# 版本控制

## SVN

## Git

## SVN原子提交的理解

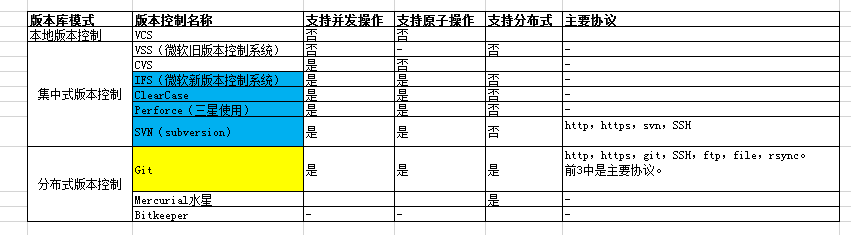
SVN一次性提交很多文件时，要么全部提交成功，要不全部提交失败。

## Git分布式版本控制的概念理解

用户从远程仓库分支clone文件到本地仓库，用户大部分操作都是本地仓库完成的。用户之间不受影响。这解决了本地仓库对远程仓库的网络依赖性，降低了各个用户本地仓库之间与各个用户本地仓库与远程仓库版本不同。

本地仓库完成操作后，通过pull或format-patch方式将更改同步到远程服务器。

## 说出常见的版本控制系统及其区别



svn：文件比较

Git：快照

## 说说SVN与Git孰优孰劣，为何？

Git和SVN？

Git支持分布式。

Git用C语言开发，原理是基于快照的，并且提交到本地。因此，速度很快。

Git能建立很多成千上万的分支，数据存储量大，管理适合管理大型项目。

svn有更严格和更好的权限控制。

svn更易操作。

基于以上：

svn经常使用于企业的商业软件，因为自由度不大，以及操作简单。

Git经常用于开源项目，甚至大型开源项目，和大型项目，例如Android源码、linux内核。

## SVN和Git的区别？

1、SVN与Git在概念和特性上的区别

|  |  |
| --- | --- |
| SVN | Git |
| C/S结构 | 分布式 |
| 客户端只有工作区域 | 客户端有工作区域、临时缓存区、本地仓库 |
| SVN不方便离线操作 | Git可以一直离线操作（删除后，本地可以找回） |
| SVN速度慢 | Git速度快（快照） |

2、SVN与Git在操作上的区别

|  |  |
| --- | --- |
| **SVN** | **Git** |
| checkout | chone,pull, fetch |
| svn branch 建立新目录 | Git branch不是 |
| Update | checkout |
| Commit | push |
| Delete | rm |

## 说说Mercurial 和 Git孰优孰劣 ，为何？

水星操作更简单，但速度比Git慢。

因为Git是用C语言，快照原理。水星是Pyton语言。

## Git如何建立版本仓库

git int

git chone

git pull

git fetch

## 根据Android源码下载过程介绍一下Git

略。

## Repo与Git的关系

Repo操作很多和Git很类似。

Repo是用来管理Android源码的Python脚本。

google为了简化下载，使用python语言写的脚本，里面其实还是是Git管理。

下载Repo需要Curl的支持。

## Git如何进行权限管理？

和SAH1有关。

Git的权限管理不是很好，可但通过其他工具也可以达到有限的权限控制，比如：gitosis、CodeBeamer MR。

## Git常见操作

略。

## SVN常见操作

略。

# 其他

# 重构

重构是一个随时随地进行的工作。

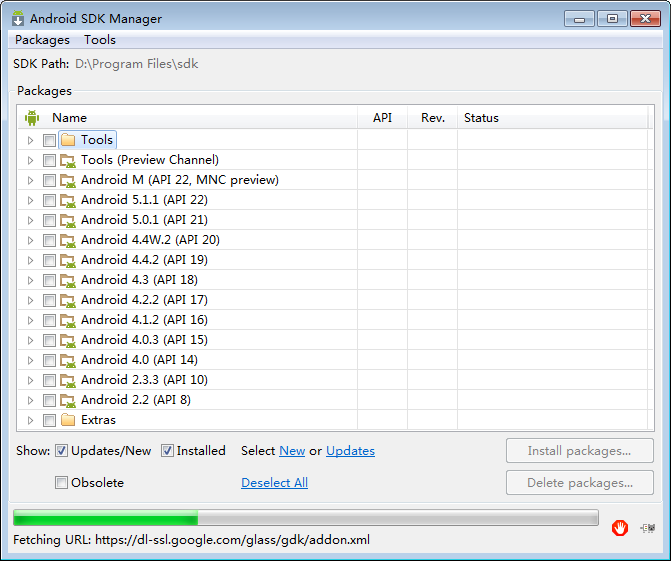
重构的准则：

Eclipse工具也有重构应用。菜单 -> Refactor。

# 设计模式

# Android主题

## Android版本号与API版本号的对应关系



开发时向上API 19, Android4.4.2，源码也是。向下兼容Android 2.3.3,API10。

## Android常见工具

### adb

（1）、什么是adb？

adb 是Android Debug Bridge的缩写，它起到调试桥的作用。

（2）、adb 作用？

作用1：使用Eclipse时，通过adb使得能够用DDMS调试程序，它就是一个debug工具。

adb的工作方式比较特殊采用监听Socket TCP 5554等端口的方式让IDE和Qemu通讯。运行Eclipse时adb进程就会自动运行。

Qemu是一种模拟器。

作用2：adb是连接Android手机与PC端的桥梁，让用户在电脑上对手机进行全面的操作。

借助adb工具管理设备或手机模拟器的状态。例如，安装软件、系统升级、运行shell命令等。

（3）、adb常见命令？

adb devices 查看设备。

adb install 安装软件

adb uninstall 卸载软件

adb shell 进入shell

adb push 上传文件到设备

adb pull 从设备下载文件

adb bugreport 查看bug报告

sqlite3 数据库操作

获取设备的ID和序列号

adb get-product

adb get-serialno

什么是Android ?

Android特性

Android的系统架构的理解

Android相关的四大网站

重点难点：

考核目标：

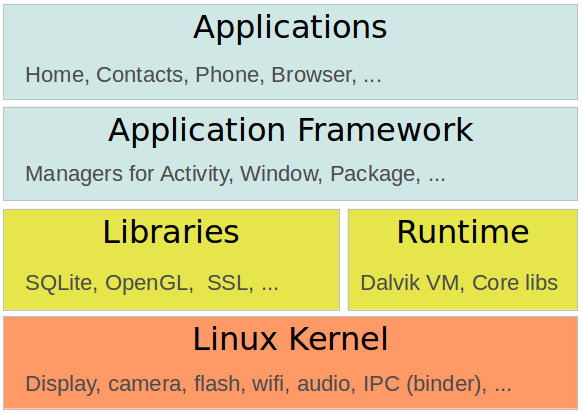
课后练习：

## 什么是Android？

Android是google公司开发的一套基于linux内核的操作系统。

## 整体架构

### Android系统架构图理解、Android的系统结构有哪几层？



如上图，android基于linux操作系统，使用linux内核与设备的硬件进行交互。在内核之上，又抽象出了一层，包括Dalvik虚拟机等。

因为dex是Dalvik VM Executes的全称，即android Dalvik虚拟机执行程序。

### LOG

Log的设备驱动是Logger（软驱动）

log.i -> Lib -> Driver

### 软驱动与硬驱动

什么是软驱动？

Log.i-> Lib -> Driver

驱动不需要硬件支持。

什么是硬件驱动

驱动有硬件支持。

## Android开发人员具备哪些素质

（1）、勇于并且能够承担自己的责任。

（2）、有开放的心态，能够启发周围的同事。

（3）、设定目标。

（4）、随时准备学习，并且时刻学习。坚持每天进步一点点。

（5）、积极接受公司的培养和自我培养。

（6）、注意细节，做事尽量多走一步。

（7）、踏实、好学、勤学并且勤于总结。

红牛→你的能力和创造力超乎想象。

## 看源码的工具有哪些？

看源码的工具

文本查看器Source Insight。

文本编译器notepad

文本编译器EditPlus

文本编译器UltraEdit

比较源代码工具Beyond Compare 3。

我经常的使用Source Insight来查看源码，使用UltraEdit和有道云记录笔记，使用Beyond Compare 3、Eclipse Git或EclipseSVN查看来比较源代码。

## 常见的命名法

1、Pascal命名法：每个单词首字母大写。

String MyName - Delphi

int MyAge ;

2、Camel命名法：首字母小写。

String myName – java

3、匈牙利命名法：前缀标定类型和特征

Android采用Android源码的命名方法一致。Android命名规范：都采用，但不完全采用。

Pascal命名法：每个单词首字母大写。文件名、类名

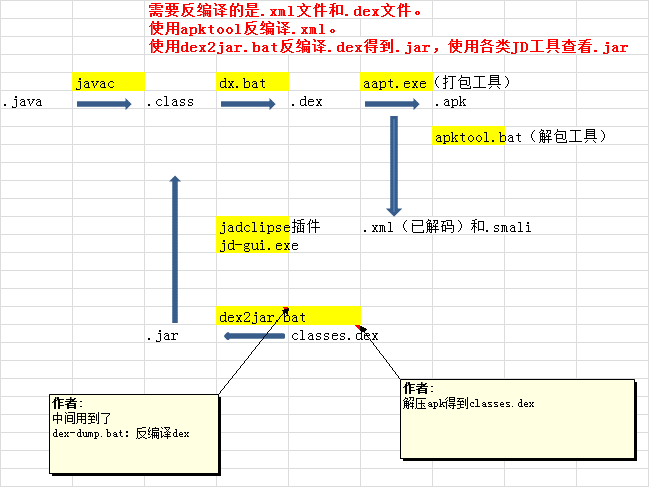
Camel命名法：首字母小写。变量，方法。

匈牙利命名法：私有成员变量加m，静态属性加s

下滑线命名法：常量。

## Android反编译与代码混淆

### 反编译



apk是一个特殊的压缩包。反编译.apk就是反编译dalvik二进制文件.dex和压缩的.xml文件。

反编译的方式:apktool + dex2jar + Jd-gui/jadclipse。

### 混淆

使用ProGuard配置防止反编译：

sdk/docs\_Android 4.4.2（API 19）/tools/help/proguard.html

## Lesson3：Android应用程序基础以及原理概念

### 掌上设备的局限性会制约需求

（1）、更低的处理器

（2）、有限的内存

（3）、有限的存储能力：一次性读取2000条并填充到ListView中，发现屏幕延迟非常严重。Sqlite 独占，不能并发。

（4）、小屏幕和低分辨率：

（5）、数据传输的高成本:压缩传输、小传输。

（6）、数据传输的低速度和高延迟：

（7）、不稳定的数据连接：无线网络的不稳定性，考虑网络连接等的异常处理。

（8）、有限的电量：功耗比较低。

### 在做开发时应该考虑的非功能性需求？

（1）、性能（更快和更灵敏）

（2）、响应能力（拒绝ANR）

（3）、与其他程序的跳转要流畅，前台后天的转换要流畅。

（4）、安全：对操作权限的控制。必要时加密。

### Android**应用程序**的基本特征：

（1）、.apk是android应用程序安装格式,.dex是Android二进制执行文件格式

（2）、Android操作系统是基于Linux的多用户操作系统，每一个应用程序都是使用的不同的用户。

每一个应用程序pid，系统临时建立的用户去启动的，而不是root去启动的。并且这个用户userid对应于pid。

保证用户对各个自己的应用程序的权限足够大。

保证各个应用进程之间的安全性。

（3）、默认情况下，一个应用程序一个userid，系统设定的，你不需要手动去设定，针对这样的应用程序，该用户有权限进行所有操作

（4）、每一个进程有自己的虚拟机实例，所以每个应用程序相对独立。

（5）、默认情况下，应用程序都有自己的linux进程，系统会决定何时关闭它，当资源不够的时候。

（6）、例外的情况：可以给两个应用程序共享同一个userid，让彼此之间共享资源。

应用程序可以请求访问系统资源，比如通讯录，短信等，但需要安装时候被授权。

### raw(res\raw)与assets之间的区别？

raw与assets通常会放置一些小于1M的文件。一个原则就是最好不要将过大的文件打包到APK中，如果资源很大，例如视频等等，应该单独存储在文件系统中。

raw中文件会被打包，然后解压到程序路径。

raw资源映射到R.java中，即生成一个资源ID。

raw不允许存在目录结构。

raw通过R直接访问。

assets是应用下的一个普通的文件夹。assets目录下的文件将不做任何处理被打包。

assets不会映射到R.java文件。

assets允许出现多级目录。

assets需要借助AssetManager通过相对路径来访问（getResource().getAsserts），通常用于大文件。asserts下可以建多个文件夹。

### 如何选择raw和assets？

raw需要解压到程序路径。文件小，而且不多时，可以使用raw。

mp3之类，放在asserts，使用asserts更灵活，且能减少R文件消耗。

### Android应用程序结构、应用程序资源的详细了解

src

gen

assets

bin

lib

res>raw

res>drawable-hdpi

res>layout 通用.xml文件或图片

res>aim 动画

res>menu

Manifest.xml

### 对几大组件的理解，特别是Intent

Android四大主键

Activity、Service、Broadcast、ContentProvider。

一个程序应该只有一个action为"main"动作和category 为"launcher" 的Activity。

一个程序通常由多个activity组成，但至少1个Activity。

Android五大主键

Activity、Service、Broadcast、ContentProvider、多线程

Android六大主键

Activity、Service、Broadcast、ContentProvider、多线程、Notification

Android七大主键

Activity、Service、Broadcast、ContentProvider、多线程、Notification、Intent

### AndroidManifest.xml文件，解释element内容

（1）、manifest是什么？

1，应用程序的全局配置文件

2，应用程序的入口配置



（2）、最常见信息？

版本号、版本名称

<uses-sdk>最低兼容版本、目标版本

<uses-permission />设置用户授权

<permission /> ：自定义权限。

<application> 标签设置Activity、service、receiver、

sdk/docs\_Android 4.4.2（API 19）/reference/android/Manifest.html

android包下有Manifest和R文件说明

### manifest是如何解析的？

1,应用程序安装的时候，manifest信息会放到一张系统索引表里面

2,系统启动的时候，PackagedManageService读取Manifest信息

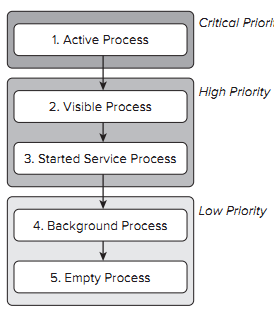
3，应用程序启动时候，Launcher进行系统级判断，比如最小SDK等

4，四大组件在需要的时候实例化。

### Android进程生命周期介绍

### Android进程如何启动和消亡的

### Android进程等级，以及优先级别，以及5个等级的转换关系？



（1）、 Active Process

正处于Activity Resume() 状态

正处于与bound服务交互的状态

正处于前台运行的状态 ， （startForeground() 被调用）

Service生命周期函数正在被执行 (BroadcastReceiver 正在执行onReceive()方法onCreate() , onStart() , onDestroy())

杀死Foreground Process 需要用户响应-因为这个安全优先级是最高的

2) Visible Process

当前进程不是焦点进程，Activity 不在前端显示 ， 但也没有完全隐藏，能够看得见，比如弹出一个对话框 。(Input Method)

一个bound到visible 或者 foreground 的activity的 Service

3) Service Process

正在运行的，不在上述两种状态的Service

4) Background Process

不可见状态的Activity进程，又没有任何相关Service运行的(onStop()被调用），系统有个后台进程列表，一般情况采取后进先出的原则来释放后台进程资源

5) Empty Process

没有运行任何Components的进程，保留这个进程主要是为了缓存的需要。

Active Process → Visible Process → Service Process → Background Process → Empty Process

Background Process：按Home键后。

Empty Process：用户一直按Back键后，处于Empty Process。此时，进程依然活着。

### 如何杀死进程？

finish(); finish不能成功杀死进程。finish()调用后，系统会调用destroy方法杀死这个Activity，通常与回传值相关。

System.exit(0); 成功杀死进程。系统进程的正常退出

Process.killProcess(Process.myPid())。成功杀死进程。比较粗暴，类似于垃圾回收。

由于EmptyService占用很少资源，主要是为了缓存使用。不建议强制杀死。但是如果有后台耗时服务在运行，可以强制杀死。

## 四大组件之Activity

### **什么**activity?

【官方】activity是一个应用程序组件，它提供完成某些事情的用户交互。

【理解】activity是业务类，是承载应用程序界面和业务行为的基础。

### 什么是tasks and back stack（回退栈 ）？或者Activity栈管理方式？或者Activity栈模型？

从A窗口启动B窗口时，A会被压到栈中。此时，B被称为活动窗体，B会被压栈，放到栈顶，并且获得用户焦点。当用户在这个B使用返回键，B会从栈中弹出并被销毁，同时前面的A恢复了。

用户能看到的永远是最上面的窗体。

回退栈为系统分配的，不需要用户维护。

回退栈遵循后进先出原则。

### 什么是任务（Tasks）？

多个Activity按顺序组成的业务逻辑。这些Activity可能是属于同一个进程或不同进程。

可以使用getTaskId()来判断是否属于同一个task。

### 用户操作时Activity的顺序关系？

2015/9/4, vicky

任务栈 Tasks Stack。

### Activity和界面的关系？

Activity不是一个界面，界面也不是一个Activity。Activity包含界面。

### Activity的生命周期是什么？

2015/9/4, vicky

主要是看A→B，B→A，A按下Home键或电源键，然后A按Back键。

这里，窗体，B为全窗体。

如果B为半窗体，情况要复杂的多。这里不举这种例子。

原则：把当前窗口先pause，然后创建新的窗体，直到这个窗体能被用户看到，才stop掉前面的窗体。目的是避免出现黑屏、闪屏，提高用户友好性。

按Home和按电源键的效果是一样的。

执行stop后，窗体就看不见了。



测试跳转到另一界面，生命周期到底发生了什么？

FirstActivity 是主界面，JumpActivity是第二个界面（全屏幕）。验证onPause -> onStop -> onRestart -> onStart -> onResume

启动主界面

/FirstActivity(29172): onCreate func called!

/FirstActivity(29172): onStart func called!

/FirstActivity(29172): onResume func called!

点击跳转按钮，跳转到JumpActivity

/FirstActivity(29172): onPause func called!

/JumpActivity(29172): onCreate func called!

/JumpActivity(29172): onStart func called!

/JumpActivity(29172): onResume func called!

/FirstActivity(29172): onStop func called!

按返回键，从JumpActivity跳转到主界面

/JumpActivity(29172): onPause func called!

/FirstActivity(29172): onRestart func called!

/FirstActivity(29172): onStart func called!

/FirstActivity(29172): onResume func called!

/JumpActivity(29172): onStop func called!

/JumpActivity(29172): onDestroy func called!

按返回键，退出主界面FirstActivity

/FirstActivity(29172): onPause func called!

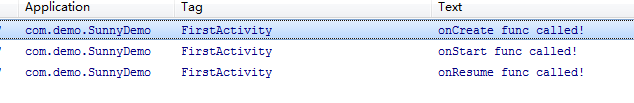
/FirstActivity(29172): onStop func called!

/FirstActivity(29172): onDestroy func called!

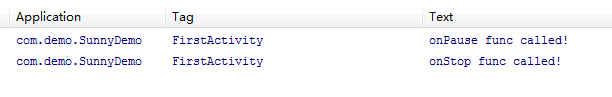
FirstActivity是主界面（全屏幕）。

按下电源键和按下Home的行为是一样的。

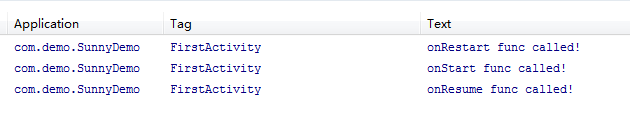
启动时，验证onCreate -> onStart -> onResume



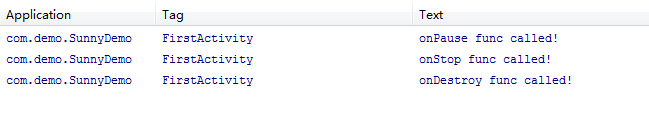
按下了解锁键，锁住屏幕 验证onPause -> onStop -> onRestart -> onStart -> onResume



按下了解锁键，解锁



按下了back键 验证onPause -> onStop -> onRestroy



### 如何初始化Activity？

第一个被初始化和显示的是mian Activity。其他Activity是用的时候才初始化。

### 如何使用一个Activity？

第1步：先是建立Activity类。

第2步：定义XML。

第3步：注册Activity到manifest。

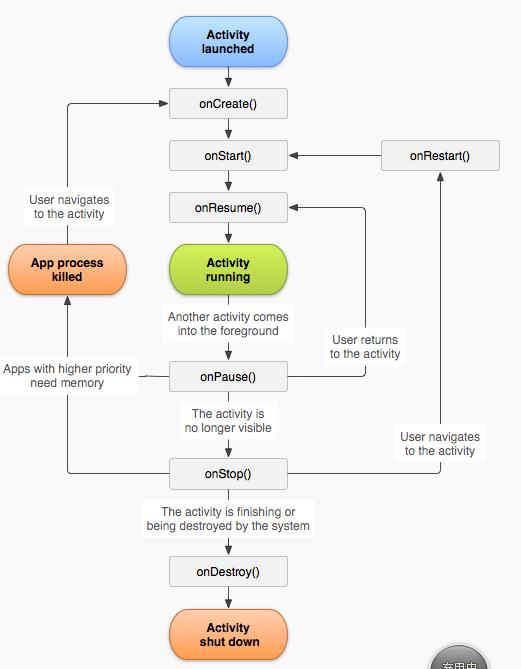
第4步：在启动函数（OnCreate）中，实现业务。

### Managing the lifecycle管理生命周期

Activity主要的3种状态：

Resumeed、Pauseed、Stopped。

直接杀死程序？后台运行？跳转到其他界面时？生命周期如何。



生命周期函数均为call back ，回调函数。

### Activity的各种生命周期

全部的生命周期：onCreate ~ onDestroy

可见的生命周期：onStart ~onStop

前台的生命周期：onResume~ onPause。因为这个状态经(onPause和onResume)常转换，所以在这两个方法中的代码应该是轻量级的，以避免因为转换慢而导致用户需要等待。

### 各个生命周期函数通常会干什么？有哪些特点？

onCreat：使用setContentView初始化窗体。设置全局状态，创建视图，绑定数据到列表，加载资源，注册BroadcastReceiver。

onStart：调用之后，Activity由不可见→可见。

onResume：获取焦点，用户交互，可见。

onPause()：选择性地保存轻量级数据，实现数据持久化。停止动画和其他高消耗CPU东西，失去焦点，调用后不可见，不能交互。执行后，Activity被暂停。被暂停的A程序是完全活着的。A被保存在内存中，它保持着所有状态和内存信息。在窗体管理中被关联着，但是当系统及其缺少内存可能被杀死。

onStop：注销BroadcastReceiver。执行后窗口不可见了。停止的活动仍是有生命的，（这个对象仍驻留在内存中，它维持着所有的状态和内存信息，但不关联到窗口管理器）

onDestroy：释放所有资源。

### 什么是手机现场保护（暂时性）？

Handling configuration changes，手机现场保护

当配置变化时，保存Activity状态，就是手机现场保护（官方）。

保存Activity状态的用处：

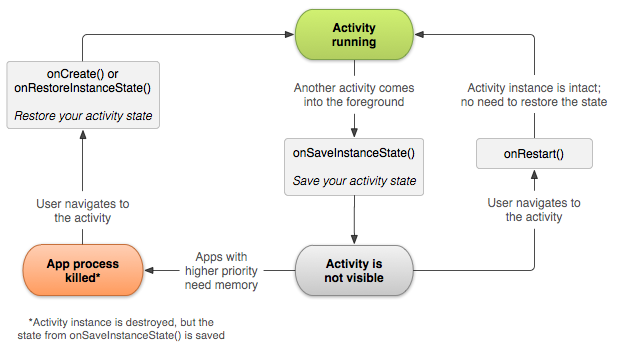
保情景（如游戏场景）到硬盘；

发生异常后能恢复到异常前的状态；

系统资源紧缺时杀死了这个Activity，系统会自动恢复。

### 保护Activity状态的方法是什么？或者，Activity如何保存和恢复状态？或者，如何进行手机线程保护

2015/9/4, vicky



2015/9/4, vicky

onSaveInstanceState()方法的默认实现保存了UI的有用的信息，但有时需要重写。

用onSaveInstanceState() 方法保存状态，用onRestoreInstanceState()方法或onCreate()方法恢复状态。

现场保护时，推荐使用onSaveInstanceState与onCreate配对使用，而不是onSaveInstanceState与onRestoreInstanceState。

### 何时需要手机保护？

暂时性配置变化：语言切换，输入有效性切换(语言和输入法)等，横屏与竖屏转换。

### Activity**经常**的配置选项

locale 语言

orientation 屏幕方向

keyboard 键盘



设置屏幕方向为横屏

android:screenOrientation="landscape"

设置屏幕方向为竖屏

android:screenOrientation="nosensor"

android:configChanges="orientation"

### 横竖屏切换时候activity的生命周期?

android:configChanges="orientation"

不设置configChanges 为orientation时，

竖屏→横屏时，执行1次生命周期，不执行onConfigurationChanged。

横屏→竖屏时，执行1次生命周期，不执行onConfigurationChanged。

配置configChanges 为orientation时，

竖屏→横屏时，执行1次生命周期，执行onConfigurationChanged。

横屏→竖屏时，不执行生命周期，执行onConfigurationChanged。

android:screenOrientation="landscape"

设置screenOrientation为landscape时，

android:screenOrientation="landscape" 设置屏幕方向为横屏，无论何时都是横屏。

android:screenOrientation="portrait" 设置屏幕方向是竖屏，无论何时都是竖屏。

android:screenOrientation="nosensor" 设置屏幕不能感应变化。CY认为手机默认是竖屏，平板默认是横屏。

设置screenOrientation后，无论configChanges有没有设置orientation，屏幕方向均不会改变，当然不会执行生命周期，不执行onConfigurationChanged。

### 手机现场保护与持久化区别？

手机线程保护：暂时性保存配置变化。

持久化：在onPause中保存到数据库或XML中，并在onCreate中加载数据。

### onSaveInstanceState与onRestoreInstanceState并不总是成对出现？

onSaveInstanceState与onRestoreInstanceState并不是成对出现。仅仅系统自己杀死程序并自动重启程序时才会调用onRestoreInstanceState。显示按下Home键时是不会调用onRestoreInstanceState。

### onSaveInstanceState何时执行？

onSaveInstanceState()方法总是在onPause之后、onStop方法前被执行

### onSaveInstanceState()一定会执行吗？

onSaveInstanceState()不保证被执行。

例如，onPause之后，当系统资源极其缺乏时，窗体执行onPause后被杀死，此时onSaveInstanceState()还没有执行。

### 系统资源缺失时，在哪个生命周期可能杀死程序？，或者在哪个生命周期函数中持久化数据？

当系统资源在极端缺乏时，在执行onPause后，Activity可能被杀死。窗口被杀死之前，只有onPause保证被执行。因此，在onPause中保存一些关键性的轻量级数据。

### Activity的启动模式（或者加载模式）有哪些？

2015/9/4, vicky

android:launchMode

A -> B -> B， A -> B -> A -> B， A -> B -> A -> B -> A

B为standard， A1 => B1 => B2， A1 => B1 => A2 => B2

B为singleTop， A1 => B1 => B1， A1 => B1 => A2 => B2

B为singleTask， A1 => B1 => B1， A1 => B1 => A2 -> B1(删除A2)

B为singleInstance，A1 => B1 => B1， A1 => B1 => A2 => B1， A1 => B1 => A2 => B1 => A3

说明：

A1 => B1 => A2 => B1（B为独立栈，按返回键时，先删除B所在栈）

A1 => B1 => A2 => B1 => A3（B为独立栈，删除时先删除A所在栈）

（1）、.standard

这是活动的默认启动模式。这种启动模式，每次启动都会在栈中新建一个该活动的实例

1. singleTop

如果窗口已经在栈顶了，就不能启动新的实例了，但会通过onNewIntent方法传递Intent信息。

但是这种方法仅限于窗口在栈顶的情况，不在栈顶，也就是不在用户能看到的界面，那么还是会创建新的实例。

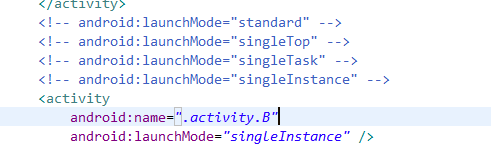
（3）、singleTask

singleTask完美解决创建重复活动实例的情况。每次启动活动，它会自动查找栈中是否存在该活动的实例，若存在直接使用，不存在才创建

（4）、singleInstance

它启动活动时，会新建一个栈来存放新启动的活动。这种模式可以解决不同应用程序之间调用活动的问题。

protected void onNewIntent(Intent intent)



### Activity的启动模式（或者加载模式）的使用场景？

（1）、singleTop适合接收通知启动的内容显示页面。例如，接到10个新订单的推送通知，使用singleTop每次打开都是同一个页面。

（2）、singleTask适合从外界可能多次跳转到一个界面。

singleTask适合作为程序入口点。例如浏览器的主界面。不管从多少个应用启动浏览器，只会启动主界面一次，其余情况都会走onNewIntent，并且会清空主界面上面的其他页面。

（3）、singleInstance适合需要与程序分离开的页面。之前打开过的页面，打开之前的页面就ok，不再新建。

singleInstance适合需要与程序分离开的页面。例如闹铃提醒，将闹铃提醒与闹铃设置分离。

singleInstance不要用于中间页面，如果用于中间页面，跳转会有问题，比如：A -> B (singleInstance) -> C，完全退出后，在此启动，首先打开的是B。某个应用中用到了google地图，当退出该应用的时候，进入google地图，还是刚才的界面

A -> B -> B， A -> B -> A -> B， A -> B -> A -> B -> A

因为B为singleInstance，A1 => B1 => B1， A1 => B1 => A2 => B1， A1 => B1 => A2 => B1 => A3

A1 => B1 => A2 => B1（B为独立栈，按返回键时，先删除B所在栈）

A1 => B1 => A2 => B1 => A3（B为独立栈，删除时先删除A所在栈）

### 使用Intent跳转界面，或者跳转到其他程序的一个界面。

方式1：Starting an activity使用Intent跳转界面

方式2：Starting an activity使用Intent跳转界面并携带数据

方式3：Starting an activity for a result使用Intent跳转，并回传值

### Starting an activity使用Intent跳转并携带数据

startActivity与Intent

Starting an Activity：startActivity()，用来启动其他窗体，窗体与窗体之间的传值。

Intent，想做什么，即想启动或想描述的activity

 android.content.Intent类

启动过程当中携带小量数据，即窗体与窗体之间的传值

intent传递大数据：序列化对象或者bundle集合。

### Starting an activity for a result使用Intent跳转，并回传值

如果A → B，A向B传值。然后B → A，B又反过来传值给A。

A中startActivityForResult代替SartActivity()，A向B传值。

B中setResult，B向A传值。

A中重写onActivityResult。

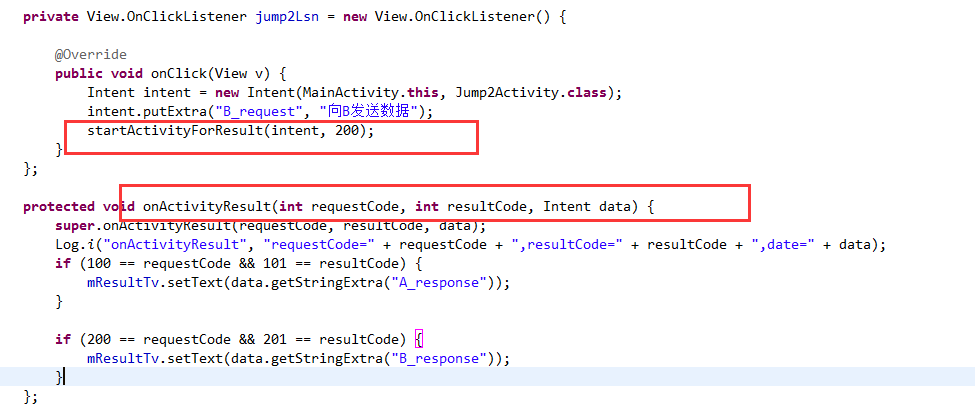
使用请求码requestCode来区分发送到不同界面的请求。

使用返回码resultCode来区分来自不同界面的返回。

A：画面：

public void startActivityForResult(Intent intent, int requestCode)

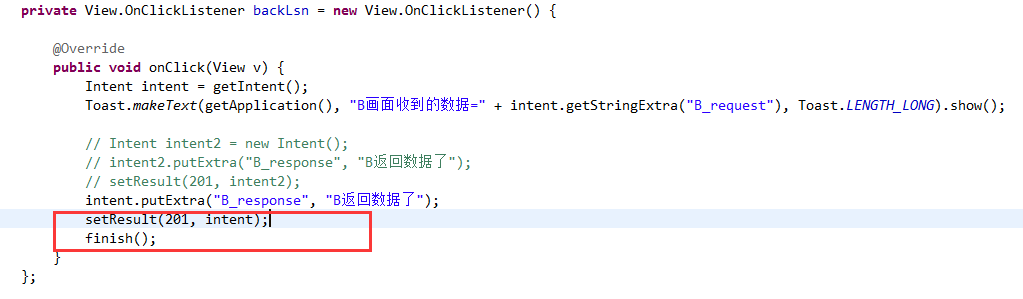
protected void onActivityResult(int requestCode, int resultCode, Intent data)



B画面：

setResult(201, intent);

finish();



### Shut Down An Activity 停止一个Activity

使用finish()停止当前界面。

使用finishActivity关闭以前启动的界面。

在程序中调用 Activity.finish()方法时，结果和用户按下 BACK 键一样：他告诉 Activity Manager该Activity实例可以被“回收”,随后 Activity Manager 激活处于栈第二层的 Activity 并重 新入栈，把原 Activity 压入到栈的第二层，从 Running 状态转到 Paused 状态。

使用finishActivity()停止用SartActivityForResult启动的当前界面。

### Intent的概念，或者Intent是什么？

2015/9/4, vicky

Intent 是对一个将要执行动作的抽象描述。一般来说是作为参数来使用，由Intent来协助完成Android各个组件之间的通讯。

例如，

调用startActivity()来启动一个activity；

使用sendBroadcast(intent)发送广播，使用registerReceiver(receiver, filter) 注册广播监听；

使用startService()/bindservice()来启动一个后台service。

使用ContentProvider

Intent提供了在不同应用之间的一种运行时绑定（run-time binding）机制。Intent最大用处是用来启动其他的activity。

### Intent用法？

2015/9/4, vicky

其他提问方式：

Intent寻找目标组件的两种方式？

在使用中，使用Intent的形式？

对于明确指出了目标组件名称的Intent，称之为“显式Intent”。

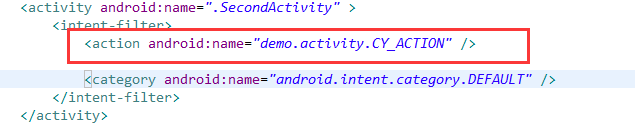
对于没有明确指出目标组件名称的Intent，称之为“隐式 Intent”。

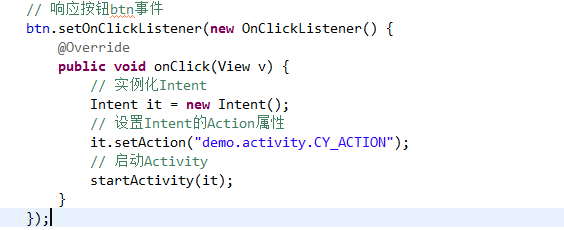
举例时，使用Activity跳转。

显示Intent：



隐式Intent：





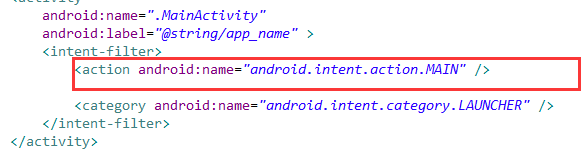
### Intent由什么组成？

其他提问方式：Intent的属性是什么？Intent的构成是什么？

Intent的组成包含

（1）、Action：指明要实施的动作是什么。

最常见的的action.MAIN，自定义一个action。



（2）、Data： 要事实的具体的数据，一般由一个Uri变量来表示。

（3）、Category：一个字符串，包含了关于处理该intent的组件的种类的信息。

（4）、Type：显式指定Intent的数据类型（MIME）。

### Intent的解析机制

2015/9/4, vicky

对于直接Intent，Android系统不需要去解析，因为目标组件已经很明确。

Android需要解析的是隐式Intent，通过解析将 Intent映射给可以处理此Intent的Activity、Service或Broadcast Receiver。

Intent的解析机制主要是通过查找已经注册在AndroidManifest.xml中的所有IntentFilter以及其中定义的Intent，通过PackageManager来查找能处理这个Intent的component，最终找到匹配的Intent。

在这个解析过程中，Android是通过Intent的action、type、category这三个属性来进行判断的，判断方法如下：

1.1 如果Intent指明定了action，则目标组件的IntentFilter的action列表中就必须包含有这个action，否则不能匹配；

1.2 如果Intent没有提供type，系统将从data中得到数据类型。和action一样，目标组件的数据类型列表中必须包含Intent的数据类型，否则不能匹配。

1.3 如果Intent中的数据不是content:类型的URI，而且Intent也没有明确指定type，将根据Intent中数据的scheme（比如 http:或者mailto:）进行匹配。同上，Intent 的scheme必须出现在目标组件的scheme列表中。

1.4 如果Intent指定了一个或多个category，这些类别必须全部出现在组建的类别列表中。比如Intent中包含了两个类别：LAUNCHER\_CATEGORY和ALTERNATIVE\_CATEGORY，解析得到的目标组件必须至少包含这两个类别。

说明：

PackageManager能够得到当前设备上所安装的application package的信息。

### Activity使用Intent传递数据量很大（>=8MB）时，程序会发生ANR，或者直接报错误？

传递信息时，一般采用intent绑定bundle的方式传值，但在不要用bundle传递大容量数据：目前测试的是8M字节左右。

Bundle只能传递较小的数据量，如何通过Intent传递大数据？

数据量不是很大是的情况下：

方式1：压缩数据。例如，压缩图片。但数据也不能太大。（推荐）

方法2：使用Parcelable接口或Serializable接口序列化对象。但数据也不能太大。（推荐）

更大是数据怎么办？

方式3：使用临时文件或数据库作为中转。（推荐）

将需要传递的数据写在临时文件或者数据库中，再跳转到另外一个组件的时候再去读取这些数据信息，这种处理方式会由于读写文件较为耗时导致程序运行效率较低。

方法4：使用静态变量。

方法5：使用全局变量。（推荐）

Application来保存全局变量，并且在package创建的时候就存在了。

调用Context的 getApplicationContext或者Activity的getApplication方法来获得一个Application对象，以设置或读取全局变量的值。

使用全局变量与使用静态变量更适合：

Java中的静态变量在这里是适合的，但其值在Activity调用了System.exit(0)或finish()后就丢失了。使用ApplicationContext这种全局变量方法相对静态类更有保障，直到应用的所有Activity全部被destory掉之后才会被释放掉。

### 如何使用Context.getApplicationContext()在同一个进程共享内存？

2015/9/5, vicky

通过建立建立一个全局可用的实例。

第一步：创建一个属于自己程序android.app.Application的子类，为想要共享的private全局变量增加setter和getter方法。

public class MyApp extends Application{

private String globalVariable;

public String getGlobalVariable() {

return globalVariable;

}

public void setGlobalVariable(String globalVariable) {

this.globalVariable = globalVariable;

}

}

第二步：manifest中为application标签的android:name属性设置为这个类。

**<application** android:name=".MyApp" android:icon="@drawable/icon" android:label="@string/app\_name"**>**

第三步：使用Context.getApplicationContext()方法获取这个实例，进而获取其中的变量。

// ============使用全局变量传递参数==============

MyApp myApp = ((MyApp) getApplicationContext());//获得我们的应用程序MyApp

myApp.setGlobalVariable("全局变量");

Intent intent = new Intent();

intent.setClass(ListDemoActivity.this, GlobalActivity.class);

startActivity(intent);

// ============接收全局变量的参数==============

MyApp myApp = ((MyApp) getApplicationContext());

String globalVariable = myApp.getGlobalVariable();

## 四大组件之Service

### 什么是Service？

2015/9/4, vicky

其他提问方式：Service的概念。

Service是一个应用组件，它在后台执行耗时操作，且不提供用户界面。其他组件可以启动一个服务，即使用户切换到其他应用，该服务仍可在后台继续运行。另外，组件可以绑定某个服务，进行来与它通信，甚至包括执行进程间通信(IPC).例如，某个服务在后台可能在处理网络传输，播放音乐，执行文件I/O,或者与内容提供者交流.

Service在默认情况下，Service运行在应用程序进程的主线程中。如果直接在Service中执行耗时操作，会发生ANR错误。执行耗时操作，要在Service中新建线程。

A Service is an application component that can perform long-running operations in the background and does not provide a user interface. Another application component can start a service and it will continue to run in the background even if the user switches to another application. Additionally, a component can bind to a service to interact with it and even perform interprocess communication (IPC). For example, a service might handle network transactions, play music, perform file I/O, or interact with a content provider, all from the background

Service是一个应用组件，它在后台执行耗时操作，且不提供用户界面。其他组件可以启动一个服务，即使用户切换到其他应用，该服务仍可在后台继续运行。另外，组件可以绑定某个服务，进行来与它通信，甚至包括执行进程间通信(IPC).例如，某个服务在后台可能在处理网络传输，播放音乐，执行文件I/O,或者与内容提供者交流.

### Service可以两种方式（Started和Bound）工作吗？

2015/9/4, vicky

Although this documentation generally discusses these two types of services separately, your service can work both ways—it can be started (to run indefinitely) and also allow binding. It's simply a matter of whether you implement a couple callback methods: onStartCommand() to allow components to start it and onBind() to allow binding.

虽然本文档概要的分别讨论服务的这两种类型，但你的服务可以以两种形式工作（同时不能以两种方式工作）—它可以被started(无限运行)，也可以binding。它取决回调函数的实现:onStartCommand()允许组件启动它，onBind()方法允许组件绑定它。

### 默认情况，Service运行在哪个线程中？

2015/9/4, vicky

其他提问方式：

如何防止使用Service时发生ANR错误？

启动服务，不用多线程，界面是否会阻塞?

Service会新开进程或者线程么？

Caution: A service runs in the main thread of its hosting process—the service does not create its own thread and does not run in a separate process (unless you specify otherwise). This means that, if your service is going to do any CPU intensive work or blocking operations (such as MP3 playback or networking), you should create a new thread within the service to do that work. By using a separate thread, you will reduce the risk of Application Not Responding (ANR) errors and the application's main thread can remain dedicated to user interaction with your activities.

注意:服务运行在它的宿主进程的主线程中——服务不创建它自己的线程，并且不会运行在一个独立的进程里(除非另有指定)。也就是说，如果服务要执行任何高消耗CPU的操作,或者阻塞操作(比如，MP3播放，网络操作),应该给服务创建一个新的线程来处理那些事情。通过使用独立线程，你将能减少ANR风险,并且能让主线程专注使用Activity与用户交互。

### 启动服务后，按Home键，服务是否还在运行

2015/9/4, vicky

在运行。

### 启动服务后，退出进程，是否服务还会运行？

2015/9/4, vicky

不在运行。

### 服务是个新的进程么，服务是个新的线程么？

2015/9/4, vicky

服务不是一个新的进程，也不是一个新的线程 。 与当前进程绑定的。

服务运行在它的宿主进程的主线程中——服务不创建它自己的线程，并且不会运行在一个独立的进程里(除非另有指定)。

### Service的分类？

2015/9/4, vicky

（1）、按照启动方式分类，Started Service和Bound Service

（2）、按服务性质分类，Local Service和Remote Service。

（3）、按照实现方法分类，Java Service和Native Service。

### 如何创建Service？

2015/9/4, vicky

前提是service已经在manifest中定义，并且已通过继承service class来定义具体service。

（1）、对于startedService，Context.startService()

调用者与服务之间没有关联，即使调用者退出，服务仍可运行。

传给startService()方法一个Intent来启动一个服务。这个Intent指定Service，并且包含了服务使用的一些数据。该服务将会在onStartCommand()方法中接收到这个Intent意图。

当一个服务被启动started后，它就拥有了独立于启动它的组件的生命周期，并且即使启动它的组件被销毁，它也可以无限地运行在后台。因此，当Service工作完成后，它应该自己调用stopSelf()方法结束自己，或者其他的组件调用stopService()方法停止这个Service。

（2）、Context.bindService()

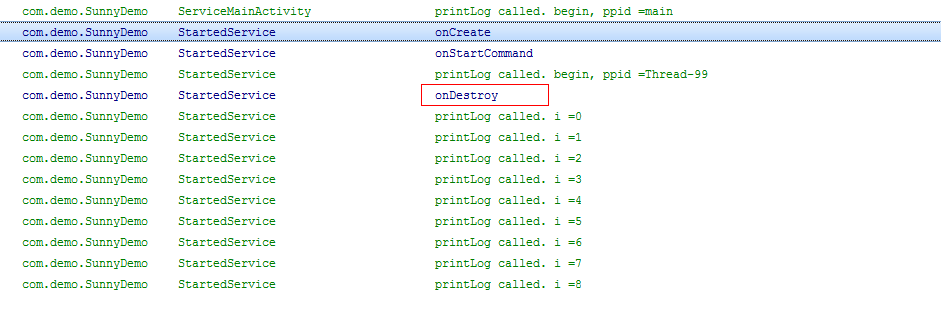
调用者与服务绑定在一起，所有的调用者一旦退出，服务也就终止。

### 如何停止Service？

（1）、对于startedService，对于startedService，如果调用stopSelf(startId) 或stopService(intent)，即使任务执行完了，也不调用onDestroy。

当Service工作完成后，它应该自己调用stopSelf()方法结束自己，或者其他的组件调用stopService()方法停止这个Service。

如果onStartCommand中开启了线程，调用stopSelf(startId);或stopService(intent)后，Service虽然执行onDestroy，执行完onDestroy表示Service被销毁了，但子线程继续运行，直到子线程运行结束，或主线程挂掉。



（2）、对于bounded service，不用手动停止。当所有Clients

### 如何公开Service被其他应用访问？

2015/9/4, vicky

在manifest中，service通过定义intent filters，让其他组件使用隐式Intent来调用服务。

如果声明的intent filters与其他应用传给startService()方法的Intent相匹配，则安装在用户设备上的任何应用的组件都潜在地可启动这个服务.

官方上直说了startService()方法来启动，那么我可以认为能公开Started Service，但对于bound service不太清楚。

### 如何私有Service只能被自己的应用访问？

2015/9/4, vicky

在manifest中设置Service 的android:exported属性为"false", 能确保Service为应用私有，即使Service提供了intent filters。

官方文档中并没有指明是started Service，还是bounded Service。

### 如何使用IntentService

2015/9/4, vicky

创建继承于IntentService的Service，然后实现abstract onHandleIntent(Intent)，在里面可以执行耗时操作。

### 什么是IPC？以及Linux IPC的种类：

2015/9/4, vicky

IPC是Inter-Process Communication，进程间通信。

Linux IPC有管道，socket，共享内存，BInder，四种方式。

Binder机制是Android IPC一种机制。Binder是google为Android进程间通信传递的改造出来的一种机制。需要Binder IPC硬件支持。

### 如何使用service何时使用多线程处理请求？何时使用工作队列处理请求

2015/9/4, vicky

（1）、每个调用onStartCommand()时，若同时执行多个请求，则可以为每个请求创建一个新线程，并且以正确的方法运行它。

创建Service实例时，继承于Service而不是继承IntentService.。

（2）、每个调用onStartCommand()时，若等到前一个请求执行完成再执行下一个，则使用工作队列处理请求。

创建Service实例时，继承于IntentService。

IntentService是Service的子类，是一个startedService。

### IntentService实现原理

2015/9/4, vicky

其他提问方式：

如果你去实现IntentService，该怎么实现？

IntentService如何实现执行耗时操作的？

对于每个发起的请求，它使用一个工作线程执行工作，并且每次只处理一个。

（1）、创建一个独立于应用主线程的工作线程HandlerThread，在这个作线程中处理传给onStartCommand()方法的所有意图 。

（2）、创建一个工作队列，这个队列每次传一个Intent给onHandleIntent()方法实现,所以不必担心多线程的问题。

（3）、提供一个默认的onStartCommand()方法实现，该方法来发送Intent到工作队列，然后(由工作队列)发到onHandleIntent()方法实现

（4）、提供一个默认的onBind()方法的实现,该实现返回null.

（5）、当Handler处理完所有请求后停止Service，Handler调用stopSelf。

Looper + Handler + HandlerThread

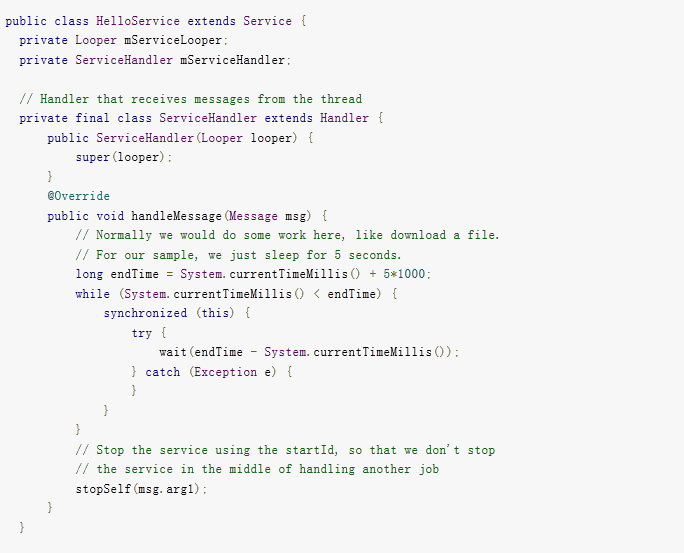
IntentService如何实现执行耗时操作的？

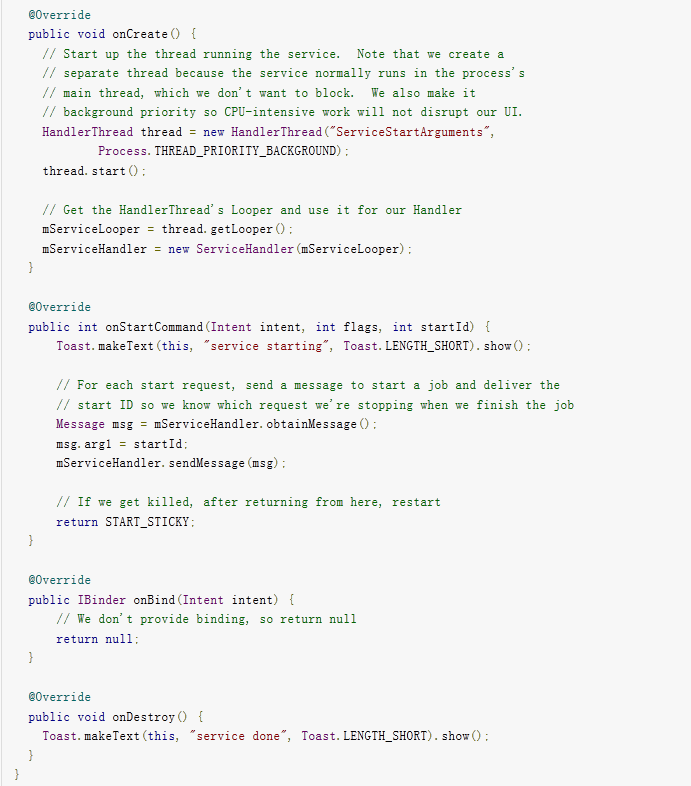
IntentService中包含Looper + Handler + HandlerThread

onCreate中创建HandlerThread线程，用来执行耗时操作。

handleMessage中执行完毕消息后，自动调用stopSelf结束当前Service。

onStartCommand → 工作队列 → onHandleIntent → HandlerThread去处理。





IntentService的实现原理？

IntentService中使用了消息队列Looper、消息池Message、Handler、HandlerThread。

onCreate方法时时创建并启动HandlerThread实例，用于处理耗时操作、初始化Looper、初始化ServiceHandler。

onStartCommand方法调用onStart方法，onStart方法中使用ServiceHandler发送Message，该message的obj为intent对象。

ServiceHandler的handleMessage用于处理接到消息时，首先onHandleIntent方法操作后，然后调用stopSelf关闭service。

onDestroy方法时关闭Loop。

### 当描述了在系统杀掉本服务的情况下，系统应该如何继续该服务？

onStartCommand()方法必须返回一个整数值。这个整数值，描述了在系统杀掉本服务的情况下，系统应该如何继续该服务

onStartCommand()该方法返回的值必须是下面常量中的一个：

START\_NOT\_STICKY

If the system kills the service after onStartCommand() returns, do not recreate the service, unless there are pending intents to deliver. This is the safest option to avoid running your service when not necessary and when your application can simply restart any unfinished jobs.

在onStartCommand()返回后，如果系统杀掉了该服务，除非传递pending intents，系统不重创建服务。这是没有必要时避免运行你的服务，以及当应用可以简单地重新开始没有完成的工作时的一种安全的选择。

START\_STICKY

在onStartCommand()返回后，如果系统杀掉了该服务，重创建服务并调用onStartCommand()方法，但是不传递最后的意图.而是系统传一个null，给调用的onStartCommand()。除非传递pending intents，那样的话，这些意图被传送。

这适用于类于多媒体播放器的服务，它们不需要执行命令，但是要无限运行并等待工作。

START\_REDELIVER\_INTENT

在onStartCommand()返回后，如果系统杀掉了该服务，重新创建服务并调用onStartCommand()方法，并将最后一个传给服务的意图传给onStartCommand()方法。

这适合那要立即恢复并积极主动去执行工作的服务，比如下载文件。

### Service的生命周期

2015/9/4, vicky

Service有两种生命周期，startedService和BoundedService。



startService：stopSelf()，stopService()。

bindService

onStart()旧版本，onStartCommand()新版本

### 使用Service来写一个音乐播放程序，要求分别使用startService和bindService方法 。

### Bound Service的分类？

2015/9/4, vicky

Bound Service的分类？

如果是Bound Service是同一进程，就是Local BoundService。

如果是Bound Service是不同进程，就是Remote BoundService。

### Local BoundService的创建和使用？

2015/9/4, vicky

BoundedService:

第1步：通过继承Binder的子类，定义和实现IBinder接口。

第2步：通过onBind()方法，发布IBinder接口。

第3步：在AndroidManifest中注册服务

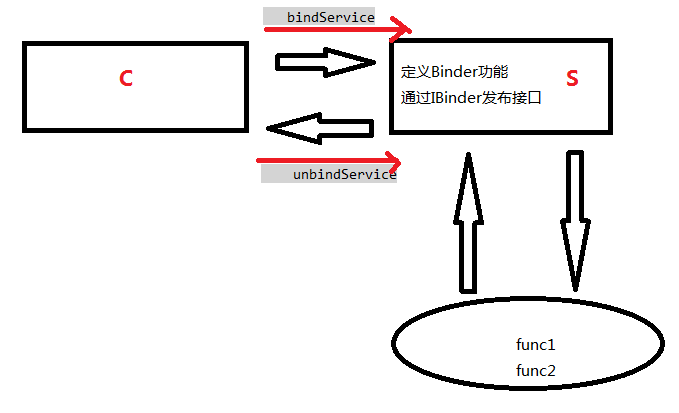
第4步：Activity中使用ServiceConnection管理连接。

第5步：Activity中onStart，Bind to LocalService（bindService）(经常这样使用)

第6步：调用Toast.makeText(ServiceMainActivity.this, myBoundedService.getSpeakToHello(), Toast.LENGTH\_LONG).show();

第7步：Activity中onStop，Unbind from the service（unbindService）(经常这样使用)

ServiceManager是由C++或C语言



### 对于Bound Service，该Service运行在主线程，还是子线程？

运行在主线程。所以，如果是耗时操作，需要在Service中新建线程。

### 先执行Bound Service中onBind，还是先执行Activity中ServiceConnection->onServiceConnected？

2015/9/4, vicky

先执行onBind

### 先执行Bound Service中onUnbind，还是onDestroy？

2015/9/4, vicky

先执行onUnbind。

### 介绍Binder？

2015/9/4, vicky

Linux IPC（进程间通信）的机制有：管道，socket，共享内存，Binder。

经过对Binder IPC机制进程改造，使得Binder机制成为Android IPC一种机制。

Binder IPC机制需要硬件支持。Android框架从低到高包括linux底层 → libraray层 → application framework层，都有Binder对应的支持。

Binder进程间通信机制（Binder IPC）是一个种C/S模式。Bound Service作为Server端。调用Service方作为Client端。Binder实现进程间通信，它是在同一个进程，或在不同进程的通信机制。Activity跳转也是通过IBinder实现的，因为要跳转的Activity可能是同一个进程，也可能是不同进程的。

Bound Service定义和实现Binder功能，使用onBind()方法通过IBinder接口发布Binde功能。Client通过IBinder使用功能。

Binder IPC机制需要ServiceMananger支持。ServiceMananger是android中比较重要的一个进程，它是在init进程启动之后启动

ServiceManager由底层实现，由C或C++语言实现。

ServiceManager在Binder系统中相当与DNS，Server会先在这里注册，然后Client会在这里查询服务以获得与Service所在的Server进程建立通信的通路。

ServiceMananger是android中比较重要的一个进程，它是在init进程启动之后启动，它是用来管理系统中的service。比如：InputMethodService、ActivityManagerService等。在ServiceManager中有两个比较重要的方法：add\_service、check\_service。service需要通过add\_service把自己的信息注册到ServiceManager中，当需要使用时，通过check\_service检查该service是否存在。

Bound Service注册Service时，通过add\_service 注册到ServiceManager。当需要使用时，通过check\_service检查该service是否存在。

### start Service和Bound Service的区别？

（1）、两种服务最本质区别 ：boundService 其实是有一个客户端，服务器端的概念，借助于Binder 来通讯。

（2）、生命周期不同。

StartedService 和启动它的Activity不同生共死，当前进程关闭，StartedService才会销毁。

BoundService 和启动它的Activity同生共死。

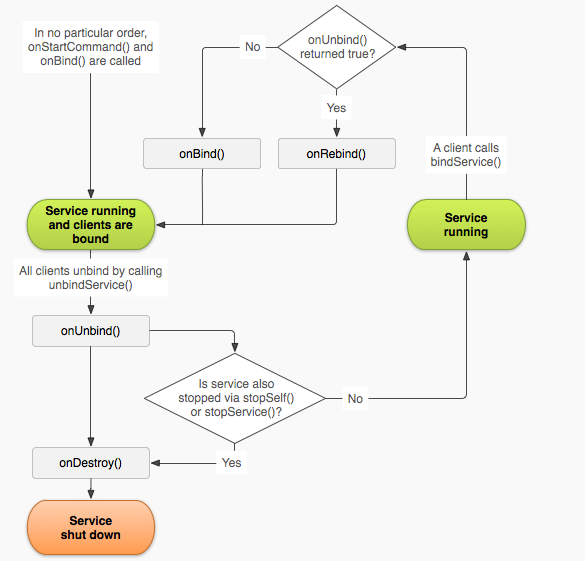
（3）、再次使用时不同。

无论是start or bind，只要service没有实例化，就都会被create 。

StartedService启动时，每次都会调用onStartCommand 。

BoundService绑定时， 则如果已经绑定，就不再调用onBind了，而是调用rebind。

rebind调用的场景：新的client连接bound Service时，调用rebind。



### IPC通信，即进程间通信的方式有几种？

2015/9/4, vicky

Android IPC通信有两种：Messager（底层也是基于AIDL），AIDL。

### 使用BoundService，如何提供IBinder接口？

2015/9/4, vicky

其他提问方式：

提供IBinder接口的方式？

当创建一个提供绑定的service时，你必须提供给一个IBinder，来提供客户和service交互的接口。

有三种方法可以定义这个接口：

（1）、继承Binder类。

适用情况：service是应用私有的，并且和客户端在同一个进程中运行。

（2）、使用Messenger

适用情况：跨进程通信，且不需要多线程，为单线程处理且线程安全。

Messenger在一个单独的线程中创建一个所有客户端请求的队列，所以service一次只接收一个请求。

使用Messenger的方法底层结构实际上是基于AIDL的。

这是执行跨进程通信（interprocess communication (IPC)）的最简单的方法，因为Messenger把所有的请求排列进一个单独的线程，所以在设计service时不用为了线程安全而做特殊的设计。

service定义一个 Handler用来响应不同类型的 Message 对象。

这个Handler是 Messenger和客户分享一个 IBinder的基础，让客户端可以使用Message 对象向service发送命令。

另外，客户端也可以定义自己的Messenger，这样service就可以发送信息给客户端。

（3）、使用AIDL。

适用情况：跨进程通信，能够多线程处理且是线程安全的。

AIDL (Android Interface Definition Language)执行分解对象的工作，它把对象分解成primitives（原生事务），操作系统可以理解并将这些primitives跨进程分组（marshall）来执行IPC。

service同时处理多个请求。

要直接使用AIDL，必须创建一个.aidl文件，定义编程接口。Android SDK使用这个文件来生成抽象类，实现接口和处理IPC，你可以在你的service中继承它。

注意：多数应用不应该使用AIDL去创建一个bound service，因为它需要多线程，会导致一个复杂得多的实现。

### AIDL的全称是什么？如何工作？能处理哪些类型的数据？接口实现原则？支持的哪些类型数据

2015/9/5, vicky

第1问：AIDL的全称是什么？

AIDL (Android Interface Definition Language)，android接口定义语言。

第2问：如何工作？

AIDL是进程间通信IPC机制的实现。

AIDL是一种IDL 语言，用于生成可以在Android设备上两个进程之间进行进程间通信(IPC)的代码。

AIDL IPC机制是面向接口的，它是使用代理类在客户端和实现端传递数据。

AIDL的IPC的机制和COM或CORBA类似, 是基于接口的，但它是轻量级的。

接下来，以BoundService的通信机制为例子。

第3问：能处理哪些类型的数据？

AIDL语法很简单,可以用来声明一个带一个或多个方法的接口，也可以传递参数和返回值。

下面是些AIDL支持的数据类型:

（1）、简单数据类型，不需要import。

（2）、String, CharSequence，不需要特殊声明。

（3）、List, Map和Parcelables类型，这些类型内所包含的数据成员也只能是简单数据类型。

第4问：实现接口的原则:

（1）、抛出的异常不要返回给调用者，跨进程抛异常处理是不可取的。

（2）、IPC调用是同步的。执行耗时操作时，IPC接口应该新建线程处理。

（3）、不能在AIDL接口中声明静态属性。

第5问：支持的哪些类型数据？

in表示是输入参数，out表示是输出参数，可以传出某个参数值，inout表示输入输出参数，先传入参数，处理后传出。

### Remote Local BoundService的创建和使用？（TODO）

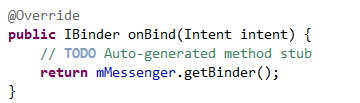
2015/9/4, vicky

一、如何使用Messenger实现Activity与Service通讯？

（1）、Service中：

第1步：定义Messenger和Handler，用来处理client发送到消息，以及回传给client消息。

第2步：OnBinder()中通过Messenger 来提供IBinder接口。



第3步：在清单文件中注册Service。

（2）、Client中：

第1步：新建ServiceConncteion服务连接。

第2步：Activity onStart中，调用bindService(intent, connection, Context.BIND\_AUTO\_CREATE) 绑定Service。

第3步：执行IBinder接口：使用Messenger发送Message向service发送消息。（同步接口，耗时操作必须新建线程）

第4步：在Activity onStop中，调用unbindService(connection)解除绑定。

二、如何使用ANDF实现Activity与Service通讯？

客户端与服务器端交互设计时，先设计服务端，客户端完成访问功能。

（1）、Service中：

Service的实现：

第1步：注册.adif文件。

第2步：创建.adif文件对应的Service。包括，使用IBinder接口、调用OnBinder()来发布IBinder接口

第3步：在清单文件中注册Service。

（2）、Client的实现：

第1步：注册.adif文件，与server保持一致（包名一致、内容一致）；

第2步：使用ServiceConncteion建立服务连接。

第2步：Activity onStart中，调用bindService(intent, connection, Context.BIND\_AUTO\_CREATE) 绑定Service。

第4步：执行IBinder接口（同步接口，耗时操作必须新建线程）

第4步：在Activity onStop中，调用unbindService(connection)解除绑定。

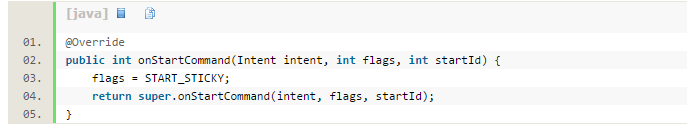
两者没有本质区别。

### 如何防止service被杀死？

2015/9/4, vicky

音乐后台播放，并在通知栏中有进度通知。点击通知栏能进度音乐画面。当系统资源紧缺时，很有可能被系统杀死。

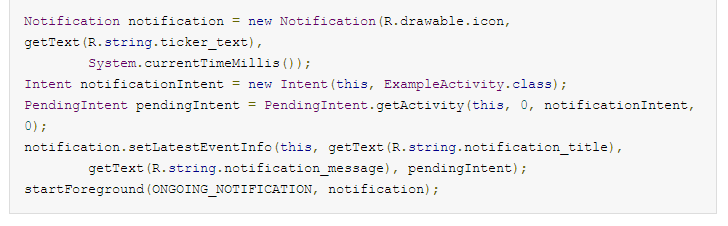
方法1：onStartCommand方法，返回START\_STICKY 或START\_REDELIVER\_INTENT。



方法2：使用startForeground 让Service前台运行，防止进程被杀死。使用stopForeground去掉。

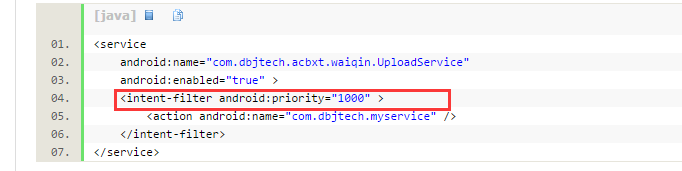
startForeground减少被杀死的几率。但是如果在极度极度低内存的压力下，该service还是会被kill掉。

在onStartCommand里面调用 startForeground，然后再onDestroy里面调用stopForeground。



方法3：使用manifest中Service的android:priority属性提高提升service进程优先级，减少被杀死的几率。

但是如果在极度极度低内存的压力下，该service还是会被kill掉。



方式4：在onDestroy方法里重启service

service +broadcast 方式，就是当service走ondestory的时候，发送一个自定义的广播，当收到广播的时候，重新启动service。

### 什么是Native Service？何时会用到Native Service？如何调用

2015/9/4, vicky

Native Service用C++/C语言写的Service。

当需要性能高的时候，会用到Native Service。

java使用JNI调用C/C++。

### Service运行中如何与用户交互？

2015/9/4, vicky

Service使用Toast 或通知跟UI交互。

### 如何在Service和线程之间选择?

2015/9/4, vicky

其他提问方式：

什么时候使用service？什么时候使用thread？

（1）、执行耗时任务时，若不需要UI用户交互，则使用service。

由于使用service仍然会运行在主线程，所以要在service中创建一个线程执行密集或阻塞操作。

（2）、执行耗时任务时，若需要UI用户交互，则使用thread，例如：AyncTask / [HandlerThread](file:///E:\DOC\docs_Android%204.4.2%EF%BC%88API%2019%EF%BC%89\reference\android\os\HandlerThread.html) / Thread。

### 播放音乐，使用Service？还是Bound Service？还是Thread？

2015/9/4, vicky

（1）、支持后台播放时？

使用Start Service。因为当Activity onDestroy后，Sart Service仍在存活，还支持后台播放音乐程序。

（2）、当仅仅在Activity运行时才播放音乐？

使用多线程。

在Activity的onCreate()方法：创建一个Thread。

在Activity的onStart()方法：运行。

在Activity的onStop()方法：停止。

使用Bound Service。

bindService()：Activity的onStart方法中调用。

unBindService()：Activity的onStop方法中调用。

### BoundService的bindService()和unBind()在Activity哪个生命周期函数调用？

2015/9/5, vicky

其他提问方式：

BoundService的bindService()和unBind()一般合适会调用？

Activity的onStart()方法：bindService()

Activity的onStop()方法：unBindService()

### IntentService的使用方法与Service与什么区别？

2015/9/4, vicky

IntentService中自动包含一个HandlerThread线程，该线程是一个独立于主线程外的单线程。

IntentService使用时重写onHandlerIntent方法，在该方法中完成耗时操作，不用额外创建Thread、Handler。

## 四大组件之ContentProvider

### 什么是Content Provier？

2015/9/7, vicky

Content providers manage access to a structured set of data.

内容提供者通过标准接口来访问结构化数据（数据库数），屏蔽掉了存储实现细节。

唯一的共享程序间数据方式。

Android中没有公共存储区域。

Content Provider内容提供者，被访问。

Content Resolver内容解析者，访问。

### ContentProvider的作用？或者为什么需要ContentProvider？

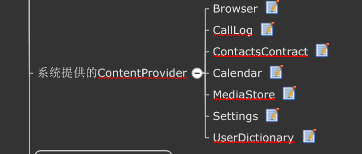
2015/9/7, vicky

默认时，系统为每个程序分配唯一的用户ID ，各个程序使用自己的用户IDF，来防止进程间随意通信。

ContentProvider实现进程间通信，提供进程间通信的标准接口，允许直接修改其他程序对应的磁盘文件。

### 举例说出系统自带的ContentProvider？

2015/9/7, vicky



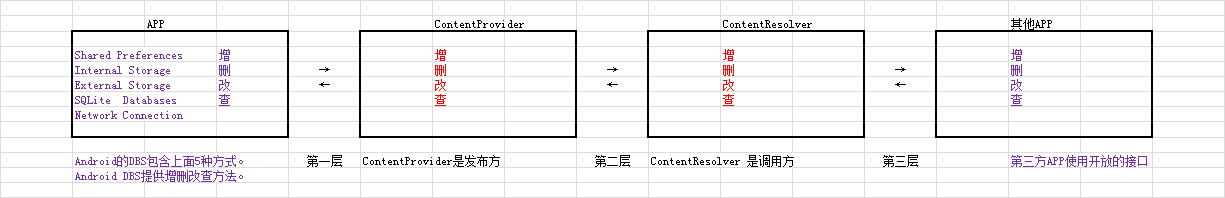
浏览器程序，通话记录，通讯录，日历，媒体存储，系统设置，用户字典（输入法输入历史、输入法记录用户习惯）。

### ContentProvider （内部）机制是什么？

2015/9/7, vicky

ContentProvider（服务器） 和 ContentResolver（客户端）可以看成一个C/S结构。

ContentProvider 和 ContentResolver一般是在不同进程，用到了进程间通知机制：Ibinder。



### 使用进程间通信机制的IBinder哪些？

ContentProvider和ContentResolver的C/S结构。

Broadcast和BroadcastReceiver的C/S结构。

BoundService的C/S结构。

Activity跳转到其他应用界面。

### ContentProvider需要手动实例化吗？

2015/9/7, vicky

其他提问方式：

ContentProvider对象何时被实例化？

ContentProvider在开机时实例化了，并开始监听请求，不必启动对应APP。对应的Service也是一样的。

CR是用来匹配。返回对象是一个Cursor。

### ContentProvider调用的时候宿主进程生命周期？

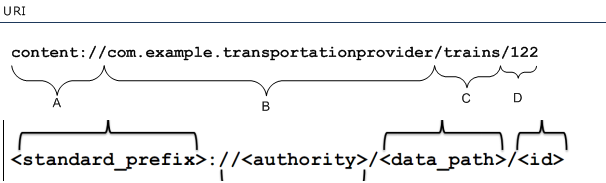
2015/9/7, vicky

接受调用的时候启动宿主进程，操作完毕后，宿主进程为empty。

### ContentProvider对外提供的数据模型形式是什么？

ContentProvider（服务器） 和 ContentResolver（客户端）可以看成一个C/S结构。

三要素：网址、发送请求、返回处理请求的结果。



A为标准前缀，B为包名（相当于数据库地址），C相当于数据库表名，D相当于表中具体记录主键iD。

### 编写ContentResolver的增删改查

第1步：

如何使用ContentProvider？

ContentProvider作为服务器：

定义ContentProvider

（1）、继承于ContentProvider定义Provier，

（2）、实现方法

onCreate() which is called to initialize the provider

query(Uri, String[], String, String[], String) which returns data to the caller

insert(Uri, ContentValues) which inserts new data into the content provider

update(Uri, ContentValues, String, String[]) which updates existing data in the content provider

delete(Uri, String, String[]) which deletes data from the content provider

getType(Uri) which returns the MIME type of data in the content provider

（3）、注册Provier。

<provider

android:name=*".PersonProvider"*

android:authorities=*"cn.com.karl.personProvider"* />

使用ContentResolver

（1）、使用getContentResolver()得到ContentResolver。

（2）、使用Uri进行操作。

ContentResolver contentResolver = getContentResolver();

Uri insertUri = Uri.parse("content://cn.com.karl.personProvider/person");

ContentValues values = new ContentValues();

values.put("name", "wangkuifeng");

values.put("age", 23);

Uri uri = contentResolver.insert(insertUri, values);

## 四大组件之Broadcast

### 注册广播的方式？以及其优缺点？

2015/9/5, vicky

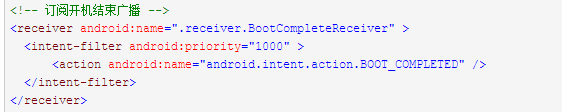
注册方式有两种：

（1）、静态注册，这样的广播成为常驻型广播。

在manifest文件里使用receiver节点注册，使用intent-fliter过滤出要接受的广播。

缺点：这种方式注册的广播属于系统级广播。它会始终处于活动状态。应用没打开也能收到广播。所以，比较消耗电量、内存等资源。

常见的应用：通过监听开机广播来启动应用。



（2）、动态注册，这样的广播成为非常驻型广播。

使用代码进行注册。使用registerReceiver(receiver, filter)注册广播。

优点：广播跟随程序的生命周期。应用启动后才监听，关闭后就不监听的效果。所以，比较节省电量、内存等资源。

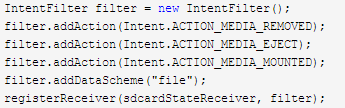
### 如何动态注册广播接受者、取消注册广播接受者?

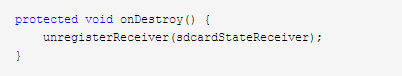
2015/9/5, vicky

注册广播接受者 registerReceiver(receiver, filter);

取消注册广播接受者 unregisterReceiver(receiver);

通常是配对操作：onCreate/onDestroy，onStart/onStop。





### 为何引用广播？

2015/9/7, vicky

其他提问方式：

Broadcast Reveiver作用。

在系统内通知状态、判定执行状态。

（1）、系统通知自身的执行状态。例如，开机，准备关机，电量低等。

（2）、APP自定义执行状态，根据状态执行某种操作。例如，发消息提示更新界面。

（3）、实现高效通信。

### 广播的使用方法：

2015/9/7, vicky

第1步：定义一个广播接受者，继承于BroadcastReceiver。重写onReceive方法（接受广播的地方）。

第2步：注册广播接受者（两种方式）。

第3步：发送广播（定义Intent）。

第4步：取消动态注册的注册广播接受者。

Intent intent = **new** Intent(MainActivity.**this**, MyBroadcastReceiver.**class**);

intent.putExtra("content", "你好，这是一个广播示例");

// 发送广播

sendBroadcast(intent);

**public** **class** MyBroadcastReceiver **extends** BroadcastReceiver {

**public** MyBroadcastReceiver() {

}

// 接收广播

@Override

**public** **void** onReceive(Context context, Intent intent) {

// **TODO** Auto-generated method stub

Toast.*makeText*(context, intent.getStringExtra("content"), Toast.*LENGTH\_SHORT*).show();

}

}

### 广播的种类？

2015/9/7, vicky

按发送方式：

（1）、普通广播：接受顺序是无序的。sendBroadcast()

（2）、有序广播：指定接受广播顺序。sendOrderedBroadcast()。使用AbortBroadcast终止广播传播。

（3）、粘性广播：接受收不会移除消息体。sendStickyBroadcast()。Android 5.0/API 21中deprecated。

（4）、粘性有序广播：接受收不会移除消息体，指定接受广播顺序。sendStickyOrderedBroadcast()。Android 5.0/API 21中deprecated。

按范围：

（1）、全局广播（Android 系统）。使用Context.注册、发送和取消注册的广播。

（2）、局部广播（APP 内部使用）。使用LocalBroadcastManager. 注册、发送和取消注册的广播。

### Broadcast消息处理完了，消息体是如何内删除的？

2015/9/7, vicky

消息处理完了，消息体是如何内删除的？

默认情况下，消息体是被Looper接受到的，消息体被Receiver接受后，消息体是被Receiver移除的。

一般消息是什么样的结构？

1. 消息体Message，
2. 消息队列MessageQueue，线性结构，先进先出。
3. 消息管理Looper 监听，一个接一个取出和分发。分发给接受者。

消息队列和消息管理，是系统维护，不用干涉。

要做是定义消息体，使用Broadcast Receiver得到消息体。

### 广播的发送消息和接受消息是同步还是异步的？

2015/9/7, vicky

### 发送消息是同步还是异步？

2015/9/7, vicky

发送消息是异步的，把消息放到消息队列就可以，然后立刻返回。（asynchronous）

接受消息是同步的。在主线程中。onReceive方法接受消息执行时间，一定要小于10秒。如果处理时间很长，需要手动开一个多线程。

### 广播的实际使用例子？

2015/9/7, vicky

（1）、android.intent.action.BOOT\_COMPLETED广播，实现开始启动。

（2）、自定义广播，接受广播后，更新list。

（3）、垃圾短信拦截。

## 五大组件之多线程

### 手机开发中为何会使用多线程？

2015/8/30, viky

默认时，应用程序的组件运行在同一个线程和进程，称之为UI线程，又成为主线程，或UI主线程。

当应用程序登录时，系统为该程序创建唯一的主线程。它负责分发事件给对应的组件，包括绘制事件。

系统不会为每个组件创建新的线程，所有组件均运行在UI主线程。

如果在线程执行了耗时操作，将阻塞主线程，那么什么事件不能分发，包括绘制事件。此时界面会响应缓慢、甚至发生ANR错误。

因此，执行耗时操作要放到非主线程中运行，从而避免阻塞主线程。

为了保证良好的用户体验，建议对超过50ms的操作，都使用线程处理。

### 什么是ANR错误？以及发生ANR错误的条件是什么？

2015/8/30, viky

ANR：application not responding。

发生ANR错误的条件是：5秒消息没有接收，或者10秒钟消息没有处理完。

### 常见的错误有哪些？为何会产生，以及如何解决？

2015/9/5, vicky

（1）、FC，Forceed Close，程序异常退出。

原因是：程序出现空指针或未捕获的异常。

解决：①编写程序时逻辑连贯，思维缜密。②能捕获异常

（2）、ANR，application not responding，程序无响应。

原因：5秒消息没有接收，或者10秒钟消息没有处理完。

1. 应用主线程卡住，对其他请求响应超时。②死锁。③系统反应迟钝。④CPU负载过重。

解决：①不要阻塞主线程。②程序优化。

（3）、uncaused exception，程序异常退出，

原因是：程序出现异常。异常为捕获。

解决：①编写程序时逻辑连贯，思维缜密。②能捕获异常

Android平台程序崩溃大家都应该遇到过，force close和ANR应该是大家遇到较多的。

这里把Android平台程序崩溃的各种类型做一个简述和原因列举。

1.ANR（可见ANR）：

发生场景：应用发生ANR。

崩溃症状：系统弹出窗口询问用户选择“Force Close”或者“Wait”。

       “Force Close”将杀掉发生ANR的应用进程。“Wait”将会等待系统择机恢复此应用进程。

发生原因：（1）应用主线程卡住，对其他请求响应超时。（2）死锁。（3）系统反应迟钝。（4）CPU负载过重。

2.Force Close：

发生场景：应用进程崩溃。

崩溃症状：系统弹出窗口提示用户某进程崩溃。

发生原因：空指向异常或者未捕捉的异常。

3.Tombstones：

发生场景：Native层崩溃

崩溃症状：如果发生崩溃的native层和UI有关联（比如Browser），我们可以在UI上发现这个崩溃。

    如果发生崩溃的native层是在后台并且和UI没有直接联系，那么对于用户来说是不可见的，如果是debug版本可能会有Log打印出当时的底层现场。

发生原因：各种各样，需要具体情况具体分析。

4.系统服务崩溃（System Server Crash）：

发生场景：系统服务是Android核心进程，此服务进程发生崩溃。

崩溃症状：手机重启到Android启动界面

发生原因：（1）系统服务看门狗发现异常。（2）系统服务发生未捕获异常。（3）OOM。（4）系统服务Native发生Tombstone。

5.Kernel Panics：

发生场景：Linux内核发生严重错误

崩溃症状：手机从bootloader开始完全重启

发生原因：（1）Linux内核内存空间发生内存崩溃。（2）内核看门狗发现异常。（3）空指针操作内核。

### 如何分析android内存？

### 为了保证良好的用户体验，建议对超过50ms的操作，都使用线程处理。为什么是50ms？

2015/8/30, viky

1000ms/50ms = 20FPS。

20FPS的含义是，每秒钟屏幕更新 20 幅画面。

得到的帧数20FPS是人眼的视觉暂留特性的最低要求。小于20FPS，会感觉画面会反应迟滞、甚至卡、更甚至是出现ANR错误提示框，大于30就看不出来了。

### 什么时候要使用多线程？

2015/8/30, viky

当执行耗时操作时，为了避免阻塞主线程，就需要多线程。

（1）、耗时IO操作。包括文件操作、网络操作、数据库操作等。

（2）、耗时复杂运算。

（3）、耗时定时操作。

### Android中异步操作的三大方式，也就是如何使用多线程或异步任务？

2015/8/30, viky

方式1：Thread + Handler。

使用Thread执行耗时操作，执行完成后，使用Handler把操作结果通知给主线程或其他线程，但一般是通知给主线程。

方式2：AsyncTask

使用AsyncTask执行耗时操作。一般是，边执行边更新UI进度。当结束后，更新进度，并通知用户完成。例如，下载。

方式3：AsyncTaskLoader 和 CursorLoader。适合于加载耗时比较长的数据时使用。

（1）、Loader（装载器）在，API11 ，Android 3.0之后开始出现。Loader使得在Activity或Fragment中使用异步的方式加载数据。

（2）、AsyncTaskLoader是一个抽象loader，继承于Loader，它提供了AsyncTask去执行操作。

AsyncTaskLoader包含一个类LoadTask ，LoadTask继承于AsyncTask<Void, Void, D>，并实现了Runable 接口。

final class LoadTask extends AsyncTask<Void, Void, D> implements Runnable {

（3）、CursorLoader是AsyncTaskLoader的子类。它查询ContentResolver对象，并且返回一个Cursor对象。它是基于AsyncTaskLoader对象用后台线程来执行游标查询，以便应用程序的界面不会被死锁。

使用CursorLoader是从ContentProvider对象中异步加载数据的最好方法，而不是通过Fragment或Activity的APIs来执行被管理的查询。

### 使用Loader相对于ReQuery加载数据的特点？也就是说有哪些优点？

2015/8/30, viky

Loader使得在Activity或Fragment中使用异步的方式加载数据。

Loader的特点如下：

（1）、装载器对于每个Activity和Fagment都是有效的。

（2）、装载器提供异步数据加载的能力。

（3）、装载器监视数据资源，并且当内容改变时发送新的结果。

（4）、在配置改变后重建的时候，装载器自动重连最后的装载器游标，因此，不需要重新查询数据。

Cursor.requery()：不推荐使用。

（1）、当数据量很大时，查询时间会比较长。

（2）、数据库发生变化后，Cursor需要重新加载数据。

（3）、配置变化后，Cursor重新加载数据。

### 如何保证线程安全？

2015/8/30, viky

（1）、只能在主线程才能更新UI。不要在UI线程外直接操作UI

（2）、不要Block UI Thread

### 多线程如何与主线程或界面进行交互？即，子线程如何把结果通知给主线程？

2015/8/30, viky

（1）、Activity. runOnUiThread(Runnable action)

（2）、Handler.post、Handler.send

（3）、View.post(Runnable) 、View.postDelay(Runnable , long)

（4）、AsyncTask

以上所有方式，最终是调用enterQueue方法。

### Handler是什么？

2015/8/30, viky

Handler允许发送和处理Message（消息）和Runnable objects，这里的Runnable objects与消息队列（MessageQueue）关联。

每一个Handler实例只与一个线程和这个线程的消息队列相关联。

当创建一个新Handler时，它被绑定到创建这个handler的线程或这个线程的消息队列。从那一刻起，它会传递Message和runnables到这个消息队列，并且当Message和runnables从消息队列中出来时执行他们。

### Hanlder作用:

2015/8/30, viky

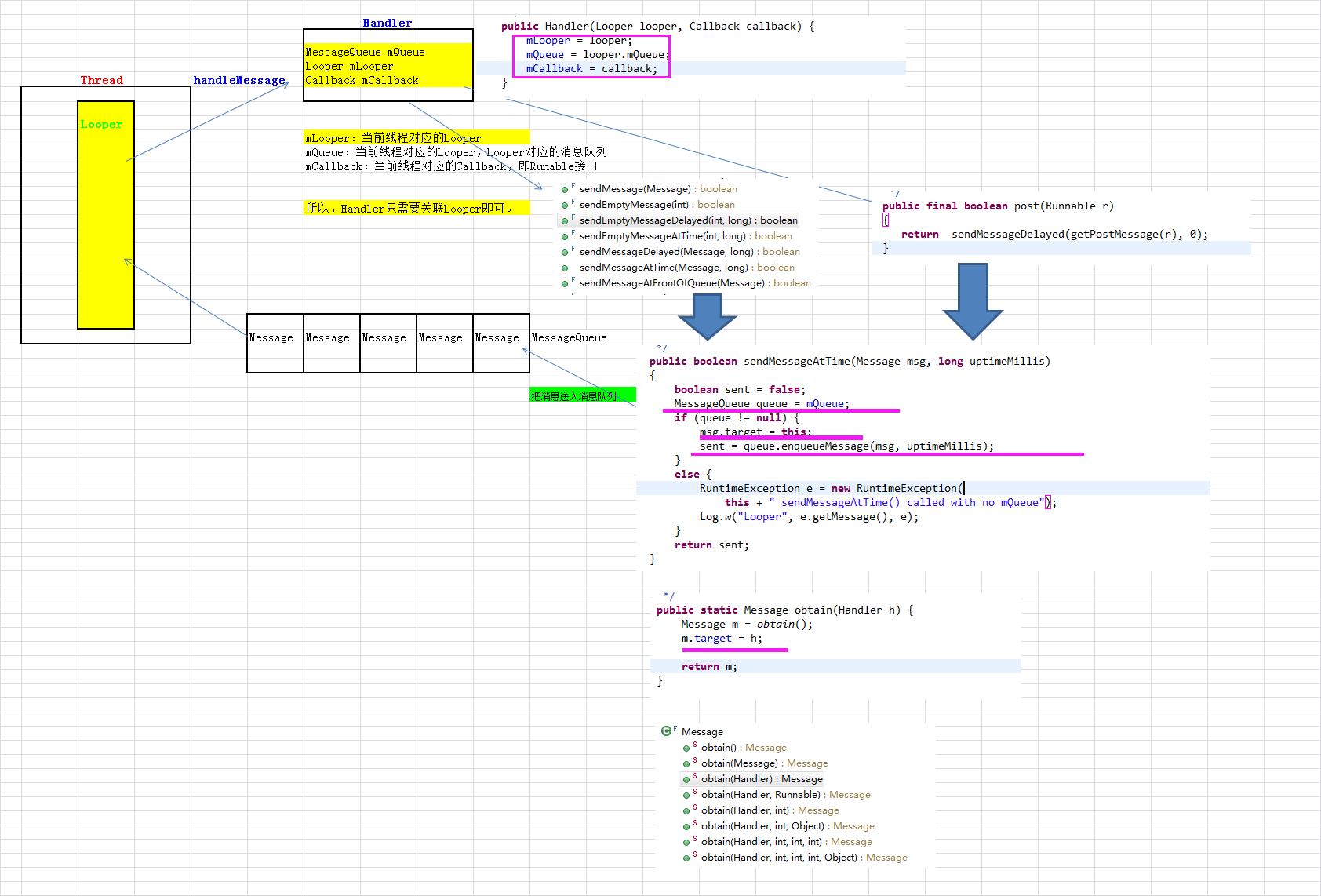
（1）、执行计划任务。在预定的实现执行某些任务，可以模拟定时器

（2）、线程间通信。最经典操作是，子线程完成耗时任务，执行完成后，会发送

（3）、确保操作始终在某个特定的线程中运行。例如，数据库的查询要在同一个子线程中运行。

### Handler的实现原理？或 简述Android的消息机制？或者MessageQueue、Looper、Handler、或者，与Thread的关系以及作用？或者Handler与线程的关系？或者，Handler 异步实现方法？

2015/8/30, viky



Android的消息机制和Window的消息机制很相似，都是使用消息队列实现的。

Android的消息机制涉及到了Thread（线程）、Looper （循环）、MessageQueue（消息队列）、消息（Message）、Handler.

其中，

Thread中关联唯一的Looper、MessageQueue 。

Handler通过Handler(Looper looper)方法关联唯一的Thread。

MessageQueue包含Message。

（1）、消息（Message）：包含被传送数据。

（2）、消息队列（MessageQueue）：管理消息（Message），使得传递的消息有序，否则顺序一乱图片显示不正确。

考虑在消息队列中找消息，如果没有则创建，如果有则利用。这样消息的回收能提高效率，防止程序出现内存泄露。

（3）、Handler：Handler允许发送和处理Message（消息）和Runnable objects。 Hanlder负责存消息、取消息、管理消息队列的出队和入队操作。

（4）、Looper：不断循环并分发消息。获取和分发消息的循环是死循环。Looper获取消息队列后，不断从消息队列中获取消息。当消息为空时，退出循环。当消息非空时，分发消息。

### Looper的实现原理是什么？

2015/8/30, viky

Looper的实现和Windows编程的消息机制非常的类似。

Looper不断从循环，从消息队列中获取消息，并把消息发送给对应的Handler，然后Handler来处理消息。

### Handler一定是在主线程中吗？或者，如何在子线程中创建Handler?

2015/8/30, viky

不一定。

通过HandlerThread创建一个含有Looper的子线程：HandlerThread handlerThread = new HandlerThread("threadone");

启动线程后：handlerThread.start()

使用getLooper方法得到这个子线程的Looper： Looper looper = handlerThread.getLooper();

把Looper指定到Handler：final Handler myHandler = new Handler(looper) {

那么，Handler就关联到了这个子线程。

### Handler的post 和sentMessage方法，使用的是一个队列还是两个？

2015/8/30, viky

是一个消息队列。最终都会调用setMessagAtTime方法。这个方法中执行enqueueMessage方法将Message送到关联的消息队列中。

### 子线程中建立一个handler，然后sendMessage会怎样？

2015/8/30, viky

程序会崩溃。

使用new Handler()，默认会关联到主线程。

如果关联一个Handler到子线程，应该关联Looper到这个Handler。

### Handler中不同的消息有可能被同时处理么？

不会。按MessageQueue先进先出的顺序执行。

### 多个Handler使用同一个Thread。同一个rThread中可以有多个Handler吗？

2015/8/30, viky

可以。

### 多个Handler使用同一个Thread。同一个Thread中使用多个Handler的Message会混淆么？为什么？

2015/8/30, viky

不会。因为Message的构造方法中会执行.target = handler。这个target就是要Looper分发Message时，要分发到的handler。

### 如何得到一个Looper？

2015/8/30, viky

（1）、Loopr.myLooper()：返回跟当前线程关联的Looper。

static Looper myLooper()

（2）、getMainLooper()：返回主线程关联的Looper。

static Looper getMainLooper()

（3）、通过HandlerThread.getLooper()获取指定线程的Looper。

### 如何指定Handler运行的线程？

第一步：得到要运行线程的Looper。

如果是子线程，一定要执行Looper. Prepare()方法得到子线程的Looper。执行Looper.loop()方法使得让Looper开始工作，从消息队列里取消息，处理消息。

第二步：调用Handler(Looper)以确定其使用的Looper

### 子线程建立handler , 构造的时候传入主线程的Looper?

2015/8/30, viky

Handler关联的是主线程。消息会在主线中被处理。

### 当Handler在处理消息时，如果收到removeMessage()， 会中断这个处理么？或者，对于正在执行的Message，调用removeMessage会有什么后果？

2015/8/30, viky

不会。

removeMessage只能移除队列中的Message

### 使用Handler是异步的，它会建立新线程么？

2015/8/30, viky

不会。通过设置Looper来选择其依附的线程。

### AsyncTask机制实现方式是什么？

2015/8/30, viky

Android为了降低异步操作开发难度，结合Handler和线程池（最大支持128个线程），提供了AsyncTask。AsyncTask后台执行耗时操作同时能与更新UI。

### AsyncTask与线程及线程池的关系

2015/8/30, viky

AsyncTask内部封装了关联主线程的InternalHandler类型的Handler、线程池。Android4.4.2中线程池能最大放入123个线程。

### AsyncTask是多线程么？

2015/8/31, viky

AsyncTask不是多线程。

### 如何使用AsyncTask ？

2015/8/31, viky

规则1：AsyncTask必须在主线程创建

规则2：execute() 必须在主线程执行，而且只能执行一次。execute()执行时，只有当AsyncTask处于pending状态才会执行。

规则3：不要自己调用onPreExecute(), onPostExecute(), doInBackground(), onProgressUpdate()。

以上，只有doInBackground()在子线程中执行，其他方法都是在UI线程执行。

### AsyncTask使用时需要重写哪些方法？

2015/8/31, viky

onPreExecute()： 在UI线程执行。在后台线程执行之前执行。

doInBackground()： 被后台线程执行。执行后台耗时操作。执行过程中可以调用publicProgress(Progress…)来更新任务的进度。

onProgressUpdate()：在UI线程执行，更新UI进度

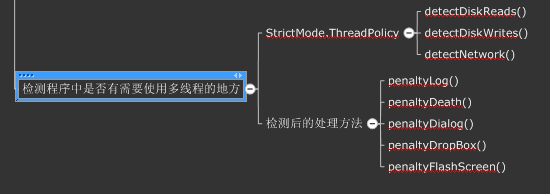
onPostExecute()： 在UI线程执行。后台线程完成后执行。相当于Handler 处理UI的方式，

onCancelled(): 在UI线程执行，要做的操作。

以上，只有doInBackground方法是必须的。

### 检测程序中需要多线程（异步任务）？

2015/8/31, viky



检测数据库、文件等硬盘操作、网络操作是否需要多线程？

### 检测内存泄露？

检测是否有Sqlite对象或要未关闭的对象泄漏内存？

sdk/docs\_Android 4.4.2（API 19）/reference/android/os/StrictMode.html



### AsyncTask三个状态：

2015/8/31, viky

pending(等待执行) ，running , finished

execute()执行时，只有当AsyncTask处于pending状态才会执行。

### AsyncTask能创建多少个Thread实例？

2015/8/31, viky

Android 4.4.2中AsyncTask中线程池最大能创建128个Thread实例。如果超过这个数目，待创建线程需要等待一些已存在线程执行结束，才能进入线程池。

### AsyncTask会创建线程么？

2015/8/31, viky

AsyncTask包含Handler、ThreadPoolExecutor。

ThreadPoolExecutor会创建一个新的Thread。

ThreadPoolExecutor → ThreadFactory通过newThread方法调用new Thread()。这个Thread

### 与Handler相比， AsyncTask消耗资源情况如何,为什么？

2015/8/31, viky

AsyncTask包含Handler、ThreadPoolExecutor。

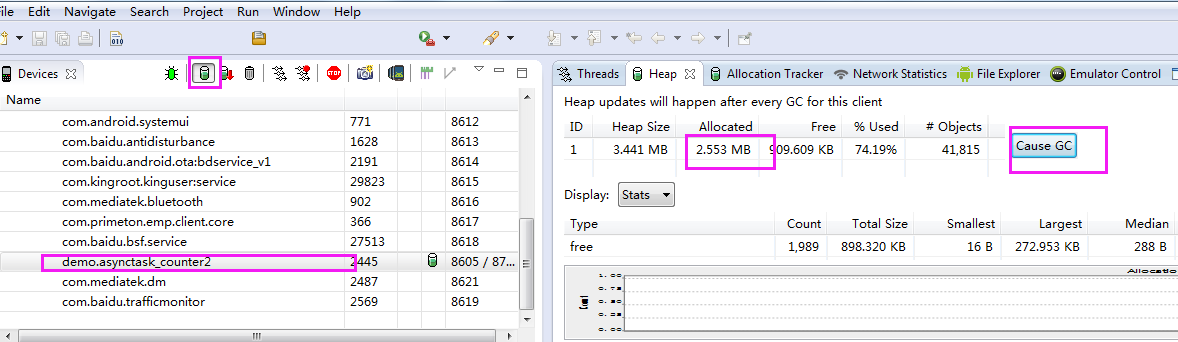
ThreadPoolExecutor会创建一个新的Thread。

ThreadPoolExecutor → ThreadFactory通过newThread方法调用new Thread()。这个Thread

### 如何分析**进程**的内存占用？

2015/8/31, viky

Update Heap图标



### 如何正确选择使用AsyncTask还是(Thread +) Handler?

2015/8/31, viky

异步任务允许边执行耗时操作边与UI交互。使用异步任务交少时，代码比较优雅简洁。

异步任务封装后灵活性不够。

异步任务消耗更多资源。

Handler使用更灵活。

Handler消耗内存较少。

Handler适合于使用多线程较多的地方，

选择异步任务：执行耗时时同时与UI交互，子线程与UI交互。

选择Handler：执行耗时任务的地方比较多，子线程与子线程交互，子线程与UI交互。

## UI

### 如何使得Button仅仅执行LongClick，不执行Click事件？

2015/9/6, vicky

OnLongClickListener和OnClickListener均继承与onTouchEvent。

button long click事件包含了click事件。如果同有Click 和Long Click，系统先处理View.OnLongClickListener，再处理View.OnClickListener。因此，消息机制分发完LongClick后继续分发Click事件。

如果只执行onLongClick，OnLongClickListener执行完毕后返回true。

### ImageButton如何设置背景？

2015/9/6, vicky

ImageButton，通过android:src="@drawable/button\_icon" 或者setImageResource(int)加载图片。

### Button，使用图片和文本时，android:drawableLeft属性和android:background属性的区别？

2015/9/6, vicky

使用android:drawableLeft等属性在文字的左侧等加载图片，通过android:background="@drawable/button\_custom"显示背景图片。

使用android:drawablePadding="1dp"设置图片和文本的距离。

类似android:drawableLeft属性：android:drawableTop、drawableBottom、drawableRight、drawableStart、drawableEnd。

### ImageButton和Button，如何展示状态变化？

2015/9/6, vicky

其他提问方式：

如何简单实现扁平化设计？

第一步：在res下建立drawable文件夹。并建立一个xml配置文件，例如button\_custom.xml。

方式1：设置包含selector 的xml，selector中设置获取焦点以及常规状态的布局。

方式2：设置包含selector 的xml，设置点击状态和常规状态的shape(solid + corners)（扁平化设计）

<?xml version=*"1.0"* encoding=*"utf-8"*?>

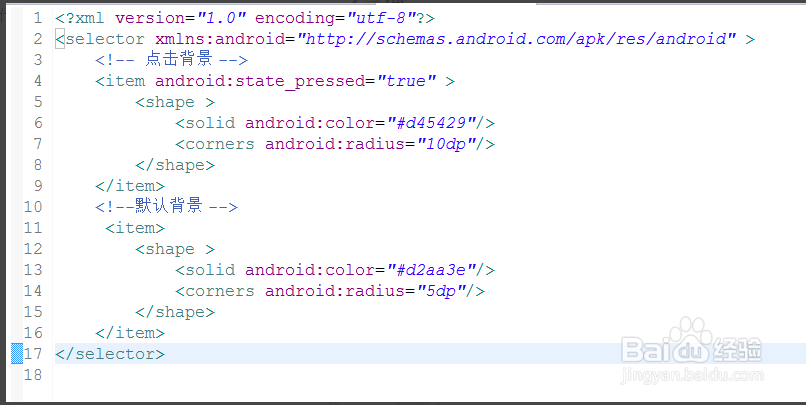
<selector xmlns:android=*"http://schemas.android.com/apk/res/android"*>

<item android:drawable=*"@drawable/button\_pressed"* android:state\_pressed=*"true"*/>

<item android:drawable=*"@drawable/button\_focused"* android:state\_focused=*"true"*/>

<item android:drawable=*"@drawable/button\_default"*/>

</selector>



solid属性为边框的颜色。

corner属性为圆角值。

第二步：在要自定背景状态的按钮上加上android:background="@drawable/button\_custom"属性。

### 扁平化设计与拟物化设计的选择？

扁平化设计(flat design)

扁平化设计聚焦两点：视觉的极简主义，功能的最优表达。用最简单的用户界面达成最详备的功能。

即：去掉了各种不必要的视觉效果，更专注于内容呈现和交互。

何时扁平化？：

（1）、流线型的设计，提高响应速度和加载时间

（2）、简化设计，减少设计的复杂度，提高速度。

（3）、内容过载，信息量大使，整洁、有序的设计风格时可以让用户读者快速发现需要的内容。

何时不需要扁平化？

设计高度依赖视觉、交互元素，来来吸引特定用户。例如儿童。

### ToggleButton（开关按钮）如何切换背景

方式1：ToggleButton通过代码切换背景

方式2：ToggleButton通过drawable XML文件（selector）切换背静。

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<selector xmlns:android=*"http://schemas.android.com/apk/res/android"*>

<item android:drawable=*"@drawable/on"* android:state\_checked=*"true"*/>

<item android:drawable=*"@drawable/off"*></item>

</selector>

### ToggleButton的常见事件？

2015/9/6, vicky

mToggleButton.setOnCheckedChangeListener(new OnCheckedChangeListener()

### 如何设置background？

2015/9/6, vicky

View android:background 背景色、背静图片（单个背静图片，或者XML布局）

### CheckBox（多选按钮）的主要事件？如何改变状态？

2015/9/6, vicky

主要事件：OnClickListener、OnCheckedChangeListener

改变状态：使用setChecked(boolean) or toggle()改变状态。

### RadioButton（单选按钮）的主要事件？如何设定默认选择？

2015/9/6, vicky

主要事件：OnClickListener、OnCheckedChangeListener

使用android:checked="true"设置某个RadioButton为默认选中。

多个RadioButton通常与RadioGroup同时使用。

### 列出常见的Adapter？

2015/9/6, vicky

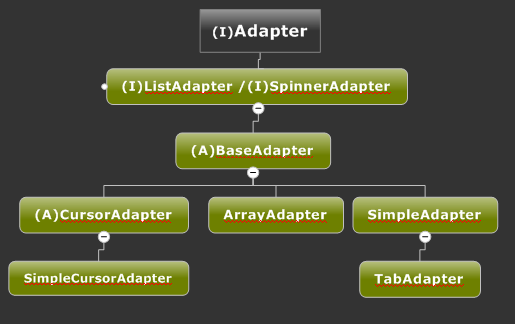
比较重要的Adapter

BaseAdapter：所有Adapter的基类。

ArrayAdapter：表结构为多行多列。

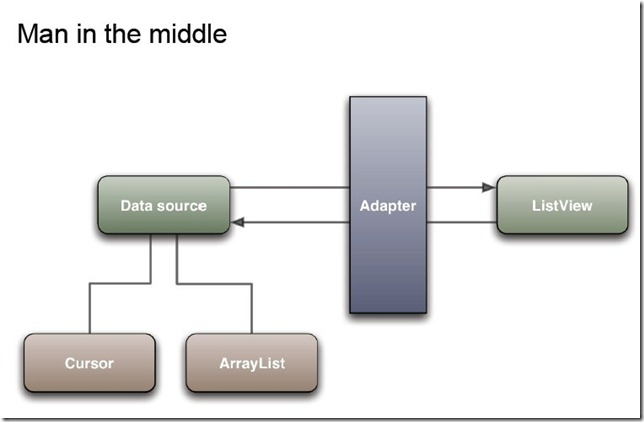
CursorAdapter：数据库相关。表结构为多行多列。

SimpleAdapter：



### Adapter模型？

2015/9/6, vicky



### 如何使用Adapter？

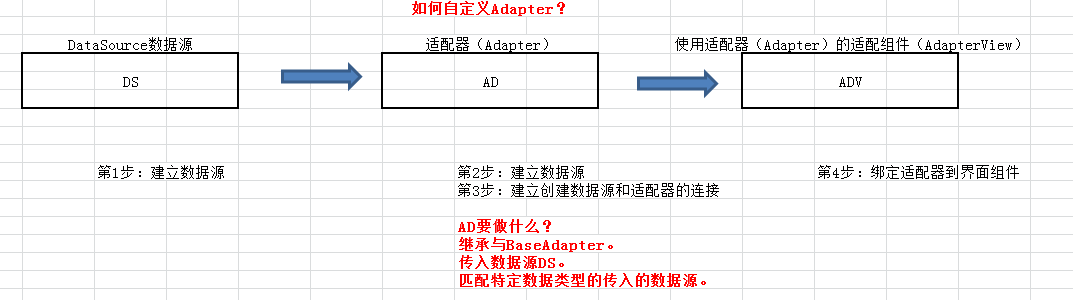
2015/9/6, vicky

1，建立数据源

2，建立adapter

3，建立adapter – 数据源连接

4，绑定adapter到界面组件



### Spinneer下拉框的主要事件？

实现AdapterView.OnItemSelectedListene接口

<Spinner android:id="@+id/spinAddCus\_cusSort"

      android:prompt="@string/cus\_cusSort"

      android:entries="@array/cus\_sort"

     android:layout\_height="wrap\_content"

    android:layout\_width="fill\_parent">

</Spinner>

android:entries：spinner的数据源。也可以通过adapter获取数据源。

android:prompt：标题。必须要引用 strings.xml 中资源ID，而不能在这里直接用 raw text。

entries是entry的复数

entry美: ['entri] 英: ['entri] n.参赛；词条；记录；参赛作品网络登入；条目；入口词形变换entries

### 进度条ProgressBar

### SeekBar（拖动条）的常见事件？

2015/9/6, vicky

SeekBar.OnSeekBarChangeListener

### RatingBar（评分条的常见事件？

2015/9/6, vicky

RatingBar.OnRatingBarChangeListener

### EditText（输入框）的常见属性？

android:hint="请输入内容" 提示内容

android:inputType输入类型。输入框的类型决定了在这个区域中能允许输入的字母类型，它可能唤起虚拟键盘去最优化使得频繁使用的字母最优化

android:singleLine="true" 设置单行输入

android:ems = "8" 设置TextView或者Edittext的宽度为8个字符的宽度。

android: maxLength： 最大长度

android:imeOptions="actionSend" 改变回车键的类型，例如发送；搜索等。

android:inputMethod 指定输入法。如果找不到会报错。

### 自定义UI组件 – 自定义Toast？

2015/9/6, vicky

Toast是一个Android自带layout中textview。

Toast默认显示位置：屏幕底部、垂直居中对齐。

定义TextView。

弹出自定义的toast：使用LayoutInflater填出view。

### 自定义UI组件 – 自定义Dialog？

### 自定义UI组件 – 自定义进度条？

如何自定义一个组件使用，例如实现一个带百分比的进度条。

方案1：继承ProgressBar。然后重写setProgress和onDraw。onDraw负责绘制百分比矩形text，setProgress负责传递数值。必要时，通过回调函数的形式增加新的事件。

方案2：自定义一个layout，包括一个ProgressBar和textview。(推荐)

### ListView为什么会分页加载？如何实现分页？

2015/9/6, vicky

一、分页加载的原因：

减少服务器的交互压力。

减少客户端的交互压力、存储压力、流量消耗。

二、如何实现分页？

为了达到分页，需要记录一下东西：

（1）、记录已加载的总页数。

交互时，把要请求的页数、每页请求的个数传给服务器。

翻页时，仅当第一次需要设置适配器，其他使用notifyDataSetChanged（// 数据变动后，通知View刷新）。

（2）、记录服务端的总页数。当已加载页数<总页数时，才会执行请求。

分页的方动作：

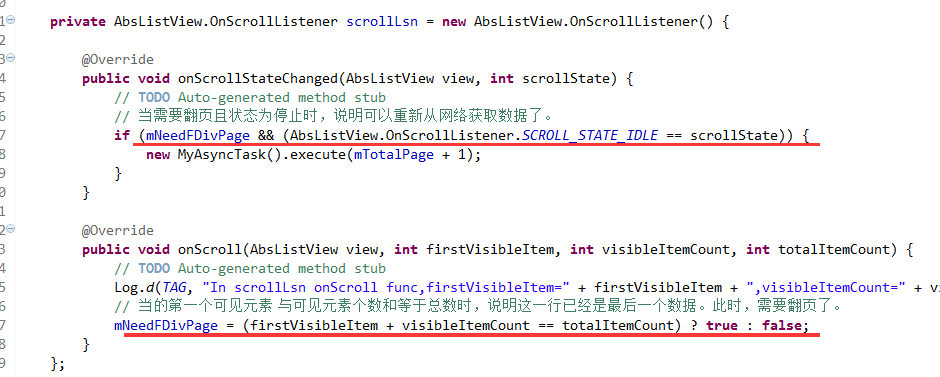
（1）、滑动最低端时，通过下拉刷新。

判断何时应该翻页？

在OnScrollListener中做如下判断？需要 + 滑动停止

条件1：当(firstVisibleItem + visibleItemCount == totalItemCount)。此时，需要翻页了。

条件2：当滑动停止（最后页且手已经松开）时，说明可以开始分页、重新从网络获取数据了。





### ListVIew下拉刷新的原理？

2015/9/6, vicky

扩展了ListView ，实现下拉刷新PullToRefresh

下拉刷新原理：

（1）、设置头部显示的layout。

（2）、重新ListView

在OnTouchEvent监听事件中，监听手势各种动作。根据动作，设置layout显示。

当向下拖动，满足一定条件时，头部视图执行动画、更新文字等处理。

添加一个OnRefreshListener接口，定义一个回调函数onRefresh（），在回调方法中添加异步操作处理。

开源组件：PullToRefresh

### scrollView的使用？

2015/9/6, vicky

scrollView主要用于大的信息浏览。比如新闻浏览。ScrollView只出现在API说明中。

HorizontalScrollView（纵向滑动）

ScrollView（横向滑动）的使用：

（1）、滚动的条件：内容大于屏幕物理尺寸。

（2）、 A ScrollView is a FrameLayout。

（3）、ScrollView 中只能加入一个子元素，通常是一个 垂直方向的LinearLayout。可以在LinearLayout中放入各种负责的元素。

（4）、使用ScrollView时不能使用ListView。如果使用，需要解决ScrollView和ListView冲突。

### 如何使用ExpandableListView？

2015/9/6, vicky

实现ExpandableListView，需要定义一个继承BaseExpandableListAdapter的Adapter，提供一级数据List<String>和二级数据List<List<String>>。

例子：QQ好友列表、有道云的二级目录。

### 如何实现滑动切换界面？

2015/9/6, vicky

TabHost

ViewPager

Fragment+FragmentTabHost（推荐使用）

### 时钟的格式？

2015/9/6, vicky

DigitalClock数字时钟

TextClock：文字时钟。Android4.2(API Level 17)。（推荐），代替DigitalClock。

AnalogClock：模拟时钟

TimePicker：时间选择器。

DatePicker：日期选择器。

### 使用的布局？

2015/9/6, vicky

（1）、相对布局RelativeLayout、

（2）、线性布局LinearLayout、

（3）、网格布局GridLayout，

（4）、表格布局TableLayout）

（5）、帧布局 FrameLayout

（6）、绝对布局 AbsoluteLayout

原则：复杂界面用TableLayout，简单界面用RelativeLayout

### 线性布局常见的属性？

<!—横向 -->android:orientation="horizontal"

<!—纵向 --> android:orientation="vertical"

gravity 与 layout\_gravity的区别？

gravity，如果有子对象，表示子对象在父对象中的位置。如果没有子对象，表示内容的位置。

layout\_gravity：表示当前控件位于父控件的位置。

margin属性与padding属性的区别？

margin：外边距。

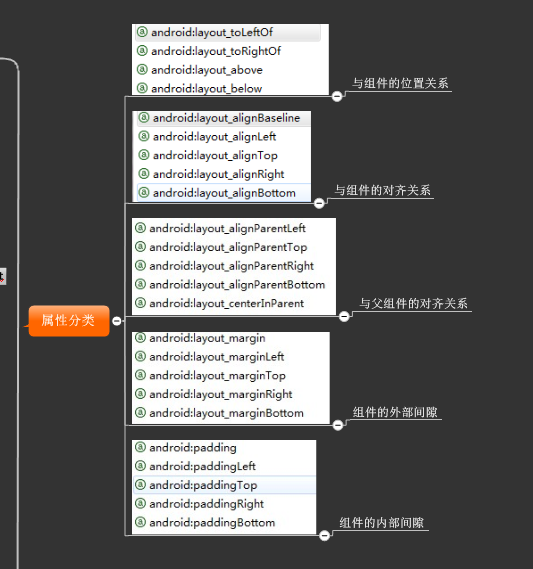
padding：内边距。

layout\_weight：比重，比重越大，所占有的空间越大。当为0时，不拉伸。为了保证界面适应性，至少有1列是自动延伸的。

### RelativeLayout相对布局的常见属性？

2015/9/6, vicky

相对布局的对齐、位置属性比较重要。



### TableLayout（表格布局）的常见属性？

2015/9/6, vicky

TableLayout属性(也叫全局属性)：\*代表所有列？

android:shrinkColumns：设置可收缩的列，(内容过多，则收缩，扩展到第二行，控件没布满TableLayout时不起作用)

android:stretchColumns：设置可伸展的列，(有空白则填充)

android:collapseColumns：设置要隐藏的列(索引列从0开始)

android:collapseColumns="2"

android:shrinkColumns="0,1"

android:stretchColumns="0,1,2"

说明：

列可以同时具备stretchColumns及shrinkColumns属性

一个TableRow就代表TableLayout的一行。

TableLayout行数等于TableRow个数+直接在TableLayout中控件个数

TableLayout列数等于最多子控件的TableRow的列数

直接在TableLayout加控件，控件会占据一行，默认宽度占据一行。

问题：TableRow的属性？

TableRow是LinearLayout的子类，

TableRow方向为android:orientation="horizontal"， layout\_width为 MATCH\_PARENT，layout\_height为 WRAP\_CONTENT，这3个属性不用设置。

问题：内部控件属性？

android:layout\_column -------该单元格在第几列显示

android:layout\_span -------该单元格占据列数，默认为1

### 如何指定border？

方法1：使用view。

方法2：使用背景色设置各行原色。各个行之间留些margin。

### fragment的生命周期？

2015/9/6, vicky



### 菜单分类？

OptionMenu（下拉式菜单，最常规）、ContextMenu（上下文菜单）、PopupMenu（弹出式菜单）。

### OptionMenu的使用？

第1步：定义menu xml。

第2步：在Activity中，

1. 通过onCreateOptionsMenu填充menu布局。
2. 重写onOptionsItemSelected处理menu Item点击方法。

### ContextMenu的使用？

ContextMenu类似于鼠标右键的长按操作。但是通常用在AdapterView，最常用的是ListView和GridView。

当长按ListView 的Item，系统会调用onCreateActionMode显示contextual action bar。

如何在ListView或GridView中使用ContextMenu？

（1）、定义menu xml。

（2）、在Activity中，

①实现AbsListView.MultiChoiceModeListener接口。并重写接口的onActionItemClicked方法，onCreateActionMode方法。

onCreateActionMode方法用于填充menu布局，onActionItemClicked处理menu Item点击方法。

1. 设置setChoiceMode()为CHOICE\_MODE\_MULTIPLE\_MODAL。

listView1.setChoiceMode(ListView.CHOICE\_MODE\_MULTIPLE\_MODAL);

### PopupMenud的使用？

（1）、用setOnMenuItemClickListener()方法注册PopupMenu

popup.setOnMenuItemClickListener(this);

（2）、给Activity或Fragment实现PopupMenu.OnMenuItemClickListener接口。

public class MainActivity extends ActionBarActivity implements OnMenuItemClickListener

ublic boolean onMenuItemClick(MenuItem item)

### Popup Menu和Context Menu的用户体现区别？

2015/9/6, vicky

Popup Menu：用户选择控件时，如轻轻点击时，弹出Popup Menu。

Context Menu：用户长按ListView Item时，会在上面出现Context Menu。经常用于ListView和GridView。

### 对话框的常见事件？

2015/9/6, vicky

AlertDialog

setNegativeButton：设置否。实现DialogInterface.OnClickListener接口。

setPositiveButton：设置是。实现DialogInterface.OnClickListener接口。

setNeutralButton：设置忽略。实现DialogInterface.OnClickListener接口。

### TimePickerDialog和DatePickerDialog的常见事件？

2015/9/6, vicky

TimePickerDialog.OnTimeSetListener：在调用TimePickerDialog构造函数中实现该接口。

DatePickerDialog.OnDateSetListener：在调用DatePickerDialog构造函数中实现该接口。

### 通知Notification如何定义通知的action？

2015/9/6, vicky

mBuilder.setContentIntent(pendingIntent);



### SufaceView与View的区别？

2015/9/6, vicky

SufaceView，继承与View。通常用于游戏。由于游戏中有大量的图片切换，使用二级缓存把图片组合好后，然后再显示出来。

SurfaceView和View最本质的区别在于：

surfaceView是在一个新起的单独线程中可以重新绘制画面。

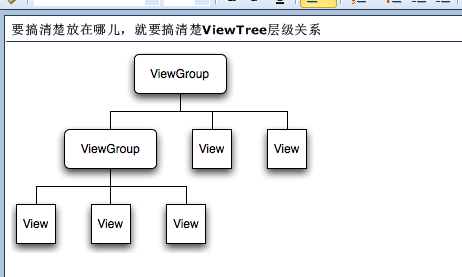
View必须在UI的主线程中更新画面。

根据游戏特点，一般分成两类。

1、被动更新画面的。使用View。

2、主动更新。使用surfaceView。例如，小人不断地在跑。

### 掌握View和ViewGroup类



视图树

ViewGroup：容器控件。

### NinePatch是什么的？为何会用到它？如何用？

为什么要用到9-Path图片？

Button自适应内容。横竖屏切换，手机尺寸不同，以及放大图片，图形会因为长宽的变化而产生拉伸,造成图形的失真变形。

在android平台下使用点九PNG技术，可以将图片横向和纵向同时进行拉伸，以实现在多分辨率下的完美显示效果。

Develop -> Tools， Tools Help 下Draw 9-patch。

sdk\tools \draw9patch.bat

画点是定义可伸缩区域。（show patches）

取任意一个长，画两个点。

取任意一个宽，画两个点。

画线是定义内容。

取任意一个长，画一条线。

取任意一个宽，画一条线。（showcontent）

保存的文件格式：.9.png

把图片放到Drawable下。

## Android数据存储

### 如何进行使用单元测试。

第1步、manifest文件中注册要测试的包和依赖包。

<instrumentation android:name="android.test.InstrumentationTestRunner" android:targetPackage="com.example.c11\_1\_datastorage\_sharedpreferences"/>

<uses-library android:name="android.test.runner"/>

第2步、定义重写继承AndroidTestCase类的类。

第3步、运行Android Junit Test

### getSharedPreferences是线程安全的吗？

2015/9/6, vicky

getSharedPreferences 是Context的单例模式方法，即读的同时不可以写，写的同时不可以读，这样保证信息统一。

写文件时，调用 edit()来得到一 个SharedPreferences.Editor，使用commit()或者apply()新数值。

### commit()与apply()的区别？

2015/9/6, vicky

其他提问方式：

使用SharedPreference提交比较多的数据编写时，使用commit()还是apply()?

commit是同步执行修改到磁盘文件。（主线程）

apply是异步执行修改到磁盘文件。（子线程）

### 三种方式得到preference区别？

2015/9/6, vicky

public SharedPreferences getPreferences (int mode)

通过Activity对象获取，获取的是这个Activity私有的Preference。

保存在系统中的xml形式的文件的名称为这个Activity的名字，因此一个Activity只能有一个，属于这个Activity。

public SharedPreferences getSharedPreferences (String name, int mode)

通过Activity对象获取，但是属于整个应用程序，可以有多个，以第一参数的name为文件名保存在系统中。

public static SharedPreferences getDefaultSharedPreferences (Context context)

PreferenceManager的静态函数，保存PreferenceActivity中的设置，属于整个应用程序，但是只有一个，Android会根据包名和PreferenceActivity的布局文件来起一个名字保存。

### SharedPreference的使用步骤？

2015/9/6, vicky

Shared Preferences操作分为3个步骤：得到 → 编辑 → 提交。

getSharedPreference

（写）get SharedPreferences.Editor

（写）Editor.commit()

MODE\_MULTI\_PROCESS（Constant Value: 4 (0x00000004)）：跨进程模式。

在Android 2.3及以前，这个标志位都是默认开启的，允许多个进程访问同一个 SharedPrecferences 对象。Android 2.3以后Android版本，必须通过明确的将 MODE\_MULTI\_PROCESS 这个值传递给mode参数，才能开启多进程访问。

在Android2.3之后不进行显式设置，若读与写在不同的线程，得到的是2个不同的SharedPreference，读写数据会不同步。

Content Providers：提供向其他程序提供访问数据甚至私有数据，访问提供权限限制。

跨进程访问时，使用Content Providers。

### 如何使用PreferenceActivity做设置界面？

2015/9/6, vicky

第1步：在 res/xml/preferences.xml文件夹创建preferences xml配置文件。

第2步：设置xml的内容。

使用PreferenceCategory分组。

使用PreferenceScreen子屏幕分组。

SwitchPreference：开关按钮，保存类型为boolean类型。

EditTextPreference 打开含EditText的对话框。保存类型为String。

CheckBoxPreference 以checkbox形式展示出来。保存类型为boolean类型。

ListPreference 打开含单选列表的对话框。保存类型为任何类型。

MultiSelectListPreference 打开含多选列表的对话框。保存类型为String Set。

RingtonePreference：设置铃声

第3步：关联xml和Activity。

### shared Preferenced文件保存位置？

2015/9/6, vicky

data/data/包名/shared\_prefs/文件名.xml。

### shared Preferenced取消自动保存？

2015/9/6, vicky

操作任何一个控件后立刻保存。

使用onPreferenceClick拦截，取消自动保存。

### shared Preferenced如何找到控件ID？

2015/9/6, vicky

id是自动生成的。根据key找到

### 文件的存储位置？

2015/9/6, vicky

data/data/包名/files/文件名.xml。

### External Storage保存数据时，是选择保存txt格式，还是二进制格式？

2015/9/6, vicky

不要保存为text文档，一般保存保存序列化对象为二进制文件。

游戏除了下载APP后外，一般还会下载一个数据包到sd卡，通常是二进制文件。

### Android 平台上的 SQLite 数据库的驱动是什么？

2015/9/7, vicky

SQLDroid 是 Android 平台上的 SQLite 数据库 JDBC 驱动程序。

String url = "jdbc:sqldroid:" + getDataPath() + "/main.sqlite";

### SQLite移植时，放在哪一层？

2015/9/7, vicky

跟Application Framework层相关。

跟Lib层有关。

跟底层驱动时无关。

### SQLite3支持这些常规的关系型数据库特性吗？

关系型数据库一般支持这些特性：

（1）、支持SQL

（2）、支持7种主外键关系，即多表查询。

（3）、支持事务

（4）、支持存储过程

（5）、支持视图

以上SQLite都支持。

### SQLIte的特性或者优点？

轻量级，仅占几百k的内存（赞）。

跨平台（赞）。

多语言接口（赞）。

独立性

隔离性

安全性

文本数据库是手机一个特性。

文本数据库的特点

不需要安装；

存储数据类型单一；

存储容量比较小；

数据是一个文件，可以直接查看。

### SQLite的数据库存储位置？

2015/9/7, vicky

SQLite database 存放位置：data/data/包/databases/下。

### 数据库常见的函数

实际使用中通过继承SQLiteOpenHelper类，封装数据库常见操作。这是Android提供的数据库管理类。

真正起作用的是SQLiteDatabase类

1、SQLiteOpenHelper类中常用方法

（1）、public abstract void onCreate(SQLiteDatabase db)

当数据库第一次被创建的时候调用。在这个函数中，创建表和初始化表。

（2）、public abstract void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion);

当数据库升级时被调用。在这个函数中，做更新数据库结构的事情。例如，增加表的列，删除表的列，重命名表名，删除表，新增表。

该方法内部按事务执行，因此，执行失败后，自动回滚。

onCreate()和onUpgrade ()方法是必须的。

（3）、public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion)

（4）、public SQLiteDatabase getWritableDatabase()

以写方式，打开或新建数据库。

（5）、public SQLiteDatabase getReadableDatabase()

以读方式，打开或新建数据库。

（6）、public String getDatabaseName()

获取数据库名称

（7）、public synchronized void close()

关闭数据库

（8）、public void onConfigure(SQLiteDatabase db)

配置数据库

### 数据的常见操作？

使用原则：

（1）、使用execSql，rawQuery、RawWithFactory执行原生的增删改查，代替Android insert，query，update，delete API。

RawWithFactory执行查询，效果会更高。但一般不用。

executeSQL：没有返回值。对应create 、 insert、 update、delete。

Query：有返回值。对应select。

（2）、查询结束后，关闭cursor。

（3）、操作结束后关闭数据库。

（4）、大量数据库操作时，新建子线程。下面的例子中没有添加子线程。

（5）、批量操作时使用事务提高处理速度和数据操作完整性。例如，批量插入时，sqlite插入数据的时候默认一条语句就是一个事务，有多少条数据就有多少次磁盘操作，重新打开和关闭数据库多少次。所以速度当然会很慢。而且不能保证所有数据都能同时插入。添加事务处理，把5000条插入作为一个事务。

增删改的统一操作：

db.executeSQL(String sql);

db.executeSQL(String sql, Object[] bindArgs);//sql语句中使用占位符，然后第二个参数是实际的参数集

查

db.rawQuery(String sql, String[] selectionArgs);

**public** **class** Person {

**public** **int** \_id;

**public** String name;

**public** **int** age;

**public** String info;

**public** Person() {

    }

**public** Person(String name, **int** age, String info) {

**this**.name = name;

**this**.age = age;

**this**.info = info;

    }

}

增

 //插入数据

 db.execSQL("INSERT INTO person VALUES (NULL, ?, ?)", **new** Object[]{person.name, person.age});

  /\*\*

     \* add persons

     \* @param persons

     \*/

**public** **void** add(List<Person> persons) {

        db.beginTransaction();  //开始事务

**try** {

**for** (Person person : persons) {

                db.execSQL("INSERT INTO person VALUES(null, ?, ?, ?)", **new** Object[]{person.name, person.age, person.info});

            }

            db.setTransactionSuccessful();  //设置事务成功完成

        } **finally** {

            db.endTransaction();    //结束事务

        }

    }

删

//删除数据

db.delete("person", "age < ?", **new** String[]{"35"});

改

//更新数据

 db.update("person", cv, "name = ?", **new** String[]{"john"});

查

db.rawQuery(String sql, String[] selectionArgs);

    /\*\*

     \* query all persons, return cursor

     \* @return  Cursor

     \*/

**public** Cursor queryTheCursor() {

        Cursor c = db.rawQuery("SELECT \* FROM person", **null**);

**return** c;

    }

/\*\*

     \* query all persons, return list

     \* @return List<Person>

     \*/

**public** List<Person> query() {

        ArrayList<Person> persons = **new** ArrayList<Person>();

        Cursor c = queryTheCursor();

**while** (c.moveToNext()) {

            Person person = **new** Person();

            person.\_id = c.getInt(c.getColumnIndex("\_id"));

            person.name = c.getString(c.getColumnIndex("name"));

            person.age = c.getInt(c.getColumnIndex("age"));

            person.info = c.getString(c.getColumnIndex("info"));

            persons.add(person);

        }

        c.close();

**return** persons;

    }

  Cursor c = db.rawQuery("SELECT \* FROM person WHERE age >= ?", **new** String[]{"33"});

**while** (c.moveToNext()) {

**int** \_id = c.getInt(c.getColumnIndex("\_id"));

            String name = c.getString(c.getColumnIndex("name"));

**int** age = c.getInt(c.getColumnIndex("age"));

            Log.i("db", "\_id=>" + \_id + ", name=>" + name + ", age=>" + age);

        }

        c.close();

### 什么数据都适合存放到数据库中吗？

2015/9/7, vicky

下面的几种情况就不适合放到数据库中：

（1）、图片等二进制数据：如果是图片的话，可以将文件名称或者路径保存到数据库中，真正的文件可以作为缓存文件保存在文件系统中。

（2）、临时数据：定位获取到的Location，登录的Session等。

（3）、日志数据：可以写入文件中，通常是log\_xxxx.txt。

### 了解文件存储的内外存储分别

2015/9/7, vicky

外置SD卡设计到Mount、UNMount状态、只读、只写状态。其他操作和内部存储一致。

### 序列化对象的方法？

2015/9/5, vicky

其他提问方式：

实现序列化的方法？

使用Parcelable接口（Android）或Serializable（JavaSE）接口序列化对象。

### 为什么要序列化对象？

2015/9/5, vicky

把对象序列化，变成二进制形式，使得传输速度更快。

（1）、永久性保存对象，保存对象的字节序列到本地文件中。

①网站做缓存（Cache）的时候，必须序列化。

②游戏随时保存状态。

③数据库持久层框架ORM，都要求数据库表对应的实体类是可序列化的。框架底层好传输和存储。

（2）、通过序列化对象在网络中传递对象。

为了传输方便，将要传输的对象序列化为二进制的数据流，效率极高，接收时通过反序列化转化成对象，从而达到一个传输的效果

（3）、通过序列化在进程间传递对象。

### Android中Intent传递对象有两种方法：

这些Object是有一定的条件的，前者是实现了Serializable接口，而后者是实现了Parcelable接口。

使用两种方式传递的语法分别为：

bundle.putSerializable(key,object);

bundle.putParcelable(key,object);

使用两种方式接收的语法分别为：

object=(Object) getIntent().getSerializableExtra(key);

object=(Object) getIntent().getParcelableExtra(key);

### 选择序列化方法的原则

2015/9/5, vicky

（1）、使用内存的时候，Parcelable比Serializable性能高，所以推荐使用Parcelable。

使用Parcelable的场景：同一进程内使用Intent传输数据，进程间使用IBinder传输数据。

Serializable在序列化的时候会产生大量的临时变量，从而引起频繁的GC。

（2）、数据持久化（文件存储、网络传输、数据库存储、SharedPreference存储）时，使用Serializable。

因为Parcelable在外界有变化的情况下，Parcelable不能很好的保证数据的持续性。其次，android不同版本Parcelable可能不同。

### 使用Serializable的步骤？

2015/9/5, vicky

Serializable的实现，只需要implements Serializable 即可。这只是给对象打了一个标记，系统会自动将其序列化。

### 实现Parcelable步骤

2015/9/5, vicky

[复制代码](javascript:void(0);)

public interface Parcelable

{

//内容描述接口，基本不用管

public int describeContents();

//写入接口函数，打包

public void writeToParcel(Parcel dest, int flags);

//读取接口，目的是要从Parcel中构造一个实现了Parcelable的类的实例处理。因为实现类在这里还是不可知的，所以需要用到模板的方式，继承类名通过模板参数传入

//为了能够实现模板参数的传入，这里定义Creator嵌入接口,内含两个接口函数分别返回单个和多个继承类实例

public interface Creator<T>

{

public T createFromParcel(Parcel source);

public T[] newArray(int size);

}

}

[复制代码](javascript:void(0);)

（1）、implements Parcelable

（2）、重写writeToParcel方法。将对象映射成Parcel对象。

（3）、重写describeContents方法，内容接口描述，默认返回0就可以

（4）、实例化静态内部对象CREATOR实现接口Parcelable.Creator

public static final Parcelable.Creator<T> CREATOR

其中public static final一个都不能少，内部对象CREATOR的名称也不能改变，必须全部大写。

CREATOR中createFromParcel方法将Parcel对象映射成对象。

两者代码比较：

1）创建Person类，实现Serializable

[复制代码](javascript:void(0);)

public class Person implements Serializable

{

private static final long serialVersionUID = -7060210544600464481L;

private String name;

private int age;

public String getName()

{

return name;

}

public void setName(String name)

{

this.name = name;

}

public int getAge()

{

return age;

}

public void setAge(int age)

{

this.age = age;

}

}

[复制代码](javascript:void(0);)

2）创建Book类，实现Parcelable

[复制代码](javascript:void(0);)

public class Book implements Parcelable

{

private String bookName;

private String author;

private int publishDate;

public Book()

{

}

public String getBookName()

{

return bookName;

}

public void setBookName(String bookName)

{

this.bookName = bookName;

}

public String getAuthor()

{

return author;

}

public void setAuthor(String author)

{

this.author = author;

}

public int getPublishDate()

{

return publishDate;

}

public void setPublishDate(int publishDate)

{

this.publishDate = publishDate;

}

@Override

public int describeContents()

{

return 0;

}

@Override

public void writeToParcel(Parcel out, int flags)

{

out.writeString(bookName);

out.writeString(author);

out.writeInt(publishDate);

}

public static final Parcelable.Creator<Book> CREATOR = new Creator<Book>()

{

@Override

public Book[] newArray(int size)

{

return new Book[size];

}

@Override

public Book createFromParcel(Parcel in)

{

return new Book(in);

}

};

public Book(Parcel in)

{

bookName = in.readString();

author = in.readString();

publishDate = in.readInt();

}

}

[复制代码](javascript:void(0);)

## 多媒体

### Android动画分类及编写方式？

API Guides → Animation and Graphics

2015/9/7, vicky

1、Property Animation（Need Ver3.0）

（1）、ViewAnimation

（2）、ObjectAnimation

2、VIew Animation，又称为TweenAnimation。

渐变动画（AlphaAnimation）

旋转动画（RotateAnimation）

缩放动画（ScaleAnimation）

位移动画（TranslateAnimation）

动画集合AnimationSet（上面四种的结合）

3、Drawable Animation，有成为FrameAnimation

### 如何创建一个VIew Animation

第1步、创建动画。在xml中定义或通过代码创建。

第2步、为View指定动画.

第3步、监控动画的执行状态。通过为动画设定AnimationListener。

**//创建一个AnimationSet对象，参数为Boolean型，**

**//true表示使用Animation的interpolator，false则是使用自己的**

           AnimationSet animationSet = **new** AnimationSet(**true**);

**//创建一个AlphaAnimation对象，参数从完全的透明度，到完全的不透明**

           AlphaAnimation alphaAnimation = **new** AlphaAnimation(1, 0);

**//设置动画执行的时间**

           alphaAnimation.setDuration(500);

**//将alphaAnimation对象添加到AnimationSet当中**

           animationSet.addAnimation(alphaAnimation);

**//使用ImageView的startAnimation方法执行动画**

           image.startAnimation(animationSet);

### 如何创建一个Drawable Animation

同VIew Animation。

下面就给个具体的XML例子，来定义一帧一帧的动画：

<animation-list xmlns:android="http://schemas.android.com/apk/res/android"

    android:oneshot="true">

    <item android:drawable="@drawable/rocket\_thrust1" android:duration="200" />

    <item android:drawable="@drawable/rocket\_thrust2" android:duration="200" />

    <item android:drawable="@drawable/rocket\_thrust3" android:duration="200" />

</animation-list>

然后我们将以上XML保存在**res/anim/文**件夹下，命名为rocket\_thrust.xml，显示动画的代码：

AnimationDrawable rocketAnimation;

public void onCreate(Bundle savedInstanceState) {

  super.onCreate(savedInstanceState);

  setContentView(R.layout.main);

  ImageView rocketImage = (ImageView) findViewById(R.id.rocket\_image);

  rocketImage**.setBackgroundResource**(R.anim.rocket\_thrust);

  rocketAnimation = (AnimationDrawable) **rocketImage.getBackground**();

}

public boolean onTouchEvent(MotionEvent event) {

  if (event.getAction() == MotionEvent.ACTION\_DOWN) {

**rocketAnimation.start();**

    return true;

  }

  return super.onTouchEvent(event);

}

### 动画实质是什么 ？

平面和3D靠算法与矩阵实现的。

矩阵（Matrix）。

2D平面动画是3 \* 3矩阵。

3D动画是3 \* 3，甚至是4 \* 4矩阵。

### Android动画实现机制

通过Animation定义动画参数，然后让Transformation去进行运算，运算完了再由View去呈现。

### 动画相关的常见函数

|  |  |  |
| --- | --- | --- |
| ttribute Name | Related Method | Description |
| [android:detachWallpaper](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html#attr_android:detachWallpaper)是否允许监听事件 | [setDetachWallpaper(boolean)](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html#setDetachWallpaper%28boolean%29) | Special option for window animations: if this window is on top of a wallpaper, don't animate the wallpaper with it. |
| [android:duration](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html#attr_android:duration)  设置运行时间 | [setDuration(long)](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html#setDuration%28long%29) | Amount of time (in milliseconds) for the animation to run. |
| [android:fillAfter](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html#attr_android:fillAfter)  动画结束后停在后面 | [setFillAfter(boolean)](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html#setFillAfter%28boolean%29) | When set to true, the animation transformation is applied after the animation is over. |
| [android:fillBefore](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html#attr_android:fillBefore)  动画结束后停在开始 | [setFillBefore(boolean)](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html#setFillBefore%28boolean%29) | When set to true or when fillEnabled is not set to true, the animation transformation is applied before the animation has started. |
| [android:fillEnabled](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html#attr_android:fillEnabled) | [setFillEnabled(boolean)](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html#setFillEnabled%28boolean%29) | When set to true, the value of fillBefore is taken into account. |
| [android:interpolator](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html#attr_android:interpolator) | [setInterpolator(Interpolator)](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html#setInterpolator%28android.view.animation.Interpolator%29) | Defines the interpolator used to smooth the animation movement in time. |
| [android:repeatCount](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html#attr_android:repeatCount)  重复次数 | [setRepeatCount(int)](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html#setRepeatCount%28int%29) | Defines how many times the animation should repeat. |
| [android:repeatMode](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html#attr_android:repeatMode)  重复模式 | [setRepeatMode(int)](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html#setRepeatMode%28int%29) | Defines the animation behavior when it reaches the end and the repeat count is greater than 0 or infinite. |
| [android:startOffset](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html#attr_android:startOffset)  延迟启动的时间 | [setStartOffset(long)](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html#setStartOffset%28long%29) | Delay in milliseconds before the animation runs, once start time is reached. |
| [android:zAdjustment](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html#attr_android:zAdjustment)  设置运行时的Z方向  （覆盖别的图片） | [setZAdjustment(int)](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html#setZAdjustment%28int%29) | Allows for an adjustment of the Z ordering of the content being animated for the duration of the animation. |

|  |  |
| --- | --- |
| void | [setAnimationListener](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html#setAnimationListener%28android.view.animation.Animation.AnimationListener%29)([Animation.AnimationListener](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.AnimationListener.html) listener)  设置事件监听  Binds an animation listener to this animation. |

|  |  |  |
| --- | --- | --- |
| **Public Methods** | | |
| abstract void | [onAnimationEnd](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.AnimationListener.html#onAnimationEnd%28android.view.animation.Animation%29)([Animation](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html) animation)  Notifies the end of the animation. |
| abstract void | [onAnimationRepeat](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.AnimationListener.html#onAnimationRepeat%28android.view.animation.Animation%29)([Animation](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html) animation)  Notifies the repetition of the animation. |
| abstract void | [onAnimationStart](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.AnimationListener.html#onAnimationStart%28android.view.animation.Animation%29)([Animation](file:///D:\Program%20Files\Android\sdk\docs\reference\android\view\animation\Animation.html) animation)  Notifies the start of the animation. |

## Android应用程序签名及发布

### Android应用程序发布时，为什么要签名？

Created by viky,2015/8/28

应用程序签名主要用于对开发者身份进行识别，达到防范恶意攻击的目的，但不能有效地限制应用程序被恶意修改，只能够检测应用程序是否被修改过，如果应用程序被修改应该再采取相应的应对措施。

签名涉及到三个方面：商业应用、版本声明、版本发布。

（1）、在Debug模式编译时，ADT根据会自动用默认的密钥和证书来进行签名。在Release模式编译时，ADT不会自动为apk签名，这需要手工签名。

（2）、版本发布：

如果没有进行签名，盗版后可以被他人所有。

签名后会被加密，当签名被更改后，market不会承认这个被修改过的盗版程序。

（3）、版本声明与程序安装：

没有签名的程序是不受信任的，安装时提示未知程序。

如果手机在【设置】功能中关闭了“允许程序安装来未知来源的应用程序”，那么，没有签名的程序不能安装。

（4）、程序升级：版本升级时，新版本必须使用同一个证书签名。

只有当签名证书相同、包名相同，才能升级。

如果签名证书不同，系统会要新版本使用不同的包名。此时吗，必须先卸载旧版本，才能安装新的版本。

（5）、代码共享或者数据共享：

Android提供了基于签名的权限机制。以同一个证书对多个应用程序进行签名，利用基于签名的权限检查，应用程序间以安全的方式共享代码或共享数据。

### Android应用程序签名的详细步骤？

Created by viky,2015/8/28

apk的签名工作可以通过两种方式来完成：

第1种方式：通过ADT提供的图形化界面完成apk签名；

项目的右键菜单 → Android Tools → Export Signed Application Package → 打开图形界面，根据提示填写信息，就完成签名了。

第2种方式：完全通过shell/dos命令来完成apk签名。

签名时，依次用到keytool.exe → jarsigner.exe → zipalign三个工具。

其中。Zipalign是可选的，keytool和jarsigner两个工具是jdk自带的，jarsigner工具主要是用来给jar文件签名的。

（1）、Keytool：生成数字证书，即密钥，也就是.keystore文件。

（2）、Jarsigner：使用Keytool生成的.keystore文件，给apk jar包签名。

（3）、zipalign：对签名后的apk进行优化，提高与Android系统交互的效率。Android SDK1.6版本开始包含此工具。

如果第一次做Android应用程序签名，上面的3个工具都将用到；但如果已经有数字证书了，以后再给其它apk签名时，只需要用到jarsigner和zipalign就可以完成。

具体如何使用keytool、jarsigner 、zipalign命令，通过查看help命令，或以前记录的笔记。

Android只是在应用程序安装的时候才会检查证书的有效期、有效性。

签名机制标明了APK的发行机构。因此，站在软件安全的角度，通过比对APK的签名情况，判断此APK是否由“官方”发行，而不是被破解篡改过重新签名打包的“盗版软件”。

数字证书是有有效期的。如果程序已经安装在系统中，即使证书过期也不会影响程序的正常功能。

### Android应用程序的版本升级是通过覆盖安装实现的。程序覆盖安装时系统如何做出合法性检查？

Created by viky,2015/8/28

程序覆盖安装主要检查两点：

（1）、两个程序的入口Activity是否相同。两个程序如果包名不一样，即使其它所有代码完全一样，也不会被视为同一个程序的不同版本；

（2）、两个程序所采用的签名是否相同。如果两个程序所采用的签名不同，即使包名相同，也不会被视为同一个程序的不同版本，不能覆盖安装。签名证书不同，系统会要新版本使用不同的包名。此时，必须先卸载旧版本，才能安装新的版本。

### Android应用程序的签名制是什么？（略）

Created by viky,2015/8/28

Jarsigner.exe：使用Keytool生成的.keystore文件，给apk jar包签名。它使用的是signapk.jar 工具，并一个META-INF/文件夹。META-INF中包含了MANIFEST.MF, CERT.SF, CERT.RSA。

signapk.jar是Android源码包中的一个签名工具。由于Android是个开源项目，所以，很高兴地，我们可以直接找到signapk.jar的源码！路径为/build/tools/signapk/SignApk.java。

### Android应用程序发布时使用的是Release版本。Release和Debug版本有什么区别吗？

Created by viky,2015/8/28

软件开发模式：Debug和release模式。这是一个通用概念。

Release模式会删除一个日志log、测试测试、测试脚本等，使得程序更轻巧、更稳定，没有垃圾数据。

### Android应用程序发布时，如何进行版本发布？

Created by viky,2015/8/28

一、准备发布：

第1步：移除log，并移除Android:debugable属性（旧版本中含有），设置版本编号和名称。

第2步：签名。通过ADT工具叫keytool。

第3步：测试发布版本。

第4步：更新应用程序的发布资源。（有时不需要）

商业产品有图片资源有水印标识，在发布前使用工具移除水印标记。如果被修改，程序可能不能通过。这样防止代码泄露。

第5步：准备好应用程序依赖的远程服务器。

### Android APK发布到Marcket上，有什么要求？

Created by viky,2015/8/28

包名必须不同。

签名必须不同。

### Android应用程序发布时，通常有哪些发布方式？或版本控制

Created by viky,2015/8/28

第1种方式：直接APK传播。

可以放在官网或者企业appstore上，通过浏览器下载安装。

放在第三方应用程序市场。例如，360，腾讯应用宝，百度应用，豌豆荚等。

第2种方式：自动版本升级功能。

方法一、通过接口协议得到下载地址。然后使用Android系统提供Intent 和Uri调用ContentProvider-Browser来下载。

方法二、通过接口协议得到下载地址。然后使用自己写的文件下载功能进行下载。

方法三、直接放到Android Market。Market中有一套自己的管理机制。我们现在用的是公司的app store和第三方的appp store。

### Android不同应用程序之间，如何共享数据或共享代码？

Created by viky,2015/8/28

方式1：使用同一个用户ID使得不同应用程序运行在同一个进程中。

方式2：用相同的证书签名共享数据或代码。

方式3：使用ContentProvider共享数据。这是一种方法最常用。

### 在Debug模式编译时，ADT根据会自动用默认的密钥和证书来进行签名。那么，Debug签名有什么限制或风险？

Created by viky,2015/8/28

（1）、debug签名的应用程序不能在Android Market上架销售，它会强制你使用自己的签名；

（2）、debug.keystore（也就是数字签名）在不同的机器上所生成的可能都不一样，如果你换了机器进行apk版本升级，那么将会出现程序不能覆盖安装的问题。只能通过先卸载原先程序，然后再安装新程序。但如果软件有很多使用客户，这样软件不具备升级功能的。