系统架构之应用程序框架

开发人员也可以完全访问核心应用程序所使用的API框架。该应用程序的架构设计简化了组件的重用;任何一个应用程序都可以发布它的功能块并且任何其它的应用程序都可以使用其所发布的功能块(不过得遵循框架的安全性限制)。同样，该应用程序重用机制也使用户可以方便的替换程序组件。

3.Android系统架构之系统运行库

1)程序库

Android 包含一些C/C++库，这些库能被Android系统中不同的组件使用。它们通过 Android 应用程序框架为开发者提供服务。

4.Android系统架构之Linux 内核

Android 的核心系统服务依赖于 Linux 2.6 内核，如安全性，内存管理，进程管理， 网络协议栈和驱动模型。 Linux 内核也同时作为硬件和软件栈之间的抽象层。

1. **Android中发送短信方式：**

1.调用系统短信接口直接发送短信

/\*\*

\* 直接调用短信接口发短信

\* @param phoneNumber

\* @param message

\*/

public void sendSMS(String phoneNumber,String message){

//获取短信管理器

android.telephony.SmsManager smsManager = android.telephony.SmsManager.getDefault();

//拆分短信内容（手机短信长度限制）

List<String> divideContents = smsManager.divideMessage(message);

for (String text : divideContents) {

smsManager.sendTextMessage(phoneNumber, null, text, sentPI, deliverPI);

}

}

调起系统发短信功能；

/\*\*

\* 调起系统发短信功能

\* @param phoneNumber

\* @param message

\*/

public void doSendSMSTo(String phoneNumber,String message){

if(PhoneNumberUtils.isGlobalPhoneNumber(phoneNumber)){

Intent intent = new Intent(Intent.ACTION\_SENDTO, Uri.parse("smsto:"+phoneNumber));

intent.putExtra("sms\_body", message);

startActivity(intent);

}

}

1. **谈谈你对android恶意软件的定义：**

Android软件的主要恶意行为有：恶意扣费或消耗套餐，窃取用户隐私资料，无提示联网[下载](http://download.tech.qq.com/)软件，大量传播恶意软件等。其中**恶意扣费**或**消耗套餐**，**窃取用户隐私资料**是最为常见的恶意行为，也是目前Android恶意软件的主要牟利方式。

1. **谈谈你对恶意软件的查杀处理机制：**
2. **Android系统ROOT权限的获取：**
3. **Android恶意软件的分析方法：**
4. **基于Android安全，你认为有哪些隐私泄露的风险，记录如何获取这些隐私信息**
5. **下面程序段的输出结果是：**

Void complicatedex(){

int x=20,y=30;

boolean b; b=x>50&&y>60||x>50&&y<-6||x<-50&&y>60||x<-50&&y<-60;

system.out.println(b);

}

结果为：fasle