**Srevice**

一、生命周期

startService

onCreate()

onStartCommand()

onDestory()

bindService

onCreate()

onBind()

onUnbind()

onDestory()

区别：第一种服务不会随着开启者的结束而结束，第二种会随着开启者的结束而结束。

二、怎么保证service不被杀死

开两个进程相互保护

startForeground将进程设置为前台进程

**Intent**

一、传递的数据类型：

A、Serializable  B、charsequence C、Parcelable  D、Bundle

还有基本数据类型。

**ContenProvider**

一、作用：应用程序间传递数据

A应用：ConentResolver

B应用：ContentProvider

A通过Uri来对B的应用CRUD操作

底层可以为sql实现、可以是文件

一对多多对一的问题

**ContentProvider和数据库的区别于联系**

程序通过contentProvider将自己的程序提供给其他应用程序，内容提供者可认为是程序间共享数据的接口

屏蔽了数据存储细节，只需要知道uri即可

contprovider能实现不同app的数据共享，sql只有自己程序才能访问。

**Thread与AsycTask的区别：**

thread是开启一个线程，AstcTask是对线程池的封装，使用handler完成主线程与子线程的共享。

AsyncTask在操作UI线程上更方便，不需要通过handler来发送消息。

**Android数据存储格式**

文件存储

网络存储

数据库存储

sharePerences存储：密码账号存储，需要加密

**Android中的mvc模式**

m:modle 数据库的操作，网络的操作，业务

v:view,xml，java

c:control activity

**Android怎么加速启动activty**

b的oncreate减少耗时操作

a的onpause减少耗时操作

**Android进程**

前台进程：正在交互运行的进程

可见进程：可见但不能交互，弹出对话框

服务进程：下载服务

后台进程：home键按下后应用进入后台进程

空进程：缓存，缩短应用启动时间

**AIDL：Android 接口定义语言**

作用：进程间通信

支持的传输的数据

基本数据类型

list、map、charsequence，list与map中必须是基本数据类型

自己的对象继承serializable或parcelable

AIDL与Messenger的区别

Messenger不适合大量并发的请求，串行处理客户端发来的消息

主要为了传递消息

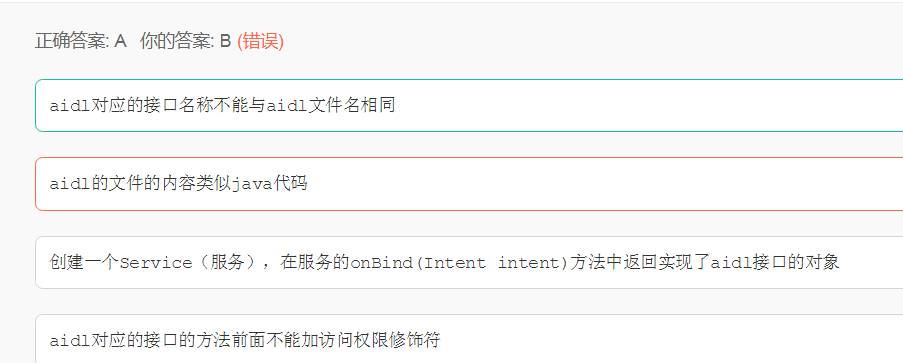
底层实现四AIDL

AIDL适用大量并发的请求，以及涉及到服务端方法调用的情况。

使用流程：

服务端创建Service监听客户端的请求，创建一个AIDL接口，定义接口给客户端使用。在service中创建一个类继承AIDL的stub类，然后创建一个对象，并实现这些接口。然后将该对象作为返回值返回(返回的是proxy对象)。

在客户端绑定服务，将服务端返回的Binder对象转成AIDL接口所属的类型，然后就可以调用AIDL的方法。这个Binder就存在于运行在客户端的额proxy代理类里面。



**BroadcastReceiver**

一、作用：实现系统不同组件间的通信，系统级别的监听器

二、注册

动态注册：代码注册

静态注册：manifest中注册

区别：

静态注册：应用程序关闭了，如果有广播信息来，你写的广播接收器同样的能接受到。应用程序不启动都可以接受。

动态注册：当应用程序结束了，广播自然就没有了，比如你在activity中的onCreate或者onResume中注册广播接收器，在onDestory中卸载广播接收器，应用结束就结束。

三、广播的种类

Normal BroadCast(普通广播)：同一时刻可以被所有的接受者接收到，传递效率高，无法终止，接受者不能将处理结果传递给下一个接受者。

Ordered BroadCast(有序广播)：接受者可以终止其传播。

**程序设计三种结构**

顺序结构

选择结构

循环结构

**ANR：程序运行无响应，但还是会恢复**

main线程执行了耗时操作

应用运行时，操作过于频繁

**Force Close：应用强制关闭**

应用出现错误，比如空指针，找不到类、被零除、数组越界访问等。

运行时异常

ArithmeticException

ArrayStoreException

BufferOverflowException

BufferUnderException

IndexOutOfBoundsException

NullPointException

可控异常（编译时出现，不会到运行那一步）

ClassNotFoundException

自定义异常可以继承Exception与RuntimeException

**android:layout\_gravity 和 android:gravity 的区别**

android:gravity是对元素本身说的，元素本身的文本显示在什么地方靠着换个属性设置，不过不设置默认是在左侧的。

android:layout\_gravity是相对与它的父元素说的，说明元素显示在父元素的什么位置。

比如说button： android:layout\_gravity 表示按钮在界面上的位置。 android:gravity表示button上的字在button上的位置。

**Android的权限机制**

**运行时权限与文件系统权限的区别**

运行时权限：Dalvik授权

文件系统权限：linux内核授权

**LruCache DiskLruCache**

**一、缓存介绍**

LRU(Least Recently Used), 即近期最少使用算法.

使用缓存策略， 对网络上下载的图片等资源文件进行缓存， 当再次请求同一个资源url时， 首先从缓存中查找是否存在， 当不存在时再从网络上下载。采用缓存， 除了提高获取资源的速度， 也对减少使用用户手机上的流量有很好的作用. 核心思想是当缓存满时，会优先淘汰那些最少使用的缓存对象。

**二、实现方式：LurCache DiskLruChache**

LurCache：内存缓存、LinkedHashMap以强引用的方式存储外界的缓存对象，提供了get与put完成缓存的获取与存储。

DiskLruChache：存储设备缓存、LinkedHashMap

LruCache只是管理了内存中图片的存储与释放，如果图片从内存中被移除的话，那么又需要从网络上重新加载一次图片，这显然非常耗时。

DiskLruCahce一般将url用md5转化为key。

**自定义VIew的过程**

自定义属性比如字体，颜色，样式

view的构造过程获得自定义的样式

onMesure()

onDraw()

使用sp做字体的大小单位

**Android内存泄露**

检测工具：LeakCanary

**Thread、Handler和HandlerThread关系**

Thread：开启线程

Handler：线程间通信

HandlerThread：子线程与子线程之间通信，封装好了lopper，如果Handler来做子线程与子线程的通信，需要自己操作自己操作Looper。我们要在子线程中调用Looper.prepare() 为一个线程开启一个消息循环，默认情况下Android中新诞生的线程是没有开启消息循环的。（主线程除外，主线程系统会自动为其创建Looper对象，开启消息循环）

**AsyncTask、HandlerThread与IntentService的差别**

AsyncTask：封装了线程池与handler，方便开发者在子线程中更新UI。在线程池中执行一步任务。

HandlerThread：具有消息循环的线程，就是说自己封装了lopper

IntentService：一个后台服务，不容易被杀死，任务执行完自动退出。封装了HandlerThread与Handler，Handler发送消息，HandlerThread会进行处理。

**ListView中item与子控件同时可点击**

1、在Item的xml文件根元素中添加属性：android:descendantFocusability="blocksDescendants"

viewgroup会覆盖子类控件而直接获得焦点，点击item时让item获得焦点

2、在冲突的子控件中添加属性：android:focusable="false"和android:clickable="true"

**进程间通信：**

ContentProvider：使用简单，适合一对多的进程间的数据共享

AIDL：功能强大，一对多，支持实时通信。

Messenger：传递Message对象，一次处理一个数据，不适合高并发。

文件共享：A写B读，简单易用，不适合高并发

socket：网络通信，支持一对多并发实时通信。

Bundle加Intent：使用简单，四大组件间的通信，数据传输类型受限

**Android横竖屏切换时，Activity的生命周期**

1. 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. 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. 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

**Java防反编译的方法**

1. 代码混淆

不能混淆：实现了 Serializable 接口的类、自定义的View、本地方法、用到反射的地方

2. 加密class文件

**移动应用如何防止内存过大**

防止内存泄露

及时关闭资源链接

善用leakCanery