**Android基础笔记**

# **android简介**

**\* Android版本进化历程**

**1.X 版本：入门级别，只适用于手机**

**2.X 版本: 趋于成熟的版本，只适用于手机**

**3.X 版本：趋于成熟的版本，只适用于平板**

**4.X 版本：趋于成熟的版本，对平板和手机进行了整合**

**5.X 版本: 支持64位cpu、默认采用ART运行模式等**

**Android 1.5 Cupcake(纸杯蛋糕、API Level 3)**

**Android 1.6 Donut(甜甜圈API Level 4)**

**Android 2.0/2.0.1/2.1 Eclair(松饼API Level 5、6、7)**

**Android 2.2/2.2.1 Froyo(冻酸奶API Level 8、9)**

**Android 2.3 Gingerbread(姜饼API Level 10)**

**Android 3.0/3.1/3.2 Honeycomb(蜂巢 API Level 11、12、13)**

**Android 4.0/4.0.3 Ice Cream Sandwich(冰激凌三明治API Level 14、15)**

**Android 4.1/4.2/4.3 Jelly Bean(果冻豆 API Level 16、17、18)**

**Android 4.4 KitKat (奇巧巧克力API Level 19)**

**Android 5.0 代号L(棒棒糖) (API Level 20)**

**Apilevel: 19 —— 4.4版本**

**18—— 4.3**

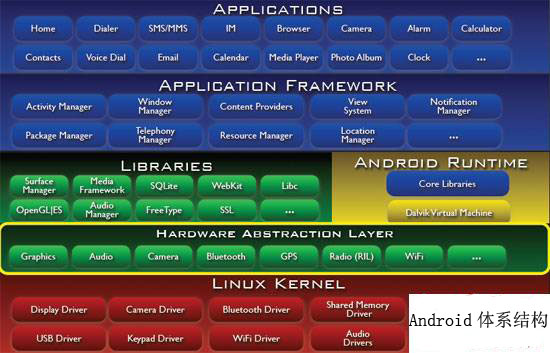
**11——3.0**

**10——2.3**

**8——2.2**

**Android体系结构**

Android体系结构

****

**1 Linux内核层 liunx kernel**

**Android是基于Linux内核的操作系统，在Linux内核层，主要实现安全管理、进程管理、内存管理、电源管理、硬件驱动管理**

**2硬件抽象层 Hardware Abstraction Layer**

**硬件抽象层是对Linux 硬件驱动程序的封装，向上提供接口，屏蔽低层的实现细节。硬件抽象层不开源，可保护硬件厂商的商业秘密，但是会影响系统的性能。**

**Android对硬件的支持分成了两层，一层放在用户空间（User Space），一层放在内核空间（Kernel Space），硬件抽象层运行在 用户空间， Linux 内核驱动程序运行在内核空间，内核驱动层只提供简单的访问 硬件逻辑，具体的实现细节，都放在硬件抽象层中 ，从而维护了硬件厂商的商业利益。**

**3原生库与Android运行时 library + android runtime**

**Android的原生库主要基于C\C++实现的一些原生组件，包括C库Bionic、浏览器引擎Webkit、多媒体引擎OpenCORE、SQL数据库SQLite、3D渲染引擎OpenGL ES、位图和字体矢量渲染引擎FreeType、2D图像渲染引擎SGL（Skia Graphics Library）、互联网安全协议SSL等。**

**Android运行时主要包括Java核心库、Dalvik虚拟机，两者一起构成了Android的应用环境基础**

**4应用框架层 application framework**

**应用框架层是Google发布的核心应用所使用的API框架，开发人员可以使用这些框架提供的API来快速开发自己的应用程序，它本身也是用Java语言实现和开发的。**

**Activity Manager(活动管理)、Window Manager(窗口管理)、View Manager(视图管理)、Notification Manager(通知管理)、Content Provider(内容提供者)、Package Manager(包管理)、Resource Manager(资源管理)、Location Manager(位置管理)**

**5应用层 applications**

**应用层是基于Android平台开发的应用，采用Java作为开发语言，包括系统自带的应用(短信管理、联系人管理、图库浏览、网页浏览等)和第三方开发的应用**

**android是google公司对外发布的一个免费开源基于Linux2.6的操作系统。**

**2、android是一个层叠式的操作系统，分为四层**

**第一层 applications 应用层 应用程序运行在这一层**

**写出Andriod核心应用程序：**

**home contacts phone browser**

**第二层 application framework 应用框架层**

**第二层与第三层之间 有 JNI(java native interface) 用于 c+ 与java 之间相互调用**

**第三层 library + android runtime 库+android运行时环境**

**library:是由c/c++编写的免费开源的库**

**android runtime:core library + DalVik vm(android每个应用程序都有自己单独的dalvik vm )**

**第四层 liunx kernel linux内核 （驱动 和硬件相关 driver）**

**Binder Driver**

**Binder Driver(IPC):原来linux系统没有，是由google工程师加入。**

**IPC:进程间通讯。**

## **JVM与DVM和ART**

**\* 共同点：**

**解释执行 byte code字节码文件**

**在操作系统的进程运行一个VM，并执行一个单独的程序**

**\* 不同点：**

**程序结构不同：JVM字节码由多个.class文件组成，Davlik 只包含一个 .dex 格式的文件，这个文件包含了程序中所有的类。**

**架构不同：Dalvik是 register-based基于寄存器，Sun JDK 是 stack-based基于栈。**

**\* Dalvik优势**

**编译时提前优化代码而不是等到运行时 。**

**虚拟机很小,占空间小。可以满足可高效运行多种虚拟机实例。**

**常量池修改为只使用32位的索引，以简化解释器。**

**\* DVM执行过程**

**1. 编译为class文件**

**2. 使用dx工具抽取class文件共性形成.dex**

**3. 维护内部常量池**

**dvm和art虚拟机：**

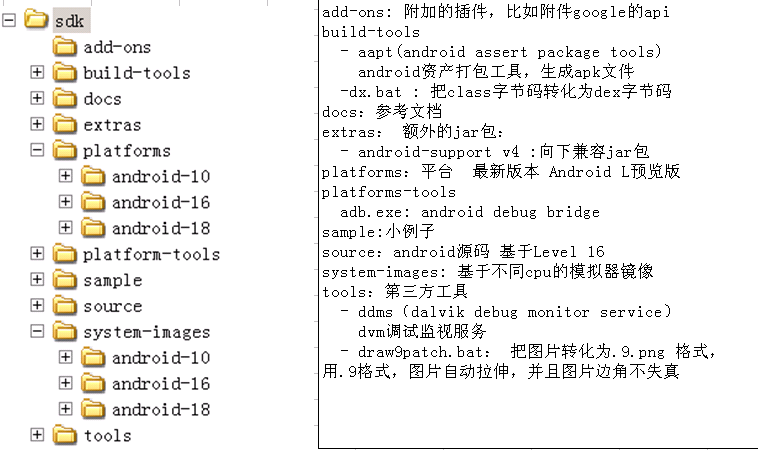
**ART模式是Android runtime的简称，在Android4.4版本后出现，**

**通过在安装应用程序时，自动对程序进行代码预读取编译，让程序直接编译成机器语言，**

**并且把它保存起来，免去了Dalvik模式要时时转换代码，**

**实现高效率、省电、手机运行流畅。ART是虚拟机，只是在安装apk时，提前编译而已。**

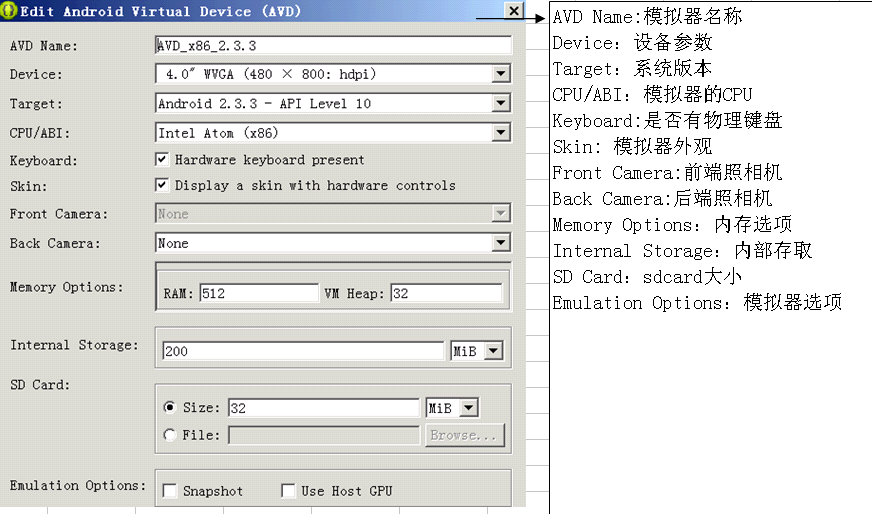
## **Android SDK目录介绍**

****

* **add-ons 这里面保存着附加库，比如GoogleMaps**
* **build-tools 编译需要用到的工具**
* **docs 开发的离线文档**
* **extras 额外工具，模拟器加速器，向下兼容的jar包**
* **platforms 开发需要的平台，每个平台的android.jar在对应的文件夹里面**
* **platform-tools 开发调试的工具 adb.exe 一个重要的调试桥**
* **sources android.jar的源码**
* **system-images 模拟器创建需要的镜像**
* **tools 开发辅助需要用到的工具，模拟器.exe**

**创建AVD模拟器**

**\* 创建模拟器**

****

**小细节**

**& AVD配置文件位置(默认)：**

**C:\Documents andSettings\Administrator\.android\avd\AVD2.3.3.avd\config.ini**

**& AVD 缩放 ：启动模拟器—>在launch Option中选择 Scale display to real size**

**& 模拟器不能保存数据：在模拟器的配置文件中删除.lock文件**

**手机参数**

**\* 手机屏幕参数：屏幕尺寸、分辨率、屏幕密度**

**&手机尺寸：5.5 、5.0、4.7、4.3、4.0、3.5英寸**

**&分辨率: 1080\*1920 、640\*1136、720\*1280、1280\*800、1024\*600、480\*854、480\*800**

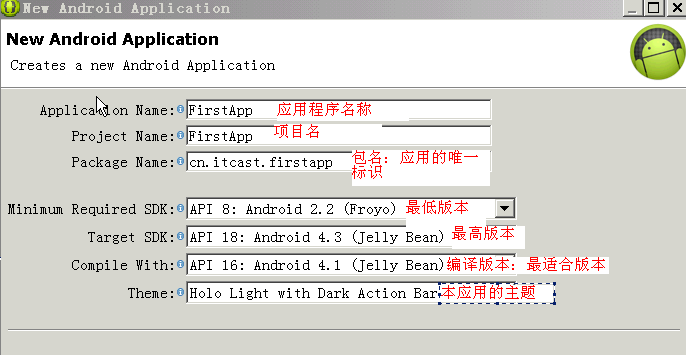
**&屏幕密度：120dpi(ldpi) 低清、160dpi(mdip)中清(标清) 、240dpi(hdpi)、360dpi(xhdpi)**

**\* 手机内存、SDCard、CPU、摄像头**

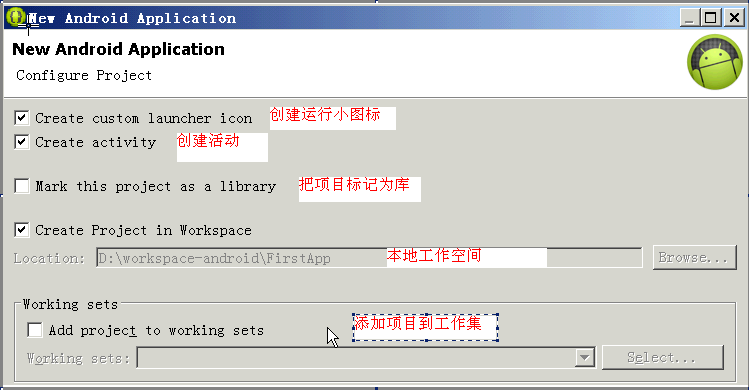
**\* 手机操作系统版本**

**创建应用程序流程**

**\* 创建新项目**

****

**配置项目**

****

**配置运行图标属性**

****

## **Android应用程序结构介绍**

**Android 应用程序由应用组件(Activity、Service、ContentProvider、BroadcastReceiver)构成，每一个组件是单独的模块，有着不同的入口**

**android框架依据这些入口进入到该应用中， 组件在AndroidManifest.xml注册，它们之间松耦合地组织在一起，从而定义应用的整体功能。**

## Android项目目录结构分析



## **Android应用 运行流程分析**

**\* 编译打包生成apk**

**Jdk dx.bat(Android打包程序生成.dex文件) aapt(Android应用程序打包工具)**

**java--class--classes.dex(dx.bat)-->apk(aapt )-->apk(签名，默认debug签名)**

**\* 通过adb（android debug bridge：android 调试桥）把 apk上传到 模拟器或者真机中，位置：/data/app/目录**

**\* 再解析AndroidManifest.xml文件，获取包名，版本信息，组件信息，权限信息，支持的最低版本与最高版本 ,再创建文件夹/data/data/<当前的package> ，用于存放该应用的私有数据**

**\* 把包等信息写入到/data/system/packages.xml 文件中，实现注册**

**⮚ 安装APK**

**运用adb工具把apk包上传到模拟器（真机）中，目录在data\app\**

**读取androidManifest.xml清单文件（包名、权限、图标等）**

**创建文件夹，在data\data\目录下创建以包名为文件夹的目录**

**写入注册表, 注册表文件data\system\packages.xml**

**⮚ 启动APK**

**Android系统是一个多用户的Linux系统，应用一旦安装，系统为每个应用分配一个独立的Linux用户ID，当用户点击应用程序图标或者导航到该应用的组件时，Android框架会创建一个虚拟机实例，开启一个进程，创建一个主线程(UI线程)，再实例化入口组件，进入组件的生命周期**

**ADB常用的命令操作**

**adb ：android 调试桥，它是windows与android系统的桥梁，通过它可以安装、卸载 程序等**

**ADB 主要作用：建立手机与电脑的联接**

**在命令行： 但是要设置环境变量 例： path=E:\Android\adt-bundle-x86-32\sdk\platform-tools**

**adb shell ：进入linux客户端shell命令模式**

**adb install apk文件 ：安装apk文件**

**adb uninstall 包名 ：删除模拟器应用**

**adb devices :列出所有设备**

**adb push <local> <remote> 把本地的文件复制到远端**

**adb pull <remote> <local> 从远端复制文件到本地**

**mksdcard 大小 <local> 创建sdcard镜像文件**

**android create avd –name <模拟器名> -target <level> 创建模拟器**

**emulator –avd avd名 启动模拟器**

**adb –s <设备名> <命令> 对于有多个设备终端**

**提示不是内部或外部命令的原因是：cmd当前操作不在系统文件夹system32下，那么只需输入：cd c:\WINDOWS\system32\  ；即可将当前操作路径切换到Windows 操作系统的系统文件夹下。然后再输入netstat -an，即可解决。**

**ADB 挂掉的三种解决方法**

* **利用eclipse重启adb：DDMS Devices 点击向下的小箭头 Reset adb**
* **使用adb指令：adb kill-server 杀掉adb调试桥，然后运用adb start-server或adb devices 重启adb，如果不管用，找到任务管理器 干掉adb的进程 然后重启Eclipse**
* **干掉占用了5037端口号的其他应用程序 ： dos 命令行 输入 netstat -aon|findstr “5037” 找到监听这个端口号的PID 在任务管理器干掉对应的程序**

**DDMS透视图**

**DDMS(Dalvik Debug Monitor Service) DVM调试监视服务**

**& Devices : 查看模拟器设备的状态,模拟器运行的进程**

**& File Explorer: 文件浏览器(导入、导出文件、创建文件夹、删除文件等)**

**& LogCat: 日志查看工具,日志猫**

**& Emulator Control: 模拟器控制，模拟一些操作(打电话、发短信)**

# Android中常用的布局

## LinearLayout (线性布局)

**线性布局，它的子控件是以单一的行或者单一的列排列，子控件不能重叠，具有方向性（水平、垂直），默认是水平方向，可以设置位置和权重**

**EditText文本输入框 ImageView 图片显示 TextView 文本显示 Botton 按钮 checkBox 复选框 ScrollView 滚动的View**

**ProgressBar 是一个进度条控件，用于显示一项任务的完成进度。 ListView 是一个控件,一个在垂直滚动的列表中显示条目的一个控件，**

**RadioButton单选按钮必须放在RadioGroup组中　组中默认的方向是垂直的**

**View控制下面的属性所有子控件都包含：padding Margin**

**\* 常用的XML属性：**

**width 宽 height 高 orientation：方向 vertical 垂直 center 居中显示**

**match\_parent 填充父元素（2.0出现）**

**fill\_parent 填充父元素 父类有多大空间，就占多大空间**

**wrap\_content 包裹内容 有多大空间 就占多大空间**

**android:background=""背景颜色**

**android:inputType 样式是否是明文显示**

**password="true" 密码是否显示**

**scaleType 拉伸图片到控件大小**

**android:****textSize=*"10sp" t*extSize控制文字大小 单位用sp 宽 高 单位使用 dp(设备的象素)**

**android:orientation 方向 ="vertical" 垂直 horizontal 水平 hint：提示**

**android:minLines=*"4" 最小行数***

**android:signleline=true 只显示一行**

**android:gravity="center" gravity 子控件相对于当前控件对齐方试**

**orientation:方向；vertical，垂直；horizontal,水平**

**layout\_gravity：当前控件相对于父控件的对齐方式**

**margin:当前控件相对于四周的间距 外边距**

**padding：当前控件中的子控件相对于当前控件四周的间距 内边距**

**android:layout\_weight 权重**

**\* 注意点：**

**- 修饰控件属性时，有layout与无layout的区别**

**有layout，表示布局属性修饰，它相对于父控件或者父控件中的其他子控件的属性修饰，**

**无layout，它只针对控件本身，是对该控件内部元素的修饰**

**- 权重**

**它是线性布局或者它的子类所特有的属性修饰，它针对的是子控件的宽和高的设置**

**先预留出没有设置权重的控件的宽或者高，之后对相对于父控件剩余的空间(宽或者高)按权重(比例)分配**

**获取控件中子控件的数量**

**int count = liearlayout.getChildCount();**

**通过ID获取控件**

**Button button = (Button) liearlayout.getChildAt(i);**

## FrameLayout(**帧布局**)

**帧布局被设计成在一个屏幕区域显示一个单一的项(single item)。层层叠加**

**通常FrameLayout显示一个单一的子控件，它支持的布局属性不够丰富，一般通过layout\_gravity来设置子控件的位置。**

**FrameLayout的子控件被绘制在一个堆栈中，最近添加进来的子控件在堆栈的顶部**

* **TableLayout**

**表格布局，是LinearLayout的子类，以行和列的形式存放子控件，它通常由多个TableRow布局控件组成，TableRow由多个单元格(cell)组成，**

**每个cell设置为View对象 ，表格布局的子控件可以设置权重，但是不能设置方向**

* **GridLayout**

**网格布局，android 4.0以后出现，布局使用虚细线将布局划分为行、列和单元格，也支持控件在行、列上都有交错排列。有方向性，默认为水平方向，相对于TableLayout，渲染速度更快，更灵活。**

**对于低版本要使用GridLayout，需添加向下的支持库v7**

**xmlns:app="http://schemas.android.com/apk/res-auto"**

**\* 常用的XML属性：**

**android:columnCount 设置列数**

**android:layout\_row 控件的起始行位置**

**android:layout\_column 控件的起始列位置**

**android:layout\_rowSpan 控件跨越行数**

**layout\_columnSpan 控件跨越列数**

**layout\_gravity=“fill” 与layout\_rowSpan或columnSpan结合使用,表示填满所跨越的行或者列**

* **AbsoluteLayout 不建议使用**

**绝对布局，子控件的位置以绝对的位置定位，子控件之间可以重叠，相对于其他布局，缺少灵活性，不建议使用**

**\* 常用的XML属性：**

**android:layout\_x : 相对于父控件的x坐标位置**

**android:layout\_y : 相对于父控件的y坐标位置**

## RelativeLayout(相对布局)

**相对布局，子控件的位置关系可以通过子控件与父控件、子控件与子控件来确定，**

**子控件之间位置可以重叠，拓展性好，灵活方便，是使用最多的布局方式**

**都是从左上角开始布局 要控制位置需要设置每个控件相对于其它控件的位置**

**\* 常用的XML属性**

**android:layout\_toLeftOf="@id/name" 指定控件的左边**

**android:layout\_toRightOf="@id/name" 指定控件的右边**

**android:layout\_above="@id/name" 指定控件的上边**

**android:layout\_below="@id/name" 指定控件的下边**

**android:layout\_alignLeft="@+id/name" 与指定控件左对齐**

**android:layout\_alignRight="@+id/name" 与指定控件右对齐**

**android:layout\_alignTop="@+id/name" 与指定控件顶部对齐**

**android:layout\_alignBottom="@+id/name" 与指定控件底部对齐**

**android:layout\_alignParentLeft="true" 与父控件的左边对齐**

**android:layout\_alignParentRight="true" 与父控件的右边对齐**

**android:layout\_alignParentTop="true" 与父控件顶部对齐**

**android:layout\_alignParentBottom="true" 与父控件底部对齐**

**android:layout\_centerHorizontal="true" 在父控件中水平居中**

**android:layout\_centerVertical="true" 在父控件中垂直居中**

**android\_layout\_centerInParent="true" 在父控件中中部居中**

**layout\_centerHorizontal（相当于父节点是水平居中）**

**layout\_centerVertical （相当于父节点是垂直居中)**

**layout\_centerInParent（相当于父节点在正中间）**

**layout\_above （在XX之上）**

**layout\_below （在XX之下）**

**layout\_toLeft （在XX的左边）**

**layout\_toRight（在XX的右边）**

**layout\_alignParentTop（和父节点的顶部对齐）**

**layout\_alignParentBottom （和父节点的底部对齐）**

**layout\_alignParentLeft （和父节点的左边对齐）**

**layout\_alignParentRight（和父节点的右边对齐）**

**layout\_alignTop （和XXX顶部对齐）**

**layout\_alignBottom （和XXX底部对齐）**

**layout\_alignLeft (和XXX左边对齐）**

**layout\_alignRight（和XXX右边对齐）**

**Android事件驱动机制**

**一般，用户经常会通过界面与应用交互，Android框架采用事件驱动的形式与用户交互，那如何处理用户界面中触发的事件?**

**可以通过从用户交互的View设置事件监听器的方式来实现对事件的处理，一个事件监听器是View类中一个包含单一回调方法的接口。当注册了监听器的View发生了对应的监听事件时，Android框架就会回调相应的监听方法。**

**常见的用户事件**

**点击事件、选择事件、触屏事件、长按事件、按键事件**

**点击事件实现的四种方法**

**单击事件是事件机制中最常见的事件，通过对控件绑定View.OnClickListener 实现单击事件的监听**

**点击事件四种书写方式**

**& 私有类实现方式**

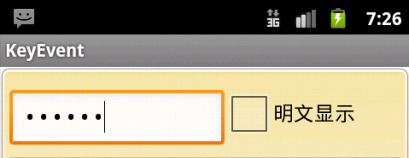
**& 匿名内部类实现方式**

**& 布局中对控件添加android:onClick**

**& Activity实现监听接口**

## 复选事件

**\* 界面设计**

****

**setChecked(true);//设置复选框选中状态**

**isClickable();// 判断勾选框(CheckBox)是否选中**

**选择事件(单选、复选、下拉选择)**

**1. 复选 :**

**控件： CheckBox**

**接口： CompoundButton.OnCheckedChangeListener 状态改变的监听**

**CompoundButton:复合按钮，它是抽象类 ，它是 CheckBox、RadioButton(单选)、ToggleButton(开关选择)的父类**

**案例： 明密文切换 ： wifi 密码 ：显示密码和不显示密码 \*\*\*\***

**显示明文　　et\_info.setInputType(EditorInfo.*TYPE\_CLASS\_TEXT*);**

**显示密文 et\_info.setInputType(129);**

## 单选事件**RadioButton与RadioGroup**

**单选事件的监听接口：RadioGroup.OnCheckedChangeListener**

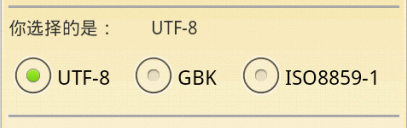
**RadioButton与RadioGroup组合使用才能实现单选功能**

**RadioGroup组中包含RadioButton　通过RadioGroup才能的找到RadioButton**

**\* 案例：选择字符集**

**通过对RadioButton控件的选择监听，实现对字符集的选择**

**\* 界面设计**

****

**onCheckedChanged(RadioGroup** **group：RadioButton int checkedId：选中控件的资源ＩＤ)**

**通过父控件找到子控件**

**RadioButton radioButton = (RadioButton) rg\_code.findViewById(checkedId);**

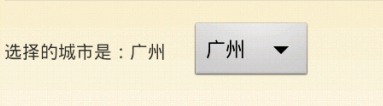
## 下拉列表选择 控件 ： Spinner

**下拉事件的监听接口：AdapterView.OnItemSelectedListener 控件 ： Spinner**

**\* 案例：城市选择**

**通过对Spinner下拉列表监听，实现对城市的选择**

**\* 界面设计**

****

**Spinner控件中　 android:entries=*""　下列的条目***

**setOnItemSelectedListener(new AdapterView.OnItemSelectedListener()**

**onItemSelected(AdapterView<?> parent：Spinner, View view：当前选择下拉项选择视图,  
int position：当前选择下数据在适配器中的位置, long id：当前选择下拉项数据在适配器中的行号属性)**

## 长按与触屏事件

**长按事件监听接口：View.OnLongClickListener**

**触摸事件监听接口：View.OnTouchListener**

**触摸、长按**

**1. 触摸**

**- 控件 View**

**- 接口 View.OnTouchLisener**

**2. 长按**

**- 控件 ： View**

**- 接口 ：View.OnLongClickListener**

**小案例： 通过长按图片 设置手机壁纸**

**3. 三种监听事件的回调顺序： 触摸-->长按 --->点击**

**\* 案例：长按图标设置手机桌面壁纸**

**clearWallpaper ：清除桌面壁纸**

**setWallpaper(BitMap bitmap) :设置桌面壁纸**

**设置壁纸要添加权限：**

**<uses-permission android:name="android.permission.SET\_WALLPAPER"/>**

**Bitmap是位图格式文件，以像素的形式来描述图片 ，比如png，jpg,bmp,gif 都是**

**Drawable :可绘制的资源，包含 Bitmap、图层，可绘制的颜色资源等，它是一个抽象类**

**ImageView可以获取前景图片.getDrawable();//取得前景图片Drawable**

**ImageView可以获取背景图片.getBackground();//取背景**

**点击事件返回结果　return true;**

**返回false，表示当前的请求没有耗尽，android框架还会继续的回调其他的监听方法**

**返回true，表示当前的请求已经结束，android框架不会继续的回调其他的监听方法**

## 键盘事件

**Activity实现了KeyEvent.backcall接口**

**onKeyDown(int keyCode, KeyEvent event) ：当键按下去触发**

**onBackPress():当返回键按下去触发 ，Activity的方法**

**模拟器常见的按键：**

**Back 返回键**

**Home 手机屏幕桌面**

**Ctrl+F11 切换模拟器横竖屏幕**

**F2 手机菜单**

**F3 电话面板**

**F8 手机网络开关**

**按键事件**

**它是实现KeyEvent.CallBack接口 回调方法**

**- onKeyDown() //按键事件**

**它是Activity定义的方法**

**- onBackPressed()//按下返回键**

**小案例： 再按一次退出系统 两秒钟之内按返回键退出系统**

**Activity、View、Window关系**

**\* android.app.Activity**

**Activity作用：Activity是Android四大组件之一，它通过内部的成员变量Window对象来展示一个与用户交互的界面，界面中的View控件捕获事件，**

**通过WindowManagerService传递消息，Android框架再回调Activity中与捕获事件相关联的方法，从而实现与用户的交互。Activity扮演的是一个控制器的角色**

**\* android.view.View**

**& View：视图，是用户接口组件的基本构建块，它在屏幕中占用一个矩形区域，它是所有UI控件的基类，如一个按钮或文本框。View负责图形界面渲染及事件处理**

**& Android已经为我们提供了一系列的标准UI控件供我们直接使用，同时，我们也可以通过继承于View或View的子类，来实现我们自定义的UI控件**

**& View及其子类的关系**

**对View常用的操作**

**Set properties 设置属性**

**Set focus 设置焦点**

**Set listeners 设置监听**

**Set visibility 设置是否可见**

**\* Window**

**Window是Android中的窗口，表示顶级窗口，也就是主窗口，每一个主窗口都拥有一个View，称之为DecorView(装饰视图)，它是主窗口的顶级View(DecorView必须是一个布局容器，因为它要容纳其它的View)。当Activtity调用setContentView()时，实际就是调用Window对象的setContentView()方法，执行该方法，把用户定义的View添加到DecorView中，最终完成对View的展示。**

**认识常用UI控件(widget)**

**Android平台提供了一套完备的、功能强大的组件化模型用于搭建用户界面，这套组件化模型以[View](http://docs.eoeandroid.com/reference/android/view/View.html)和 [ViewGroup](http://docs.eoeandroid.com/reference/android/view/ViewGroup.html)这两个基础布局类为基础。**

**平台本身已预先实现了多种用于构建界面的View子类和ViewGroup子类，他们被分别称为部件（widget）和布局（layout）。**

**部件（widget）包括[Button](http://docs.eoeandroid.com/reference/android/widget/Button.html)、[TextView](http://docs.eoeandroid.com/reference/android/widget/TextView.html)、[EditText](http://docs.eoeandroid.com/reference/android/widget/EditText.html)、[ListView](http://docs.eoeandroid.com/reference/android/widget/ListView.html)、[CheckBox](http://docs.eoeandroid.com/reference/android/widget/CheckBox.html)、[RadioButton](http://docs.eoeandroid.com/reference/android/widget/RadioButton.html)、[Gallery](http://docs.eoeandroid.com/reference/android/widget/Gallery.html)、[Spinner](http://docs.eoeandroid.com/reference/android/widget/Spinner.html)等**

**Android控件**

## ProgressBar**文件下载进度显示**

**ProgressBar 进度条，常用于文件下载进度显示、系统初始化进度等，默认情况是没有进度值的(转圈圈)**

**style="@android:style/Widget.ProgressBar.Horizontal" 样式设置**

**ProgressBar 的max 属性代表其最大刻度值，progress 属性代表当前进度值。使用方法如下：**

**ProgressBar.setMax(int max);设置最大刻度值。**

**getProgress() 得到当前进度值**

**ProgressBar.setProgress(int progress);设置当前进度值**

**给ProgressBar 设置最大刻度值和修改进度值可以在子线程中操作的，其内部已经特殊处理过了，因此不需要再通过handler发送Message 让主线程修改进度。**

**ProgressBar**

**\* 提示点：**

**style="@android:style/Widget.ProgressBar.Horizontal" 样式设置**

**setProgress(int) 设置当前进度值**

**getProgress() 得到当前进度值**

**setMax(int) 设置最大值**

**\* 提示点：**

**style="@android:style/Widget.ProgressBar.Horizontal" 样式设置**

**setProgress(int) 设置当前进度值**

**getProgress() 得到当前进度值**

**setMax(int) 设置最大值**

**RatingBar评分进度条**

**RatingBar 评分进度条，常用于对商品、用户的评价，比如满意度调查等**

**\* 提示点：**

**setOnRatingBarChangeListener 设置监听器**

**setNumStars 设置星星的个数(设置总分)**

**setStepSize 设置分数间隔**

**getRating 获得当前的分数值**

**setRating 设置当前的分数值**

**ToggleButton开关按钮**

**ToggleButton:开关按钮,它是CompoundButton的子类，实现开与关的效果，比如蓝牙开关、wifi开关、声音开关等**

**\* 提示点：**

**- 实现的监听接口：CompoundButton.OnCheckedChangeListener　该接口是监听开关按钮状态的改变**

**ToggleButton - toggle() :取反操作**

**- xml属性设置**

**textOn: 表示状态为开的提示文本**

**textOff: 表示状态为关的提示文本**

**TextView文本本身没有点击事件　需要设置可点击： android:clickable=*"true"***

**图片和文字按下如需不同的效果可以使用选择器**

**图片选择器－>新建ＸＭＬ文件( 文件名不能大写　且不能与图片名相同)－>Drawable－>selector　－>**

**android:state\_checked　能够选择**

**android:state\_pressed　按下**

**android:state\_enabled　能够使用**

**图片选择器　例**

**<item android:drawable=*"@drawable/player\_btn\_volume\_selected"* android:state\_checked=*"true"*></item>**

**<item android:drawable=*"@drawable/player\_btn\_volume\_selected"* android:state\_pressed=*"true"*></item>**

**<item android:drawable=*"@drawable/player\_btn\_metu"*></item>**

**图片控件中背景选择－选择器**

**颜色选择器　例：**

**颜色选择器－>新建ＸＭＬ文件( 文件名不能大写　且不能与图片名相同)－>Color List－>selector　－>**

**文本控件中选择文本颜色——选择器**

**选择器**

**1. 图片选择器**

**控件有很多种状态，依据控件的不同状态赋予相应属性不同的图片**

**其实选择器也是Drawable ，即也是图片，只是图片列表**

**注意： 选择器的文件名不能与图片名相同**

**2. 颜色选择器**

**控件有很多种状态，依据控件的不同状态赋予相应属性不同的颜色值**

**其实选择器也是Color ，即也是图片，只是颜色列表**

**注意： 选择器的文件名不能与颜色属性名相同**

**颜色值描述：argb ：a： alpha r ：红 g ：绿 b：蓝**

**四种格式 ： #RGB, #ARGB, #RRGGBB or #AARRGGBB**

**.9格式图片设置**

**.9格式的图片，是为了解决当图片拉伸时，边角不会失真**

**转换工具：**

**E:\Android\adt-bundle-x86-32\sdk\tools\draw9path.bat**

**点击进入后选择open..打开要转换的文件　－> 然后可以选择－>　show patches　进行设置**

**ScrollView滚屏视图**

**ScrollView:滚屏视图，它是FrameLayout的子类，能够被用户滚动的布局容器，它只能拥有一个子控件，经常应用于一个垂直方向的线性布局中。**

**ScrollView只支持垂直滚动，对于需要水平滚动，则用HorizontalScollView**

## **RatingBar评分进度条**

**RatingBar 评分进度条，常用于对商品、用户的评价，比如满意度调查等**

**\* 提示点：**

**setOnRatingBarChangeListener 设置监听器**

**setNumStars 设置星星的个数(设置总分)**

**setStepSize 设置分数间隔**

**getRating 获得当前的分数值**

**setRating 设置当前的分数值**

**案例：对欧冠精彩程度打分**

**\* 界面设计:**



**\* 代码实现**

**//设置星星的数量**

**ratingBar.setNumStars(5);**

**//设置步长值**

**ratingBar.setStepSize(0.5f);**

**//添加评分控件值改变的监听**

**ratingBar.setOnRatingBarChangeListener(new RatingBar.OnRatingBarChangeListener() {**

**ratingBar:评分控件 rating ：当前的值 fromUser ：是不是用户操作**

**public void onRatingChanged(RatingBar ratingBar, float rating,**

**boolean fromUser) {**

**if(fromUser){**

**Log.v("RatingBar from user", rating+"");**

**}else{**

**Log.v("RatingBar no from user", rating+"");**

**}**

**}**

**});**

## 通知Notification

**早期版本设置内容框**

**Notification notification=new Notification(icon, tickerText, when)**

**notification.setLatestEventInfo(context, contentTitle, contentText, contentIntent)**

**构建通知 ,在level 16及以后**

**Notification.Builder builder=new Notification.Builder(this);**

**builder.setAutoCancel(true);//自动取消 ，当点击了通知内容框，自动取消状态栏中的通知**

**builder.setDefaults(Notification.DEFAULT\_SOUND);//默认声音提示方式**

**builder.setSmallIcon(R.drawable.emoji\_080);//设置通知的图标**

**builder.setTicker("班级重要通知");//通知状态栏的滚屏字幕**

**builder.setWhen(System.currentTimeMillis());//通知的时间**

**设置通知内容框的内容(可以设置自定义界面)**

**builder.setContentTitle("班级通知");//通知内容框的标题**

**builder.setContentText("停电通知，昨天停电");//通知内容框的内容**

**进行打电话**

**Intent intent=new Intent();**

**intent.setAction(Intent.ACTION\_CALL);**

**intent.setData(Uri.parse("tel:110"));**

**pendingIntent　延迟意图，满足某种条件的意图，当点击了内容框，则执行该意图**

**context ：上下文**

**requestCode：请求码 ，当前没有使用**

**intent：意图 ，当点击内容框要进入的组件**

**FLAG\_UPDATE\_CURRENT:更新当前通知状态**

**PendingIntent pendingIntent=PendingIntent.getActivity(this, 0, intent, PendingIntent.FLAG\_UPDATE\_CURRENT);**

**builder.setContentIntent(pendingIntent);//内容意图，当点击通知内容框，后执行的意图**

**Notification notification=builder.build();//通过构建器来构建通知**

**发通知 ，在通知管理器中发通知 ，通过Context取得通知管理器服务**

**NotificationManager notificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION\_SERVICE);**

**链式调用 高版本的写法**

**Notification noti = new Notification.Builder(this)**

**.setContentTitle("我是大标题")**

**.setContentText("我是标题的内容")**

**.setSmallIcon(R.drawable.ic\_launcher)**

**.setLargeIcon(BitmapFactory.decodeResource(getResources(), R.drawable.ic\_launcher))**

**.build();**

**发送通知**

**nm.notify(10, noti);**

**兼容低版本的写法 用过时的方法**

**Notification noti = new Notification(R.drawable.ic\_launcher, "接收到了一条通知", System.currentTimeMillis());**

**实现呼吸灯 震动一下**

**noti.defaults = Notification.DEFAULT\_ALL;**

**不删除通知**

**noti.flags = Notification.FLAG\_NO\_CLEAR;**

**创建意图对象**

**Intent intent = new Intent();**

**实现打电话的逻辑**

**intent.setAction(Intent.ACTION\_CALL);**

**intent.setData(Uri.parse("tel:"+119));**

**需要添加打电话的权限 Task**

**PendingIntent pendingIntent = PendingIntent.getActivity(getApplicationContext(), 1, intent, Intent.FLAG\_ACTIVITY\_NEW\_TASK);**

**noti.setLatestEventInfo(this, "小芳", "今天晚上7天酒店....", pendingIntent);**

**发送通知**

**nm.notify(10, noti);**

**取消通知**

**nm.cancel(10);**

**通知栏和服务连用**

**当服务第一次开启的时候**

**public void onCreate() {**

**确保这个服务运行在前台进程**

**startForeground(id, notification)**

**自定义通知界面**

**RemoteViews 远程视图 ，该界面视图不是在本应用中展示 ，在远程的系统的应用中展示**

**packageName:当前项目的包名**

**layoutId:自定义界面布局的资源id**

**RemoteViews remoteViews=new RemoteViews(getPackageName(), R.layout.notification);**

**设置自定义通知界面的子控件的属性**

**remoteViews.setTextViewText(R.id.title\_tv, "有人想睡觉,但是睡不着");**

**builder.setContent(remoteViews);**

**Notification 是一条显示在应用UI界面之外的通知，一条通知到来时，它首先显示在通知栏中，当对通知往下拉放时，会显示通知的详细信息，**

**当对通知的详细信息框进行点击时，则可通过Intent激活其他组件，实现业务处理。**

**Notification是Android非常重要的消息提示机制，常与广播接收器组合使用，实现广播通知，比如：短信通知、软件更新通知等**

**Android消息提示机制**

**在某些情况下，可能需要你去通知用户发生在你应用中的事件，其中一些事件需要用户响应，有的则不需要。Android框架通过消息机制很好的完成上述的需求。**

**比如：**

**- 当一个事件完成时(比如保存文件)，需要显示一个简短的消息来确认保存成功**

**- 假如应用正在后台运行，且需要用户注意，那么该应用需要创建一个通知以方便用户做出响应**

**- 假如用户要操作一个危险的操作，则应该创建一个对话框通知，再一次确认用户的操作**

**常见的消息提示方式有：**

**- Logcat(日志猫)**

**- Toast(瞬时提示)**

**- Dialog(对话框)**

**-Notification(通知)**

**Logcat日志猫**

**\* 级别 ：v、d、i、w、e 级别由低往高**

**- verbose ：混淆的 ，所有信息都可以输出，级别最低**

**- debug ：调试级别**

**- info：信息级别**

**- warning：警告级别**

**- error ：错误级别**

**\* 添加过滤器 按照Tag进行过滤**

**System.out.println()与Log输出之间的区别**

**- System.out.println()输出的信息 等级是info级别，标记tag：System.out**

**- System.out.print()输出信息 需要System.out.flush()刷缓存输出**

**Toast瞬时提示**

**Toast通知是一种浮现在屏幕上层的消息提醒，它只填充消息所需要的空间，而当前正在运行的活动仍然保持其自身的可见性和交互性。**

**这种通知自动淡入淡出且不接受交互事件**

**-　　　 系统的Toast.makeText ： 白加黑**

**Toast.makeText(context,text,duration).show()**

## **自定义Toast**

**- 自定义吐司**

**1. 代码自定义 Toast显示界面**

**// 构建一个吐司**

**Toast toast = new Toast(context);**

**// 准备一个布局**

**LinearLayout layout = new LinearLayout(context);**

**// 默认的宽和高，可以自己设置宽和高**

**// ViewGroup.LayoutParams layoutParams=new**

**// ViewGroup.LayoutParams(LayoutParams.WRAP\_CONTENT,**

**// LayoutParams.WRAP\_CONTENT);**

**LinearLayout.LayoutParams layoutParams = new LinearLayout.LayoutParams(**

**LayoutParams.WRAP\_CONTENT, LayoutParams.WRAP\_CONTENT);**

**layout.setLayoutParams(layoutParams);// 设置布局参数，宽和高**

**layout.setOrientation(LinearLayout.HORIZONTAL);// 设置方向**

**layout.setBackgroundResource(R.drawable.bg\_yellow);// 设置背景**

**ImageView imageView = new ImageView(context);**

**// 对于代码可以不设置宽和高 ，默认的宽和高 wrap\_content**

**imageView.setImageResource(R.drawable.btn\_up\_normal);// 设置ImageView前景图片**

**// 把ImageView添加到布局中**

**layout.addView(imageView);**

**// 构建一个textView**

**TextView textView = new TextView(context);**

**textView.setText("天津爆炸");**

**textView.setTextSize(20);**

**layout.addView(textView);**

**// 在父控件中把子控件移除**

**// layout.removeView(imageView);**

**toast.setView(layout);// 设置toast的显示界面视图**

**/\*\***

**\* 设置位置 gravity:位置 xOffset： x方向的偏移量 yOffset： y方向的偏移量**

**\*/**

**toast.setGravity(Gravity.TOP | Gravity.CLIP\_HORIZONTAL, -50, 80);**

**toast.setDuration(Toast.LENGTH\_SHORT);// 设置消息的显示持续时间**

**toast.show();// 显示吐司**

**通过XML自定义Toast**

**\* LayoutInflate :把xml布局文件实例化为View对象**

**View是来自xml转换的View对象**

**\* 1.准备xml文件**

**\* 2. 把xml文件转换为View对象**

**// 构建一个吐司**

**Toast toast = new Toast(context);**

**/\*\*LayoutInflater：把布局xml文件实例化为View对象，好比打气筒的形式，逐步把布局转为为具有层次性的View对象**

**\* resource：要转换的xml布局的资源id**

**\* root：转换后的布局对象是否要假如到父控件中**

**\* attachToRoot: 是否把xml转换的View对象添加到root父控件中**

**root：**

**1.布局 （线性、相对、帧等）**

**2. AdapterView 它也是一个容器，也可以作为父控件，但是它的子控件通过Adapter来装配,不是通过addView的方式来添加子控件**

**而LayoutInflate 添加到父控件中，是采用addView的方式添加的**

**\* return:假如有父控件，则返回父控件　假如父控件为空，则返回转换的xml的根节点对象**

**View layoutView=LayoutInflater.from(context).inflate(R.layout.toast, null);**

**//通过父控件来引用子控件**

**ImageView iconIv=(ImageView) layoutView.findViewById(R.id.icon\_iv);**

**iconIv.setImageResource(R.drawable.mood\_photo);**

**TextView titleTv=(TextView)layoutView.findViewById(R.id.title\_tv);**

**titleTv.setText("底层开发！");**

**toast.setView(layoutView);**

**toast.setGravity(Gravity.TOP | Gravity.CLIP\_HORIZONTAL, -50, 80);**

**toast.setDuration(Toast.LENGTH\_SHORT);// 设置消息的显示持续时间**

**toast.show();// 显示吐司**

# LOG日志输出

**在实际开发中，打印日志用Log类 封装一个包 logcat.utils 创建一个LogUtils类 通过boolean变量来控制日志信息是否打印！！！**

**控制台下的Console输出的的是windows下运行的应用程序打印的，而android项目下的apk应用程序是在android系统下运行的。**

**日志发生错误时优先看有没有：Caused by: 信息**

**public class MainActivity extends ActionBarActivity {**

**private static final String *TAG* = "MainActivity";//类名作为标示**

**@Override**

**protected void onCreate(Bundle savedInstanceState) {**

**super.onCreate(savedInstanceState);**

**setContentView(R.layout.*activity\_main*);**

**//Android自带一套日志输出 在Log类中**

**Log.v(tag,msg) Log 类中 TAG 是一个常量标签 以类名作为常量标示 msg：信息**

**Log.*v*(*TAG*, "****verbose");****verbose 表示所有日志输出**

**Log.*d*(*TAG*, "****debug");****debug 调式**

**Log.*i*(*TAG*, "****info");****info 信息**

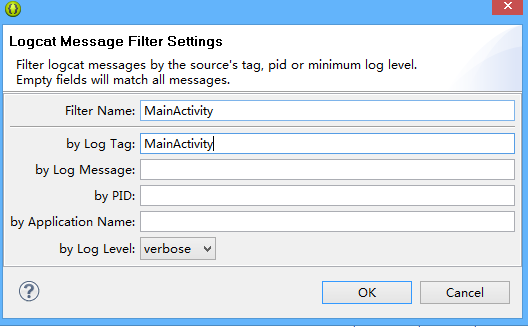
**Log.*w*(*TAG*, "****warn");warn 警告**

**Log.*e*(*TAG*, "****error");****error 错误**

**越往下 日志输出越少 一般 debug info error 用的比较多**

**设置过虑器：**

**在Logcat 左侧点击IMG_256添加**

****

标识

过滤器名

**左侧区域是日志过滤器列表，All messages（no filter）是自带的无任何过滤限制。com.itheima.xxx 包名通常是某个程序运行起来后LogCat 自动添加的过滤器，该过滤器选中后右侧只输出该应用下的日志。此外，我们还可以点击左侧绿色的加号图标去自定义一个过滤规则。**

**右侧区域分为上下两部分。上部分的左侧文本输入框可以输入关键字对日志进行筛选，比如tag:Main，那么日志只显示tag 以Main 开头的日志。**

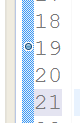
**右侧的下拉框可以选择5 种日志级别，所有的日志过滤配置会对日志的输出同时起效。**

**在日志输出区有如下列，**

**Level：代表日志级别，Time：代表日志输出时间，PID 和TID 代表线程ID，Application：代表应用包名，Tag：代表日志tag 标签，Text：代表日志正文。**

**一般在开发中 新建一个包 包名一般是 utils 创建类 LogUtil 封装方法 设置boolean 值控制方法是否运行打印**

# Android开发中的断点调试 （选择 Debug As）

** 断点放行进入下一个断点 进入下一行F6**

**双击打断点 断点打在出错的上一行**

**Debug As 打开断点调试界面 IMG_256**

**进入方法F5**

**在出错行上面击右键 选择Watch 显示具体错误**

**注意：断点测试完成后一定要放行!!!**

# 测试相关概念

**常见测试分类**

**一、根据是否知道源码**

**黑盒测试：不知道源码，是以用户的角度，从输入数据与输出数据的对应关系出发进行测试的。**

**白盒测试：知道源码,又称结构测试、透明盒测试、逻辑驱动测试或基于代码的测试。**

**二、根据测试粒度分类**

**方法测试： FunctionTest，粒度最低，测试单个方法。**

**单元测试： JunitTest 方法里面会调用其他的方法。**

**联调测试： IntegrationTest 后台与前台集成，各模块之间的集成测试。**

**三、根据测试次数分类**

**冒烟测试：顾名思义，把设备一直测试到冒烟为止。Android 下提供给我们一种冒烟测试的功能Monkey Test 猴子测试，在命令行输入adb shell，**

**进入Linux 命令行。测试整个系统命令：monkey 1000(事件的数量)；测试某个程序：monkey -p 包名事件的数量。**

**压力测试： PressureTest ，给后台用的，主体向被观察者布置一定量任务和作业，借以观察个体完成任务的行为。**

**Android Junit测试**

**在JavaSE 中我们可以使用Junit 进行单元测试，Android 也提供了单元测试框架供我们使用，不同的是该框架可以模拟Android 上下文环境，可以在单元测试中通过getContext()方法获取到context上下文进行使用。注意的是：需要把应用程序部署到真实的手机或者模拟器,在dalvik虚拟机里面运行。**

**Android单元测试框架搭建的步骤：（这里以测试一个计算器加法的业务逻辑为例）**

1. **编写一个计算器的业务类，实现计算器业务相加的方法**

**/\*\***

**\* 计算器的业务类**

**\*/**

**public class CalcService {**

**/\*\***

**\* 计算器相加的业务方法**

**\* @param x**

**\* @param y**

**\* @return**

**\*/**

**public int add(int x,int y){**

**return x+y;**

**}**

**}**

1. **创建Android下的测试框架测试刚才的计算器业务相加的业务逻辑 测试类继承AndroidTestCase**

**当前类就具备测试功能**

**注意：测试方法名必须按照testXXX()的命名规则才能生效。**

**public class TestCalcService extends AndroidTestCase {**

**//记得向测试框架抛出异常.**

**public void testAdd() throws Exception{**

**CalcService calcService = new CalcService();**

**int result = calcService.add(3, 5);**

**//断言 判断结果 参数1：期望的值 参数2：实际的值**

***assertEquals*(8, result);**

**}**

**}**

1. **在清单文件中配置测试框架，这一步很重要！**

**注意：两个标签在不同根节点下**

**<manifest>节点下配置：**

**<!-- 测试的指令集 -->**

**<instrumentation**

**android:name=*"android.test.InstrumentationTestRunner" //测试启动类***

**android:targetPackage=*"com.itheima.junit"* > //需要测试的工程包名**

**</instrumentation**

**<application>节点下配置:**

**<!-- 测试需要的jar包 -->**

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

**Android单元测试：新建一个测试工程**

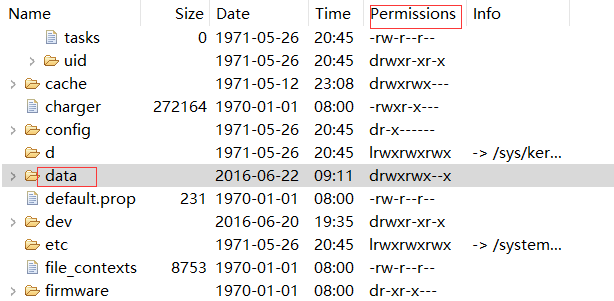
**新建工程 —— Android Test Project ——选择要测试的项目—— 新建类继承AndroidTestCase —— 写测试方法 运行**

# 文件访问权限

**android系统 就是linux系统（底层是linux）**

**根路录：/**

**权限目录**

****

**权限**

**1./data/app 2.date/date程序放置目录**

****

**/mnt/sdcard**

**SD卡目录指向/mnt/sdcard**

**android系统其实是linux系统**

**根目录：/**

**关注于：**

**/data /data/app /data/data internal (内部)**

**/system**

**/mnt /mnt/scard external （外部）**

**/sccard 指向了/mnt/sdcard**

**Linux很安全 很多时候用来作为服务器**

**他的权限控制很牛x**

**Linux 下Permissions的文件十位来表示权限 第一位单独一组 后面9位，每三位表示一组**

**第一位：表示文件类型 d:文件夹 -:文件类型 l:快捷方式**

**第一组(2-4位):该文件所属用户对本文件权限**

**第二组(5-7位):该文件所属用户组对该文件权限**

**第三组(6-9位):其它用户对该文件的权限**

**- ：表示不可执行 rwx二进制： r w x表示1 - 表示0 可以用chmod指令赋权限 r可执行(可以运行)**

**1：首先进入eclipse中的 platform-tools 文件下，记住目录 以本机为例(D:\eclipse\adt-bundle-windows-x86\_64-20140702\adt-bundle-windows-x86\_64-20140702\sdk\platform-tools)**

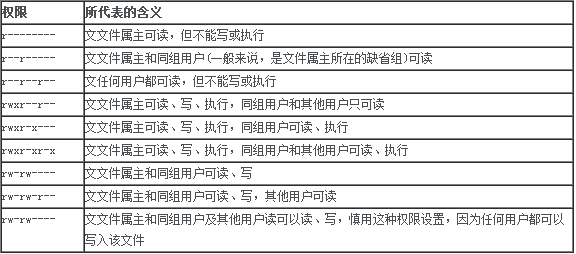
1. **进入命令提示行 键入cd+空格 + 之前复制的目录**
2. **输入 adb+空格 +shell**
3. **输入 cd + 所要改变权限的文件目录**
4. **输入cd files(进入文件夹 ls -l查看该文件夹中所有文件)**
5. **chmod(更改模式) 空格 755(权限十进制的描述 结合上面 r w x ) +文件名**

** **

**android的应用程序的权限：**

**其他的应用默认是无法去读写一个应用程序里面的内容。**

****

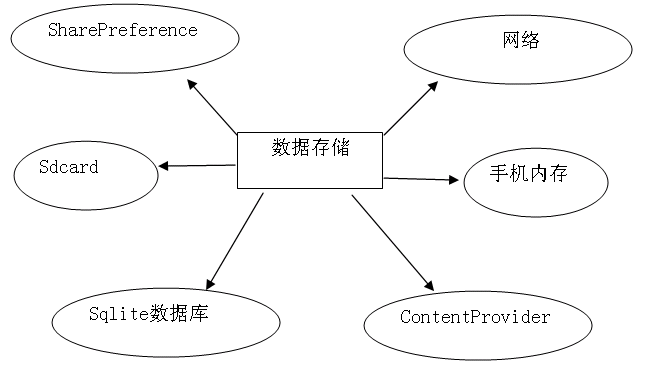
****

# Android下的数据存储

Android系统中的五种数据存储方式：

* **文件存储：以I/O流形式把数据存入手机内存或者SD卡，可以存储大数据，如音乐，图片或者视频等。**
* **SharedPreferences：它本质上是一个XML文件，以Map<Object,Object>形式存入手机内存中。常用于存储简单的参数设置，如QQ登陆账号密码的存储，窗口功能状态的存储等，使用起来简单、方便。**
* **SQLite数据库：SQLite是一个轻量级，跨平台的数据库。数据库所有信息都存储在单一文件内，占用内存小，并且支持基本SQL语法，是项目中经常被采用的一种数据存储方式，通常用于存储用户信息等。**
* **Content Provider：内存提供者，Android四大组件之一，以数据库形式存入手机内存，可以共享自己的数据给其他应用使用。**
* **网络存储：把数据存储到服务器，不存储在本地，使用的时候直接从网络获取，避免了手机端信息丢失以及其他的安全隐患。**

**各种Android移动的主要任务是处理数据，如何将需要处理和处理好的有效的存储起来是一个亟待解决的问题，Android系统提供了非常丰富的移动存储方案。**

**\* 常见移动存储方案：**

**手机内部存储**

**手机外部存储**

**SharedPreferences**

**SQLite**

**ContentProvider**

**网络**

## 外部存储介绍

**所有兼容Android的设备都支持一个可共享的“外部存储(external storage)”,可用来保存文件。这可以是一个可移动的存储设备（比如SD卡）或者一个内部的（不可移动的）存储。保存在外部存储的文件是可全局读写的。**

**\* Sdcard存储的特点：**

**& 依赖于Sdcard，使用SDCard存储，需先检测其状态**

**& 存在Sdcard的文件是可全局读写的**

**& 写入Sdcard时，需要权限**

**\* 关注点：**

**& Environment.getExternalStorageState() 取得外部存储状态**

**& Environment.getExternalStorageDirectory() 取得外部存储根路径**

**& 写入外部存储的权限**

**android.permission.MOUNT\_UNMOUNT\_FILESYSTEMS**

**android.permission.WRITE\_EXTERNAL\_STORAGE**

**\* 外存路径描述**

**& 取得外部存储的公共共享存储路径**

**Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY\_DOWNLOADS)**

**& .在外存中存放应用的私有文件，当应用删除时，该内容会同步删除 ：**

**context.getExternalFilesDir(Environment.DIRECTORY\_DOWNLOADS)**

**目录： /android/data/<包名>/files/<类型>**

**&.在外存中存放应用的私有缓存文件 ,当应用删除时，该内容会同步删除**

**context.getExternalCacheDir()**

**目录：/android/data/<包名>/cache**

**&.取得外存的根路径SD卡**

**Environment.getExternalStorageDirectory()**

**1. 手机内部存储的特点：**

**\* 存放本应用中的私有数据**

**\* 当应用卸载时，/data/data/<包名> 这个目录会同步删除，即手机内部存储的文件会删除**

**\* 存放位置： /data/data/<包名>/files**

**2. 关注点：**

**\* Context.openFileOutput(String name, int mode)**

**- 作用： 打开一个文件，返回一个FileOutputSteam，写数据到文件中**

**- name ： 文件名，有则打开，无则创建**

**- mode ：文件的操作模式 private： 默认模式 ，私有模式，覆盖模式 append ：追加模式，私有模式**

**- 文件的存储的位置：/data/data/<包名>/files ，存放的是私有数据**

**- return： FileOutputSteam**

**\* Context.openFileInput(String name) 从手机内部存储中读取文件**

**- 作用：从手机内部存储中读取文件**

**- name ：文件名**

**- FileInputStream**

**\* context.getCacheDir() 方法用于获取/data/data/<package name>/cache目录**

**- 返回的是File对象 （文件夹）**

**\* context.getFilesDir() 方法用于获取/data/data/<package name>/files目录**

**- 返回的是File对象 （文件夹）**

**手机外部存储**

**所有兼容Android的设备都支持一个可共享的“外部存储(external storage)”,可用来保存文件。这可以是一个可移动的存储设备（比如SD卡）或者一个内部的（不可移动的）存储。保存在外部存储的文件是可全局读写的。**

**1. Sdcard存储的特点(外部存储)：**

**\* 依赖于Sdcard，使用SDCard存储，需先检测其状态，能够挂载成功**

**\* 存在Sdcard的文件是可全局读写的**

**\* 写入Sdcard时，需要权限**

**2. 关注点：**

**\* Environment.getExternalStorageState() 取得外部存储状态 //环境**

**\* Environment.getExternalStorageDirectory() 取得外部存储根路径 //**

**\* 写入外部存储的权限**

**android.permission.MOUNT\_UNMOUNT\_FILESYSTEMS //挂载、卸载文件系统**

**android.permission.WRITE\_EXTERNAL\_STORAGE //写外部存储权限**

**android.permission.READ\_EXTERNAL\_STORAGE //读外部存储的权限**

**在android 4.0以后，假如你的外部存储的路径不规范，则需要该权限**

**3. 外部存储路径描述**

**1.取得外存的公共共享的存储路径**

**取目录：Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY\_DOWNLOADS);**

**/mnt/sdcard/downloads/**

**2.在外存中存放应用的私有数据，但是该数据可以全局读写，当应用删除时，该内容会同步删除 ：**

**取目录：context.getExternalFilesDir(Environment.DIRECTORY\_DOWNLOADS)**

**目录/mnt/sdcard/android/data/<包名>/files/<类型>/**

**3.在外存中存放应用的私有缓存文件 ,当应用删除时，该内容会同步删除**

**取目录: context.getExternalCacheDir()**

**目录 /mtn/sdcard/android/data/<包名>/cache**

**4.取得外存的根路径**

**取目录：Environment.getExternalStorageDirectory()**

**/mnt/sdcard**

## 数据存储私有目录

**1.写布局**

**LinearLayout + RelativeLayout**

**2.写业务逻辑**

**a.找到相应控件**

**b.设置按钮的点击事件**

**c.在onclick方法中，获取用户输入的用户名密码和是否记住密码**

**d.判断用户名密码是否为空，不为空请求服务器（省略，默认请求成功）默认用户注册成功**

**判断是否为空：TextUtils.isEmpty(判断的数据)**

**setChecked(true);//设置复选框选中状态**

**isClickable();// 判断勾选框(CheckBox)是否选中**

**e.判断是否记住密码，如果记住，将用户名密码保存本地**

**//通过context对象获取私有目录，/data/data/packagename/filse**

**context.getFileDir().getPath()**

**将数据存储到私有目录用Context对象获取**

**/ 获取私有目录写入流 name:文件名 mode :文件操作模式(私有 追加 全局读 全局写)**

**//如果文件不存在 会自动创建**

**fos = mContext.openFileOutput("userinfo.txt", Context.*MODE\_PRIVATE*);**

**f.回显用户名密码**

**//获取私有目录读取流Context对象获取**

**fis = mContext.openFileInput("userinfo.txt");**

## 存储到SD卡 （重点）

**使用Sdcard注意事项：**

**1.权限问题：**

**<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>**

**2.硬性编码问题：通过 Environment可以获取sdcard的路径**

**Environment.getExternalStorageDirectory().getPath();**

**3.使用前需要判断sdcard状态**

**if(!Environment.getExternalStorageState().equals( Environment.MEDIA\_MOUNTED)){**

**//sdcard状态是没有挂载的情况**

**Toast.makeText(mContext, "sdcard不存在或未挂载", Toast.LENGTH\_SHORT).show();**

**return ;**

**}**

**4.需要判断sdcard剩余空间**

**//判断sdcard存储空间是否满足文件的存储**

**File sdcard\_filedir = Environment.getExternalStorageDirectory();//得到sdcard的目录作为一个文件对象**

**long usableSpace = sdcard\_filedir.getUsableSpace();//获取文件目录对象剩余空间**

**long totalSpace = sdcard\_filedir.getTotalSpace();//获取文件目录的总空间**

**//将一个long类型的文件大小格式化成用户可以看懂的M，G字符串**

**String usableSpace\_str = Formatter.formatFileSize(mContext, usableSpace);**

**//将SD卡的总空间格式化用户能看懂的M，G字符串**

**String totalSpace\_str = Formatter.formatFileSize(mContext, totalSpace);**

**if(usableSpace < 1024 \* 1024 \* 200){//判断剩余空间是否小于200M**

**Toast.makeText(mContext, "sdcard剩余空间不足,无法满足下载；剩余空间为："+usableSpace\_str, Toast.LENGTH\_SHORT).show();**

**return ;**

**}**

**/data/data: context.getFileDir().getPath();**

**一个应用程序的私有目录，只有当前应用程序有权限访问读写，其他应用无权限访问。**

**一些安全性要求比较高的数据存放在该目录，一般用来存放size比较小的数据。**

**/sdcard: Enviroment.getExternalStorageDirectory().getPath();**

**是一个外部存储目录，只用应用声明了<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>的一个权限，**

**就可以访问读写sdcard目录；所以一般用来存放一些安全性不高的数据，文件size比较大的数据。**

## SharedPrefrerences数据存储

**像登陆这种案例，需要保存用户名，密码，以及存放一些标记性的信息 设置信息 等，使用SharedPrefrerences进行存储是最方便也是最常用的，**

**SharedPreferences，是一个非常轻量的数据存储方式，以xml的形式存取简单的键值对数据，数据类型包括（ints,floats,boolean,strings,longs,Set<String>(android 11以后)），存放位置：/data/data/<包名>/shared\_prefs，存放的是应用私有的数据，主要用于软件偏好设置，简单信息存取等**

**\* 获取SharedPreference对象**

**- Context.getSharedPreferences(String name,int mode)**

**name:指定XML文件名，不需要指定.xml后缀，name存在则打开，没有则创建**

**mode：文件的操作模式，见文件存储的文件操作模式表格**

**- Activity.getPreferences(int mode)**

**name为默认值：即当前访问的Activity的类名**

**\* 操作SharedPreference对象**

**& 取得参数值**

**- SharedPreference.getString(key,defValue) 取得参数值，类型为字符串**

**- SharedPreference.getBoolean(key,defValue) 取得参数值，类型为布尔型**

**& 存储参数值**

**- Editor edit = SharedPreferences.edit(); // 获取编辑器**

**- edit.putString(key, value); // 存储字符串数据**

**- edit.putBoolean(key, value); // 存布尔数据，**

**- edit.commit(); // 提交新值**

**存储的时候通过key/value键值对的形式保存在XML文件中，该文件位于：data/data/包名/shared\_prefs文件夹中。**

**获取SharedPrefrerences 第一种方式通过context.getSharedPrefrences(String name,int mode)获取SharedPrefrences的实例对象**

**在Activity中可以直接使用this代表上下文，如果不是在activity中需要使用Context上下文来获取。**

**SharedPreferences sp = this.getSharedPreferences("config", 0);**

**第一个参数config代表的是该sp文件的名字 不用加扩展名**

**第二个参数0代表的是sp文件的操作模式：私有模式MODE\_PRIVATE，代表只有本应用程序可以读写。**

**也有其他的一些模式比如：MODE\_WORLD\_READABLE MODE\_WORLD\_WRITEABLE**

**指定该SharedPreferences中的数据可以被其他应用程序读写。**

**获取SharedPrefrerences，第二种方式通过PreferenceManager获取SharedPrefrences的默认对象，**

**通过可以获取一个默认的sharepreferences对象**

**SharedPreferences sharedPreferences = PreferenceManager.getDefaultSharedPreferences(context);**

**SharedPreferences对象本身只能获取数据，并不支持数据的存储和修改，数据的存储修改需要通过SharedPreferences.Editor()对象实现，**

**登陆案例中保存数据的时候代码为：**

1. **通过Context对象创建一个SharedPreference对象**
2. **//name:sharedpreference文件的名称 mode:文件的操作模式**

**SharedPreferences sharedPreferences = context.getSharedPreferences("userinfo.txt", Context.MODE\_PRIVATE);**

**2.通过sharedPreferences对象获取一个Editor对象**

**Editor editor = sharedPreferences.edit();**

**3.往Editor中添加数据**

**editor.putString("username", username);**

**editor.putString("password", password);**

**4.提交Editor对象**

**editor.commit();**

**登陆案例中获取数据的时候代码为：**

**1.通过Context对象创建一个SharedPreference对象**

**SharedPreferences sharedPreferences = context.getSharedPreferences("userinfo.txt", Context.MODE\_PRIVATE);**

**2.通过sharedPreference获取存放的数据**

**//key:存放数据时的key defValue: 默认值,根据业务需求来写**

**String username = sharedPreferences.getString("username", "");**

**String password = sharedPreferences.getString("password", "");**

# XML生成与解析

**解析XML文件有三种方式:**

1. **DOM解析 是一种基于对象的API，它会将XML文件的所有内容以文档树方式存放在内存中，然后允许使用DOM API遍历XML树、检索所需的数据，这样便能根据树的结构以节点形式来对文件进行操作。**

**由于DOM需要将整个XML文件以文档树的形式存放在内存中，消耗内存比较大，在Android下不介意使用该种方式进行解析。**

1. **SAX解析 会逐行扫描XML文档，当遇到标签时触发解析处理器，采用事件处理的方式解析XML。 它在读取文档的同时即可对XML进行处理，不必等到文档加载结束，相对快捷，而且也不需要将整个文档加载进内存，因此不存在占用内存的问题，可以解析超大XML。但是，SAX解析只能用来读取XML的数据，无法进行增删改。**
2. **PULL解析 跟SAX解析类似，也是基于事件的逐行解析方式 。PULL解析器是一个开源的Java项目，既可以用于Android应用，也可以用与JavaEE程序。Android已经集成了PULL解析器，因此，在android中最常用的解析方式就是PULL解析。**

**SAX和PULL解析对比：**

**Pull 解析器的运行方式与SAX 解析器相似，都属于事件驱动模式。它提供了类似的事件，如：开始元素和结束元素事件，使用parser.next()可以进入下一个元素并触发相应事件。事件将作为数值代码被发送，因此可以使用一个switch 对感兴趣的事件进行处理。当元素开始解析时，调用parser.nextText()方法可以获取下一个Text 类型元素的值。**

**SAX 解析器的工作方式是自动将事件推入事件处理器进行处理，因此你不能控制事件的处理主动结束；而Pull 解析器的工作方式为允许你的应用程序代码主动从解析器中获取事件，正因为是主动获取事件，因此可以在满足了需要的条件后不再获取事件，结束解析。**

## XmlSerializer生成XML 文件

1. **通过XM类来获取XmlSerializer对象**
2. **设置XmlSerializer的参数 设置写入到哪个文件中**
3. **序列化XML的声明头**
4. **序列化一个根节点的开始结点**
5. **循环遍历集合**
6. **序列化一个根节点的结束结点**
7. **将XML写入到文件中 完成序列化**

**public static boolean smsbackup(Context mContext) {**

**try {**

**ArrayList<SmsBean> list = SmsDao.*getAllSms*();**

**System.*out*.println("......list......."+list);**

**// 1. 通过XML获取XmlSerializer对象**

**XmlSerializer xSerializer = Xml.*newSerializer*();**

**// 2. 设置XmlSerializer参数 写入到哪个文件中**

**// os:xml文件写入流 encoding：流的编码**

**xSerializer.setOutput(mContext.openFileOutput("backupsms.xml",**

**Context.*MODE\_PRIVATE*), "UTF-8");**

**// 3.序列化一个xml的声明头**

**// encoding:xml文件的编码 standalone:是否独立(没有依赖关系)**

**xSerializer.startDocument("UTF-8", true);**

**// 4.序列化一个根节点的开始结点**

**// namespace 命名空间, name：标签的名称**

**xSerializer.startTag(null, "Smss");**

**// 5.循环遍历集合 序列化数据**

**for (SmsBean smsBean : list) {**

**xSerializer.startTag(null, "Sms");**

**// namespace 命名空间, name属性名, value属性值**

**xSerializer.attribute(null, "id", smsBean.id + "");**

**// 开始标签**

**xSerializer.startTag(null, "number");**

**// 标签内容**

**xSerializer.text(smsBean.number);**

**// 结束标签**

**xSerializer.endTag(null, "number");**

**// 开始标签**

**xSerializer.startTag(null, "msg");**

**// 标签内容**

**xSerializer.text(smsBean.msg);**

**// 结束标签**

**xSerializer.endTag(null, "msg");**

**// 开始标签**

**xSerializer.startTag(null, "date");**

**// 标签内容**

**xSerializer.text(smsBean.date);**

**// 结束标签**

**xSerializer.endTag(null, "date");**

**xSerializer.endTag(null, "Sms");**

**}**

**// 6.序列化一个根节点的结束结点**

**xSerializer.endTag(null, "Smss");**

**// 7.将xml写入到文件中 完成xml序列化**

**xSerializer.endDocument();**

**return true;**

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**return false;**

**}**

**}**

**Pull解析**

**XML文件是非常重要的数据格式，它可以用来作配置文件，存储数据等，Android中可以通过SAX、DOM、PULL来解析 XML数据**

**\* pull解析：**

**Xml文件边导入内存，边解析，采用事件驱动机制，当解析到一个节点，返回的是该节点的事件类型，且不会继续往下解析，需手动指向下一个节点，才能继续往下解析，直到文档的结尾。Android默认采用pull解析。**

**\* 事件类型：**

**- START\_DOCUMENT : 开始文档**

**- START\_TAG : 开始标签**

**- END\_TAG : 结束标签**

**- END\_DOCUMENT : 结束文档**

**使用pull解析xml格式的数据**

**try {**

**// //1.通过Xml获取一个XmlPullParser对象**

**XmlPullParser parser = Xml.*newPullParser*();**

**// 2.设置XmlPullParser对象的参数，需要解析的是哪个xml文件,设置一个文件读取流**

**parser.setInput(mContext.openFileInput("backupsms.xml"), "UTF-8");**

**// 3.获取当前XML事件**

**int type = parser.getEventType();**

**// 4.判断事件类型是否是文档结束的事件类型**

**while (type != XmlPullParser.*END\_DOCUMENT*) {**

**switch (type) {**

**case XmlPullParser.*START\_TAG*:// 开始标签**

**if (parser.getName().equals("Smss")) {**

**// 创建集合对象**

**list = new ArrayList<SmsBean>();**

**} else if (parser.getName().equals("Sms")) {**

**sBean = new SmsBean();**

**// 获取属性ID 添加到对象中**

**sBean.id = Integer.*parseInt*(parser.getAttributeValue(**

**null, "id"));**

**} else if (parser.getName().equals("number")) {**

**sBean.number = parser.nextText();**

**} else if (parser.getName().equals("msg")) {**

**sBean.msg = parser.nextText();**

**} else if (parser.getName().equals("date")) {**

**sBean.date = parser.nextText();**

**}**

**break;**

**case XmlPullParser.*END\_TAG*:// 结束标签**

**// 当前结束标签是Sms的话，一条短信数据封装完成， 可以加入list中**

**if (parser.getName().equals("Sms")) {**

**list.add(sBean);**

**}**

**break;**

**}**

**// 获取下一行的事件**

**type = parser.next();**

# JSON的生成与解析

## 解析JSON

**JSONObject root\_json = new JSONObject(result);//将一个字符串封装成一个json对象。**

**JSONArray jsonArray = root\_json.getJSONArray("newss");//获取root\_json中的newss作为jsonArray对象**

**for (int i = 0 ;i < jsonArray.length();i++){//循环遍历jsonArray**

**JSONObject news\_json = jsonArray.getJSONObject(i);//获取一条新闻的json**

**NewsBean newsBean = new NewsBean();**

**newsBean. id = news\_json.getInt("id");**

**newsBean. comment = news\_json.getInt("comment");//评论数**

**newsBean. type = news\_json.getInt("type");//新闻的类型，0 ：头条 1 ：娱乐 2.体育**

**newsBean. time = news\_json.getString("time");**

**newsBean. des = news\_json.getString("des");**

**newsBean. title = news\_json.getString("title");**

**newsBean. news\_url = news\_json.getString("news\_url");**

**newsBean. icon\_url = news\_json.getString("icon\_url");**

**arrayList.add(newsBean);**

**}**

## **生成JSON**

# 异常

**SecurityException 权限拒绝异常 需添加权限**

**IllegalArgumentE 参数异常**

# 对话框

**对话框通常是一种显示在当前活动之上的小窗口，这时候下层的活动将失去焦点，由对话框来实现与用户的交互，常用于添加简单的数据或对操作进一步确认等，**

**比如删除联系人信息、添加类别等，但对话框不宜过多，会影响用户体验**

**常见的对话框有：标准对话框、单选对话框、复选对话框、自定义对话框**

## 标准对话框

**标准对话框是使用最多的一种对话框样式，包含对话框图标、标题、提示系统、确认和取消。**

**\* 创建标准对话框流程：**

1. **生成对话框构建器对象AlertDialog.Builder**
2. **设置图标Builder.setIcon**
3. **设置标题**
4. **设置提示信息**
5. **添加确认与取消按钮**
6. **对确认与取消按钮中添加监听，实现对话框的业务逻辑**

**//采用的是工厂设计模试**

**Builder builder = new AlertDialog.Builder(mContext);**

**//设置图标**

**builder.setIcon(R.drawable.*emoji\_059*);**

**builder.setTitle("标准对话框");**

**//设置信息**

**builder.setMessage("你什么时候去死");**

**//设置是否可以中断对话框 ,即点击Activity的其他界面，而不显示对话框**

**builder.setCancelable(false);**

**builder.setPositiveButton　当点击了对话框的确定按钮，则回调该方法**

**builder.setNegativeButton　当点击了对话框的取消按钮，则回调该方法**

**//创建对话框**

**AlertDialog create = builder.create();**

**create.show();**

## 菜单式对话框

**菜单式对话框适用于单选某个列表项或者对某项数据有几种不同的操作等应用场景**

**关注点**

**& Builder.setItems(items, listener) :设置单选项**

**items：单选项字符串数组**

**通过xml资源id获取数组**

**String[] stringArray = getResources().getStringArray(R.array.android\_versions);**

**当点击对话框的item项时，回调该方法 which：点击了哪一个条目，哪一个项，哪一个菜单**

**.setItems(R.array.android\_versions, new DialogInterface.OnClickListener() {**

**public void onClick(DialogInterface dialog, int which) {**

**ToastUtil.show(context, versions[which]);**

**}**

**})**

**.create()**

**.show();**

**单选对话框**

**构建AlertDialog**

**AlertDialog.Builder builder = new Builder(this);**

**builder.setTitle("请选择您喜欢的课");**

**final String items[] = { "Android", "ios", "php", "c", "C++", "html" };**

**builder.setSingleChoiceItems(items, -1, new OnClickListener() {**

**public void onClick(DialogInterface dialog, int which) {**

**取出点中的条目**

**String item = items[which];**

**Toast.makeText(getApplicationContext(), item, 1).show();**

**关闭当前对话框**

**dialog.dismiss();**

**}**

**});**

**最后一步一定要记得 show出来**

**builder.show();**

## 复选对话框

**复选对话框适用于以对话框的形式复选某个列表**

**\* 关注点：**

**& Builder.setMultiChoiceItems(CharSequence[] items, boolean[] checkedItems, OnMultiChoiceClickListener listener) : 设置复选监听**

**items：复选项字符串数组**

**checkedItems：复选项默认值 可以设置null ，即所有的复选项默认为false ， 也可以具体设置值，但是必须与复选项的长度对应**

**listener：复选单击监听**

**& AlertDialog.getListView() : 得到用于对话框的列表视图—ListView**

**ListView.getCheckedItemPositions() 得到选择项的位置信息及其状态 ，返回值：SparseBooleanArray**

**& 通过getCheckedItemPositions();获取SparseBooleanArray : 稀疏布尔数组，用于映射整数到布尔值 ，与一般的布尔数组不同，**

**下标可以允许有间隙，它比用HashMap映射Integers到Booleans 更有效率**

**private void checkDialog() {**

**String[] stringArray = getResources().getStringArray(R.array.*android\_versions*);**

**Builder builder = new AlertDialog.Builder(mContext);**

**builder.setIcon(R.drawable.*emoji\_101*);**

**builder.setTitle("复选对话框");**

**设置复选项监听 itemsId：复选项的资源id**

**checkedItems： 复选项的初始值 可以设置null ，即所有的复选项默认为false ， 也可以具体设置值，但是必须与复选项的长度对应**

**builder.setMultiChoiceItems(stringArray, new boolean[]{true,false,false,false,false,false,false}, newOnMultiChoiceClickListener() {**

**/\*\*当点击复选项时回调该方法 dialog： AlertDialog**

**\* which ：点击了哪一个复选项，复选项数据在装配中的数据的位置 position**

**\* isChecked：当前操作的复选项的状态\*/**

**@Override**

**public void onClick(DialogInterface dialog, int which, boolean isChecked) {**

**AlertDialog alertDialog = (AlertDialog) dialog;**

**//复选对话框的复选项是由ListView来实现的 ，通过ListView获取adapter，再通过适配器来获取数据**

**String itemStr = alertDialog.getListView().getAdapter().getItem(which).toString();**

**Toast.*makeText*(mContext, itemStr+isChecked, Toast.*LENGTH\_SHORT*).show();**

**}**

**});**

**builder.setPositiveButton("确定", new DialogInterface.OnClickListener() {**

**//点击确定回调该方法**

**@Override**

**public void onClick(DialogInterface dialog, int which) {**

**AlertDialog alertDialog = (AlertDialog) dialog;**

**ListView listView = alertDialog.getListView();**

**//通过listView取得稀疏布尔数组 相当于hashmap ：key(integer:position) value(boolean:复选项的状态)**

**SparseBooleanArray sparseBooleanArray = listView.getCheckedItemPositions();**

**int size = sparseBooleanArray.size();**

**StringBuilder sb = new StringBuilder("");**

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

**//判断选取的状态是否为true 如果为true保存**

**if (sparseBooleanArray.valueAt(i)) {**

**//依据保存的位置position来获取复选项复选文本**

**int keyAt = sparseBooleanArray.keyAt(i);**

**String item = listView.getAdapter().getItem(keyAt).toString();**

**sb.append(item);**

**}**

**}**

**Toast.*makeText*(mContext, sb.toString(), Toast.*LENGTH\_SHORT*).show();**

**}**

**});**

**builder.create().show();**

**}**

**另一种写法**

**构建AlertDialog**

**AlertDialog.Builder builder = new Builder(this);**

**builder.setTitle("请选择您喜欢吃的水果");**

**final String items[] = { "榴莲", "苹果", "葡萄", "香蕉", "黄瓜", "火龙果", "荔枝" };**

**final boolean[] checkedItems = { true, false, false, false, false,false, true };**

**builder.setMultiChoiceItems(items, checkedItems,**

**new OnMultiChoiceClickListener() {**

**@Override**

**public void onClick(DialogInterface dialog, int which,**

**boolean isChecked) {**

**}**

**});**

**builder.setPositiveButton("确定", new OnClickListener() {**

**@Override**

**public void onClick(DialogInterface dialog, int which) {**

**StringBuffer sb = new StringBuffer();**

**// [1]把你选中的水果给取出来**

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

**if (checkedItems[i]) {**

**// 就证明是选中的**

**String fruit = items[i];**

**sb.append(fruit + " ");**

**}**

**}**

**Toast.makeText(getApplicationContext(), sb.toString(), 1)**

**.show();**

**// 关闭对话框**

**dialog.dismiss();**

**}**

**});**

**// 最后一步一定要记得 show出来**

**builder.show();**

**}**

## 自定义对话框

**系统对话框太单一，在开发中常需要自定义对话框**

**\* 关注点：**

**& new Dialog(context,theme):以自定义的样式构建对话框**

**& Dialog.setContentView():设置对话框的布局**

**& Dialog.findViewById() : 通过findViewById找到对话框布局中的控件**

**/\*\*自定义对话框实现步骤：**

**\* 1.xml来设置对话框界面**

**\* 2. 构建一个对话框，并且设置自定义对话框界面**

**\* 3. 获取 类别编辑框和 确定按钮**

**\* 4. 对确定按钮添加点击事件监听，当点击按钮时，获取到用户输入的类别信息**

**\*/**

**自定义的样式承继Dialog的样式　例**

**<style name="MyDialogTheme" parent="android:Theme.Dialog">**

**<item name="android:windowBackground">@color/transparent</item>**

**</style>**

**//构建一个对话框 ,默认会添加一个主题(添加自定义的主题 )**

**final Dialog dialog=new Dialog(context,R.style.MyDialogTheme);**

**//设置对话框的内容视图 构建一个对话框，并且设置自定义对话框界面**

**dialog.setContentView(R.layout.diy\_dialog);**

**//通过父控件引用子控件**

**final EditText typeEdt=(EditText) dialog.findViewById(R.id.type\_et);**

**Button okBtn=(Button)dialog.findViewById(R.id.ok\_btn);**

**//设置点击监听事件**

**okBtn.setOnClickListener(new View.OnClickListener() {**

**@Override**

**public void onClick(View v) {**

**String type = typeEdt.getText().toString().trim();**

**dialog.dismiss();//消失，解雇 ，dialog显示要依赖Activtiy**

**ToastUtil.show(context, type);**

**}**

**});**

**dialog.show();//显示对话框**

**}**

## 进度条对话框

**case R.id.*bt\_progress*:// 进度条对话框**

**与进度相关的进程都可以直接在子线程更新UI**

**通过AlertDialog 创建对话框**

**final ProgressDialog dialog\_progress= new ProgressDialog(MainActivity.this);**

**dialog\_progress.setTitle("正在玩命加载中");//默认样式是转圈**

**dialog\_progress.setProgressStyle(ProgressDialog.*STYLE\_HORIZONTAL*);//改成水平样式**

**dialog\_progress.show();**

**设置进度的最大值**

**dialog\_progress.setMax(100);**

**设置当前的进度**

**dialog\_progress.setProgress(i);**

## 样式与主题

**作用： 统一风格 ，简化操作**

**语法： 通过 values ：设置style类别 来定义样式与主题**

**样式与主题的联系与区别：**

**- 定义的语法都是一样的**

**- 使用场景不同**

**1. 样式针对的是控件 ：Button、ProgressBar --style xml属性 ：style**

**例：自定义样式**

**<style name="MyButtonStyle">**

**<item name="android:layout\_width">wrap\_content</item>**

**</style>**

**2. 主题针对的是组件，比如Activity、Application、 Dialog xml属性： android：theme**

**style   Theme**

**共同点:(1)定义的方式是一样的**

**不同点:(1)style作用范围比较窄 (控件 button textview)  theme 作用在activity或者Application节点下**

**style ：定义样式与主题 可以在velues目录下 新建 xml 文件 或在velues目录下的styles.xml文件中创建**

**抽取共性**

**<style>**

**<item name=*"android:layout\_width"*>wrap\_content</item>**

**<item name=*"android:layout\_height"*>wrap\_content</item>**

**<item name=*"android:textColor"*>#000000</item>**

**<item name=*"android:textSize"*>30sp</item>**

**</style>**

**然后在布局文件中引用**

# 常见的数据库管理系统

**MYSQL ：开源免费的数据库，小型的数据库.已经被Oracle收购了.MySQL6.x版本也开始收费。**

**Oracle ：收费的大型数据库，Oracle公司的产品。Oracle收购SUN公司，收购MYSQL。**

**DB2 ：IBM公司的数据库产品,收费的。常应用在银行系统中.**

**SQLServer：MicroSoft 公司收费的中型的数据库。C#、.net等语言常使用。**

**SyBase ：已经淡出历史舞台。提供了一个非常专业数据建模的工具PowerDesigner。**

**SQLite : 嵌入式的小型数据库，应用在手机端。**

**SQL分类：**

* + **数据定义语言：简称DDL(Data Definition Language)，用来定义数据库对象：数据库，表，列等。关键字：create，alter，drop等**
  + **数据操作语言：简称DML(Data Manipulation Language)，用来对数据库中表的记录进行更新。关键字：insert，delete，update等**
  + **数据控制语言：简称DCL(Data Control Language)，用来定义数据库的访问权限和安全级别，及创建用户。**
  + **数据查询语言：简称DQL(Data Query Language)，用来查询数据库中表的记录。关键字：select，from，where等**

## 数据库字段

* **创建数据库**

**格式:**

**\* create database 数据库名;**

**\* create database 数据库名 character set 字符集;**

* **查看数据库**

**查看数据库MySQL服务器中的所有的数据库:**

**show databases;**

**查看某个数据库的定义的信息:**

**show create database 数据库名;**

**例如：**

**show create database day21\_1;**

* **删除数据库**

**drop database 数据库名称;**

* **其他的数据库操作命令**

**切换数据库：**

**use 数据库名;**

**例如：**

**use day21\_1;**

* **查看正在使用的数据库:**

**select database();**

* **格式：**

**create table 表名(**

**字段名 类型(长度) 约束,**

**字段名 类型(长度) 约束**

**);**

**主键是用于标识当前记录的字段。它的特点是非空，唯一。在开发中一般情况下主键是不具备任何含义，只是用于标识当前记录。**

**格式：**

* **1.在创建表时创建主键，在字段后面加上 primary key.**

**create table tablename(**

**\_id integer primary key,**

**.......**

**)**

* **.删除主键：alter table 表名 drop primary key;**

**alter table sort drop primary key;**

* **4.主键自动增长：一般主键是自增长的字段，不需要指定。**

**实现添加自增长语句,主键字段后加autoincrement(只适用MySQL)**

* **.删除主键：alter table 表名 drop primary key;**

**alter table sort drop primary key;**

* **4.主键自动增长：一般主键是自增长的字段，不需要指定。**

**实现添加自增长语句,主键字段后加auto\_increment(只适用MySQL)**

* **查看数据库中的所有表：**

**格式：show tables;**

* **查看表结构：**

**格式：desc 表名;**

**删除表**

**格式：drop table 表名;**

**例如：drop table sort;**

### **修改表结构格式：**

* **alter table 表名 add 列名 类型(长度) 约束;**

**作用：修改表添加列.**

**例如：**

**#1，为分类表添加一个新的字段为 分类描述 varchar(20)**

**ALTER TABLE sort ADD sdesc VARCHAR(20);**

### **修改表结构格式：**

* **alter table 表名 add 列名 类型(长度) 约束;**

**作用：修改表添加列.**

**例如：**

**#1，为分类表添加一个新的字段为 分类描述 varchar(20)**

**ALTER TABLE sort ADD sdesc VARCHAR(20);**

* **alter table 表名 modify 列名 类型(长度) 约束;**

**作用：修改表修改列的类型长度及约束.**

**例如：**

**#2, 为分类表的分类名称字段进行修改，类型varchar(50) 添加约束 not null**

**ALTER TABLE sort MODIFY sname VARCHAR(50) NOT NULL;**

* **alter table 表名 change 旧列名 新列名 类型(长度) 约束;**

**作用：修改表修改列名.**

**例如：**

**#3, 为分类表的分类名称字段进行更换 更换为 snamesname varchar(30)**

**ALTER TABLE sort CHANGE sname snamename VARCHAR(30);**

* **alter table 表名 drop 列名;**

**作用：修改表删除列.**

**例如：**

**#4, 删除分类表中snamename这列**

**ALTER TABLE sort DROP snamename;**

* **rename table 表名 to 新表名;**

**作用：修改表名**

**例如：**

**#5, 为分类表sort 改名成 category**

**RENAME TABLE sort TO category;**

* **alter table 表名 character set 字符集;**

**作用：修改表的字符集**

**例如：**

**#6, 为分类表 category 的编码表进行修改，修改成 gbk**

**ALTER TABLE category CHARACTER SET gbk;**

### **插入表记录：**

* **语法：**

**insert into 表 (列名1,列名2,列名3..) values (值1,值2,值3..); -- 向表中插入某些列**

**insert into 表 values (值1,值2,值3..); --向表中插入所有列**

* **注意：**

**插入的数据应与字段的数据类型相同**

**数据的大小应该在列的长度范围内**

**在values中列出的数据位置必须与被加入列的排列位置相对应。**

**除了数值类型外，其它的字段类型的值必须使用引号引起。**

**如果要插入空值，可以不写字段，或者插入 null。**

**对于自动增长的列在操作时，直接插入null值即可。**

* **例如：**

**INSERT INTO sort(sid,sname) VALUES('s001', '电器');**

**INSERT INTO sort(sid,sname) VALUES('s002', '服饰');**

**INSERT INTO sort VALUES('s003', '化妆品');**

**INSERT INTO sort VALUES('s004','书籍');**

### **更新表记录：**

**用来修改指定条件的数据，将满足条件的记录指定列修改为指定值**

* **语法：**

**update 表名 set 字段名=值,字段名=值;**

**update 表名 set 字段名=值,字段名=值 where 条件;**

* **注意：**
  + - **列名的类型与修改的值要一致.**
    - **修改值得时候不能超过最大长度.**
    - **值如果是字符串或者日期需要加’’.**

**例如：**

**#1，将指定的sname字段中的值 修改成 日用品**

**UPDATE sort SET sname='日用品';**

**#2, 将sid为s002的记录中的sname改成 日用品**

**UPDATE sort SET sname='日用品' WHERE sid='s002';**

**UPDATE sort SET sname='日用品' WHERE sid='s003';**

### **删除记录：delete**

* **语法：**

**delete from 表名 [where 条件];**

**或者**

**truncate table 表名;**

* **面试题：**

**删除表中所有记录使用delete from 表名; 还是用truncate table 表名;**

**删除方式：delete 一条一条删除，不清空auto\_increment记录数。**

**truncate 直接将表删除，重新建表，auto\_increment将置为零，从新开始。**

**我们在dos命令行操作中文时，会报错**

**insert into user(username,password) values(‘张三’,’123’);**

**ERROR 1366 (HY000): Incorrect string value: '\xD5\xC5\xC8\xFD' for column 'username' at row 1**

**原因:因为mysql的客户端编码的问题我们的是utf8,而系统的cmd窗口编码是gbk**

**解决方案（临时解决方案）:修改mysql客户端编码。**

**show variables like 'character%'; 查看所有mysql的编码**

**在图中与客户端有关的编码设置:**

**client connetion result 和客户端相关**

**database server system 和服务器端相关**

* **将客户端编码修改为gbk.**

**set character\_set\_results=gbk; / set names gbk;**

**以上操作，只针对当前窗口有效果，如果关闭了服务器便失效。如果想要永久修改，通过以下方式:**

* **在mysql安装目录下有my.ini文件**

**default-character-set=gbk 客户端编码设置**

**character-set-server=utf8 服务器端编码设置**

**注意:修改完成配置文件，重启服务**

## **查询的语法：**

* **查询指定字段信息**

**select 字段1,字段2,...from 表名;**

**例如：**

**select id,name from zhangwu;**

* **查询表中所有字段**

**select \* from 表名;**

**例如：**

**select \* from zhangwu;**

**注意:使用"\*"在练习、学习过程中可以使用，在实际开发中，不推荐使用。原因，要查询的字段信息不明确，若字段数量很多，会导致查询速度很慢。**

* **distinct用于去除重复记录**

**select distinct 字段 from 表名;**

**例如：**

**select distinct money from zhangwu;**

* **别名查询，使用的as关键字，as可以省略的.**

**别名可以给表中的字段，表设置别名。 当查询语句复杂时，使用别名可以极大的简便操作。**

**表别名格式:**

**select \* from 表名 as 别名;**

**或**

**select \* from 表名 别名;**

**列别名格式：**

**select 字段名 as 别名 from 表名;**

**或**

**select 字段名 别名 from 表名;**

**例如**

**表别名：**

**select \* from zhangwu as zw;**

**列别名：**

**select money as m from zhangwu;**

**或**

**select money m from zhangwu;**

* **我们在sql语句的操作中，可以直接对列进行运算。**

**例如：将所有账务的金额+10000元进行显示.**

**select pname,price+10000 from product;**

## **条件查询**

**where语句表条件过滤。满足条件操作，不满足不操作，多用于数据的查询与修改。**

**格式 :select 字段 from 表名 where 条件;**

**例如：**

**查询所有吃饭支出记录**

**SELECT \* FROM zhangwu WHERE name = '吃饭支出';**

**查询出金额大于1000的信息**

**SELECT \* FROM zhangwu WHERE money >1000;**

**查询出金额在2000-5000之间的账务信息**

**SELECT \* FROM zhangwu WHERE money >=2000 AND money <=5000;**

**或**

**SELECT \* FROM zhangwu WHERE money BETWEEN 2000 AND 5000;**

**查询出金额是1000或5000或3500的商品信息**

**SELECT \* FROM zhangwu WHERE money =1000 OR money =5000 OR money =3500;**

**或**

**SELECT \* FROM zhangwu WHERE money IN(1000,5000,3500);**

**查询出账务名称包含”支出”的账务信息。**

**SELECT \* FROM zhangwu WHERE name LIKE "%支出%";**

**查询出账务名称中是无五个字的账务信息**

**SELECT \* FROM gjp\_ledger WHERE ldesc LIKE "\_\_\_\_\_"; -- 五个下划线\_**

**查询出账务名称不为null账务信息**

**SELECT \* FROM zhangwu WHERE name IS NOT NULL;**

**SELECT \* FROM zhangwu WHERE NOT (name IS NULL);**

**数据库中的排序 \* ASC 升序 (默认) \* DESC 降序**

# Android下数据库创建

**什么情况下我们才用数据库做数据存储？ 大量数据结构相同的数据需要存储时。**

**sqlite 嵌入式 轻量级**

**SqliteOpenHelper**

**创建数据库步骤：**

**1.创建一个类集成SqliteOpenHelper，需要添加一个构造方法，实现两个方法oncreate ,onupgrade**

**构造方法中的参数介绍：**

**context :上下文 ， name：数据库文件的名称 factory：用来创建cursor对象，默认为null**

**version:数据库的版本号，从1开始，如果发生改变，onUpgrade方法将会调用,4.0之后只能升不能将**

**super(context, "info.db", null,1);**

**2.创建这个帮助类的一个对象，调用getReadableDatabase()方法，会帮助我们创建打开一个数据库**

**3.复写oncreate和onupgrdate方法：**

**oncreate方法是数据库第一次创建的时候会被调用; 特别适合做表结构的初始化,需要执行sql语句；SQLiteDatabase db可以用来执行sql语句**

**onUpgrade数据库版本号发生改变时才会执行； 特别适合做表结构的修改**

**帮助类对象中的getWritableDatabase 和 getReadableDatabase都可以帮助我们获取一个数据库操作对象SqliteDatabase.**

**区别：**

**getReadableDatabase:**

**先尝试以读写方式打开数据库，如果磁盘空间满了，他会重新尝试以只读方式打开数据库。**

**getWritableDatabase:**

**直接以读写方式打开数据库，如果磁盘空间满了，就直接报错。**

## Android下数据库第一种方式增删改查

**1.创建一个帮助类的对象继承SQLiteOpenHelper，调用getReadableDatabase方法，返回一个SqliteDatebase对象**

**2.使用SqliteDatebase对象调用execSql()做增删改,调用rawQuery方法做查询。**

**特点:增删改没有返回值，不能判断sql语句是否执行成功。sql语句手动写，容易写错**

**增加数据 删除数据 修改数据 都调用SQLiteDatabase 中的execSQL方法 没有返回值**

**执行sql语句需要sqliteDatabase对象**

**调用getReadableDatabase方法,来初始化数据库的创建**

**SQLiteDatabase db = mySqliteOpenHelper.getReadableDatabase();**

**sql:sql语句， bindArgs：sql语句中占位符的值**

**db.execSQL("insert into info(name,phone) values(?,?);", new Object[]{bean.name,bean.phone});**

**查询数据 调用SQLiteDatabase 中的 rawQuery 方法 查询方法灵活**

**sql:sql语句， selectionArgs：sql语句中查询条件占位符的值**

**rawQuery 方法返回值是一个cursor结果集**

**1. 判断cursor 是否为null 并且cursor 中是否有对象**

**if (cursor != null && cursor.getCount() > 0)**

**2. 循环判断 cursor 条件 游标是否定位到下一行**

**while (cursor.moveToNext())**

**3. 通过以下方法解析cursor 获取数据**

**[getInt](https://developer.android.com/reference/android/database/Cursor.html" \l "getInt(int))(int columnIndex) 返回值 int类型 参数传递索引值**

**[getString](https://developer.android.com/reference/android/database/Cursor.html" \l "getString(int))(int columnIndex) 返回值 String类型 参数传递索引值**

## Android下另外一种增删改查方式

**1.创建一个帮助类的对象继承SQLiteOpenHelper，调用getReadableDatabase方法，返回一个****SqliteDatebase对象**

**2.使用SqliteDatebase对象调用insert,update,delete ,query方法做增删改查。**

**特点:增删改有了返回值，可以判断sql语句是否执行成功，但是查询不够灵活，不能做多表查询。所以在公司一般人增删改喜欢用第二种方式，查询用第一种方式。**

**增加数据 SqliteDatebase 中的增加方法**

**[insert](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "insert(java.lang.String, java.lang.String, android.content.ContentValues))([String](https://developer.android.com/reference/java/lang/String.html) table, [String](https://developer.android.com/reference/java/lang/String.html) nullColumnHack, [ContentValues](https://developer.android.com/reference/android/content/ContentValues.html) values) 返回值 long类型**

**table: 表名 , nullColumnHack：可以为空，标示添加一个空行,**

**values:数据一行的值 , 返回值：代表添加这个新行的Id ，-1代表添加失败**

**例如：**

**创建数据库**

**SQLiteDatabase db = database.getReadableDatabase();**

**ContentValues 是用map封装的对象，用来存放值**

**ContentValues values = new ContentValues();**

**values.put("name", bean.name);**

**values.put("phone", bean.phone);**

**table: 表名 , nullColumnHack：可以为空，标示添加一个空行,**

**values:数据一行的值 , 返回值：代表添加这个新行的Id ，-1代表添加失败**

**long result = db.insert("perInFo", null, values);**

**册除数据**

**[delete](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "delete(java.lang.String, java.lang.String, java.lang.String[]))([String](https://developer.android.com/reference/java/lang/String.html) table, [String](https://developer.android.com/reference/java/lang/String.html) whereClause, [String[]](https://developer.android.com/reference/java/lang/String.html) whereArgs) 返回值 int类型**

**table ：表名, whereClause: 删除条件, whereArgs：条件的占位符的参数 ; 返回值：成功删除多少行**

**例如：**

**int result = db.delete("perInFo", "name = ?", new String[]{name});**

**修改数据**

**[update](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "update(java.lang.String, android.content.ContentValues, java.lang.String, java.lang.String[]))([String](https://developer.android.com/reference/java/lang/String.html) table, [ContentValues](https://developer.android.com/reference/android/content/ContentValues.html) values, [String](https://developer.android.com/reference/java/lang/String.html) whereClause, [String[]](https://developer.android.com/reference/java/lang/String.html) whereArgs) 返回值 int类型**

**table:表名, values：更新的值, whereClause:更新的条件,**

**whereArgs：更新条件占位符的值,返回值：成功修改多少行**

**例如：**

**int result = db.update("perInFo", values, "phone=?",new String[] { bean.phone });**

**查询数据(开发中一般用** rawQuery**)**

**[query](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "query(java.lang.String, java.lang.String[], java.lang.String, java.lang.String[], java.lang.String, java.lang.String, java.lang.String))([String](https://developer.android.com/reference/java/lang/String.html) table, [String[]](https://developer.android.com/reference/java/lang/String.html) columns, [String](https://developer.android.com/reference/java/lang/String.html) selection, [String[]](https://developer.android.com/reference/java/lang/String.html) selectionArgs, [String](https://developer.android.com/reference/java/lang/String.html) groupBy,[String](https://developer.android.com/reference/java/lang/String.html) having, [String](https://developer.android.com/reference/java/lang/String.html) orderBy)**

**table:表名, columns：查询的列名,如果null代表查询所有列； selection:查询条件,**

**selectionArgs：条件占位符的参数值, groupBy:按什么字段分组, having:分组的条件, orderBy:按什么字段排序**

**Cursor cursor = db.query("perInFo",new String[] { "\_id", "name", "phone" }, "name = ?",**

**new String[] { name }, null, null, "\_id desc");**

**if (cursor != null && cursor.getCount() > 0) {**

**while (cursor.moveToNext()) {**

**int id = cursor.getInt(0);**

**String name\_str = cursor.getString(1);**

**cursor.getString(cursor.getColumnIndex("money"))**

**String phone = cursor.getString(2);**

**注意:**

**1. contentValues该类底层是Map数据结构 values.put(key,values): key 对应数据库表中字段 value是想插入的值**

**2. tableName 是表名，不是数据库名**

**3.使用db进行增删改查的时候注意用完进行db.close()释放资源**

**4.cursor对象是数据库查询完毕后封装的一个结果集，可以通过moveToNext判断是否还有下一个可获取的值，通过cursor.getString(index)获取对应的值；**

**这里的index对应的是查询的时候查询的哪一列：0对应columName1；1对应columNmae2……**

### 数据库的事物

**事务： 执行多条sql语句，要么同时执行成功，要么同时执行失败，不能有的成功，有的失败**

**SQLiteDatabase 提供了对事务的支持，处于事务中的操作都是“临时性”的，只有事务提交了才会将数据保存到数据库。**

**事务的使用不仅可以保证数据的一致性，也可以提高批处理时的执行效率。SQLiteDatabase 提供的beginTransaction（）打开事务，endTransaction（）结束事务。**

**注意：结束事务并不代表事务提交，如果想让数据写入数据库需要在结束事务前执行setTransactionSuccessful（）方法。这是事务提交的唯一方式。**

**例：**

**//点击按钮执行该方法**

**public void transtation(View v){**

**//1.创建一个帮助类的对象**

**BankOpenHelper bankOpenHelper = new BankOpenHelper(this);**

**//2.调用数据库帮助类对象的getReadableDatabase创建数据库，初始化表数据，获取一个SqliteDatabase对象去做转账（sql语句）**

**SQLiteDatabase db = bankOpenHelper.getReadableDatabase();**

**//3.转账,将李四的钱减200，张三加200**

**db.beginTransaction();//开启一个数据库事务**

**try {**

**db.execSQL("update account set money= money-200 where name=?",new String[]{"李四"});**

**int i = 100/0;//模拟一个异常**

**db.execSQL("update account set money= money+200 where name=?",new String[]{"张三"});**

**db.setTransactionSuccessful();//标记事务中的sql语句全部成功执行**

**} finally {**

**db.endTransaction();//判断事务的标记是否成功，如果不成功，回滚错误之前执行的sql语句**

**}**

**}**

# listview 入门

**ListView 是一个控件,一个在垂直滚动的列表中显示条目的一个控件，**

**这些条目的内容来自于一个ListAdapter 。EditText Button TextView ImageView Checkbox 五大布局。**

**1.布局添加Listview**

**2.找到listview**

**3.创建一个Adapter适配器继承BaseAdapter，实现4个方法，其中getcount,getview必须封装**

**getcount:告诉listview要显示的条目数**

**getItem根据postion获取listview上条目对应的数据**

**getItemId:用来获取条目postion行的id**

**getview：告诉listview条目上显示的内容；返回一个View对象作为条目上的内容展示，该方法返回什么样的view,**

**Listview的条目上就显示什么样的view**

**4.创建Adapter的一个对象，设置给listview。**

**listview.setAdapter(ListAdapter adapter);**

**适配器作用：1.设置列表项布局　2.装配列表项的布局**

## listview优化

**简单优化 开发中不使用**

**adapter中getview方法会传进来一个convertView，convertView是指曾经使用过的view对象，可以被重复使用，**

**但是在使用前需要判断是否为空，不为空直接复用，并作为getview方法的返回对象。**

**TextView view = null;**

**if(convertView != null){//判断converView是否为空，不为空重新使用**

**view = (TextView) convertView;**

**}else{**

**view = new TextView(mContext);//创建一个textView对象**

**}**

**return view；**

**listview显示原理**

**1.要考虑listview显示的条目数 getcount**

**2.考虑listview每个条目显示的内容 getview**

**3.考虑每个item的高度，因为屏幕的多样化**

**4.还要考虑listview的滑动，监听一个旧的条目消失，一个新的条目显示。**

**5.- 通过convertView :可回收的View来优化适配器**

**- 通过定义一个HolderView ，来保存引用要操作的控件的地址，取代findViewById查找控件**

**以空间换时间**

**Listview 基于 mvc**

**m....mode....Bean**

**v....view....listview**

**c....control...adapter**

**一般开发中listview宽高属性都设置成填充父窗体 match\_parent**

**因为如果是包裹内容(wrap\_content) listview 需要不停调整条目数据 来适配屏幕**

**ListView适配多个列表项布局**

**子控件设置数据**

**holderView.holdMessageTv.setText("Android1: "+chatMessage.message); //设置数据**

**自定义控件的作用： 操纵列表项的子控件的属性 ，操纵列表项要填充的数据**

**holderView.holdMessageTv.setTextColor(Color.RED);**

**- 准备两个列表项布局**

**- getViewTypeCount ：该listView要适配多少种布局**

**- getItemViewType ：取得当前要展示的列表项的布局类型**

**if(getItemViewType(position)==0){**

**convertView=LayoutInflater.from(context).inflate(R.layout.chatting\_item\_right, parent,false);**

**}else{**

**convertView=LayoutInflater.from(context).inflate(R.layout.chatting\_item\_left, parent,false);**

**}**

**取得当前要展示的列表项的布局类型 返回的值 为 （0---类型总数-1）**

**@Override**

**public int getItemViewType(int position) {**

**return mData.get(position).isSelf?0:1; //假如信息是自己发的，则采用第0中布局类型，否则采用第一种布局类型**

**}**

**取得列表项视图类型的数量 2种列表项布局**

**@Override**

**public int getViewTypeCount() {**

**return 2;**

**}**

**两个布局中的控件ID需设置一样**

## 复杂listview界面显示

**1.布局写listview**

**2.找到listview**

**3.获取新闻数据封装到list集合中(才用模拟数据)，作为adapter的显示数据,怎么将获取的新闻数据给adapter**

**通过context对象将一个资源id转换成一个Drawable对象。**

**newsBean.icon = context.getResources().getDrawable(R.drawable.*ic\_launcher*);**

**4.创建一个adapter继承BaseAdapter，实现4个方法**

**getcount: 有多少条新闻数据，就有多少个条目。**

**getView:将返回一个复杂的布局作为条目的内容展示；并且显示的数据是新闻的信息。**

**public View getView(int position, View convertView, ViewGroup parent) {**

**ViewHolder viewHolder = null;**

**if (convertView == null) {**

**convertView= View.*inflate*(context, R.layout.*item\_news*, null);**

**viewHolder= new ViewHolder();**

**//与ViewHolder关联**

**viewHolder.iv\_item\_icon =(SmartImageView) convertView.findViewById(R.id.*iv\_item\_icon*);**

**viewHolder.tv\_title=(TextView) convertView.findViewById(R.id.*tv\_title*);**

**viewHolder.tv\_des=(TextView) convertView.findViewById(R.id.*tv\_des*);**

**viewHolder.tv\_comment=(TextView) convertView.findViewById(R.id.*tv\_comment*);**

**viewHolder.tv\_type=(TextView) convertView.findViewById(R.id.*tv\_type*);**

**//设置View的tag属性，把HolderView添加到tag中**

**convertView.setTag(viewHolder);**

**}else {**

**//复用convertView**

**viewHolder = (ViewHolder) convertView.getTag();**

**}**

**//创建HolderView 记录控件地址**

**public static class ViewHolder{**

**SmartImageView iv\_item\_icon;**

**TextView tv\_title;**

**TextView tv\_des;**

**TextView tv\_comment;**

**TextView tv\_type;**

**}**

**5.创建一个adapter对象设置给listview**

**6.设置listview的条目的点击事件，并封装点击事件,去查看新闻详情。**

**//****设置listview条目的点击事件**

**lv\_news.setOnItemClickListener(this);**

**//listview的条目点击时会调用该方法 parent:代表listviw view:点击的条目上的那个view对象 position:条目的位置 id： 条目的id**

**public void onItemClick(AdapterView<?> parent, View view, int position,**

**long id) {**

**//需要获取条目上bean对象中url做跳转**

**NewsBean bean = (NewsBean) parent.getItemAtPosition(position);**

**String url = bean.news\_url;**

**//跳转浏览器**

**Intent intent = new Intent();**

**intent.setAction(Intent.ACTION\_VIEW);**

**intent.setData(Uri.parse(url));**

**startActivity(intent);**

**}**

**listview界面显示思路：**

**1.布局写listview ok**

**2.找到listview ok**

**3.封装新闻数据到list集合中 ，目的是为adapter提供数据展示。**

**4.封装一个Adapter类继承BaseAdatper，写一个构造方法接受list集合数据，复写四个方法**

**a.创建一个构造方法**

**b.封装getCount方法**

**c.getView方法：**

**1.复用convertview，模板代码,如果不都能空，需要将一个布局文件转换为view对象作为getview的返回对象。**

**view = View.inflater(Context context, int resuorceId,ViewGroup root)**

**2.找到view上的这些子控件，目的是将list集合中的bean数据一一对应设置给这些子控件**

**3.从list集合中获取postion条目上要显示的数据Bean**

**4.将获取的bean中的数据设置给这些子控件**

**d.getItem方法：将list集合中指定postion上的bean对象返回**

**e.getItemId,直接返回postion**

**5.创建一个封装的Adapter对象，设置给listview**

**6.设置listview条目的点击事件**

**listview.setOnItem....**

**7.复写OnItemClicklistener方法，获取相应条目上的bean对象，最终获取到url，做Intent跳转;**

## 常用获取inflate的写法

**1.**

**//context:上下文, resource:要转换成view对象的layout的id, root:将layout用root(ViewGroup)包一层作为codify的返回值,一般传null**

**//view = View.inflate(context, R.layout.item\_news\_layout, null);//将一个布局文件转换成一个view对象**

**2.**

**//通过LayoutInflater将布局转换成view对象**

**//view = LayoutInflater.from(context).inflate(R.layout.item\_news\_layout, null);**

**3.**

**//通过context获取系统服务得到一个LayoutInflater，通过LayoutInflater将一个布局转换为view对象**

**LayoutInflater layoutInflater = (LayoutInflater) context.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);**

**view = layoutInflater.inflate(R.layout.item\_news\_layout, null);**

**arrayadapter (不用看，知道有这个玩意就行)**

**//找到控件**

**ListView lv\_array = (ListView) findViewById(R.id.lv\_array);**

**ListView lv\_simple = (ListView) findViewById(R.id.lv\_simple);**

**//创建一个arrayAdapter**

**//context , resource:布局id, textViewResourceId：条目布局中 textview控件的id, objects:条目上texitview显示的内容**

**ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(this, R.layout.item\_listview\_layout, R.id.item\_tv\_class, classz);**

**lv\_array.setAdapter(arrayAdapter);**

**simpleadapter (不用看，知道有这个玩意就行)**

**//创建一个simpleAdapter,封装simpleAdapter的数据**

**ArrayList<Map<String, String>> arrayList = new ArrayList<Map<String,String>>();**

**HashMap<String, String> hashMap = new HashMap<String, String>();**

**hashMap.put("class", "C++");**

**arrayList.add(hashMap);**

**HashMap<String, String> hashMap1 = new HashMap<String, String>();**

**hashMap1.put("class", "android");**

**arrayList.add(hashMap1);**

**HashMap<String, String> hashMap2 = new HashMap<String, String>();**

**hashMap2.put("class", "javaEE");**

**arrayList.add(hashMap2);**

**//context, data:显示的数据, resource:item布局id, from: map中的key, to:布局中的控件id**

**SimpleAdapter simpleAdapter = new SimpleAdapter(this, arrayList, R.layout.item\_listview\_layout, new String[]{"class"}, new int[]{R.id.item\_tv\_class});**

**lv\_simple.setAdapter(simpleAdapter);**

# HTTP请求

**一个完整的HTTP请求，分为三部份：**

**1\_请求行**

**请求方式 请求内容 协议名和版本**

**2\_请求头**

**有很多key:value组成的内容，每次请求都不相同**

**Accept: text/html,image/\* 浏览器可以接收什么类型的响应数据**

**Accept-Charset: GBK 浏览器可以接收什么类型的编码方式**

**Accept-Encoding: gzip,compress 浏览器可以接收gzip和compress格式的压缩数据，即浏览器收到这些压缩数据后会自动解压**

**Accept-Language: en-us,zh-cn 浏览器可以接收什么类型的语言**

**Host: www.itheima.com:80 浏览器请求的主机是www.itheima.com，使用80端口号**

**If-Modified-Since: Tue, 11 Jul 2014 18:23:51 GMT(缓存时间) 浏览器收到数据后，可缓存时间**

**Referer: http://www.itheima.com/index.html(来自哪) 请求来源于哪个URL地址**

**User-Agent:Mozilla/4.0 (compatible; MSIE 9; Windows NT 5.0) 浏览器的类型和版本**

**Cookie：后面细讲 浏览器缓存服务器响应的信息**

**Connection: close/Keep-Alive 浏览器与服务器的连接是打开还是关闭**

**Date: Tue, 11 Jul 2014 18:23:51 GMT(访问时间) 浏览器请求服务器的时间，这个时间不是中国时间**

**3\_请求体或实体内容**

**不是每次请求都有内容的，**

**GET方式请求体是没有内容的，因为内容都在请求行中**

**POST方式请求体是有内容的，例如：文件上传，表单以POST方式提交**

# HTTP响应

**一个完整的HTTP响应，分为三部份：**

**1\_响应行**

**协议名和版本 响应状态码 响应状态码的英文描述**

**2\_响应头**

**有很多key:value组成的内容，每次响应都不相同**

**Location: http://www.itheima/index.html 服务器要求浏览器访问的URL地址**

**Server:apache tomcat 服务器通知浏览器服务器的名字**

**Content-Encoding: gzip 服务器通知浏览器需要接收的压缩数据类型**

**Content-Length: 80 服务器通知浏览器需要接收的响应内容的字节数**

**Content-Language: zh-cn 服务器通知浏览器需要接收的语言类型**

**Content-Type: text/html; charset=GBK 服务器通知浏览器需要接收的类型和使用什么方式解码**

**Last-Modified: Tue, 11 Jul 2014 18:23:51 GMT 服务器通知浏览器浏览器访问的请求最近一次修改的时间**

**Refresh: 1; url=http://www.itheima.com 服务器通知浏览器1秒钟后刷新，并且访问指定的URL页面**

**Content-Disposition: attachment; filename=aaa.zip(下载文件) 服务器通知浏览器以下载方式打开资源**

**Transfer-Encoding: chunked(分块传递数据到客户端）服务器通知浏览器以分块方式下载文件的浏览器指定的目录**

**Set-Cookie:SS=Q0=5Lb\_nQ; path=/search 服务器通知浏览器需要接收数据缓存到浏览器**

**Expires: -1//3种禁止缓存的头字段 以下三个响应头都表示服务器要求浏览器不要缓存来自服务器的web页面**

**Cache-Control: no-cache**

**Pragma: no-cache**

**Connection: close/Keep-Alive 服务器通知浏览器之间的连接是否已以关闭或者打开的**

**Date: Tue, 11 Jul 2014 18:23:51 GM 服务器通知浏览器的时间**

**如果上述头中，请求也有，响应也用，我们叫其通用头,Connection和Date就是通用头**

**3\_响应体或实体内容**

**响应体和响应头之间有一个空行**

**响应状态码是服务器对这次响应设置的一人唯一编号，**

**每个编码都有其特定的含义，常见的响应编号有：**

**200：表示服务器响应正确**

**206：部分请求成功**

**302：客户端请求一台服务端的资源，该服务端并没有这个资源，服务器要求客户端自已去另一台服务端找资源，这种情况下叫做重定向**

**307：客户端请求一台服务端的资源，该服务端并没有这个资源，服务器自已去另一台服务端找资源，这种情况下叫做转发**

**304：客户端请求服务端的资源，服务器没有修改过，且已经缓存到了客户端，要求客户端去其缓存中获取即可**

**404：客户端请求服务端的URL出错了**

**500：客户端请求服务端的URL正确，但服务器处理资源出错了**

## Tomcat的目录结构

****

**bin：脚本目录**

**启动脚本：startup.bat**

**停止脚本：shutdown.bat**

**conf：配置文件目录 (config /configuration)**

**核心配置文件：server.xml**

**用户权限配置文件：tomcat-users.xml**

**所有web项目默认配置文件：web.xml**

**lib：依赖库，tomcat和web项目中需要使用的jar包**

**logs：日志文件.**

**localhost\_access\_log.\*.txt tomcat记录用户访问信息，星\*表示时间。**

**例如：localhost\_access\_log.2016-02-28.txt**

**temp：临时文件目录，文件夹内内容可以任意删除。**

**webapps：默认情况下发布WEB项目所存放的目录。**

**work：tomcat处理JSP的工作目录。**

**Tomcate的启动与运行**

**双击Tomcat下的bin下的startup.bat启动Tomcat**

# RecyclerView的使用

# 网络html源码查看器

**//通过webView看网页**

**pageWebView.loadUrl("http://192.168.32.9:8080/web/index.html");**

**pageWebView.setVisibility(View.VISIBLE);**

**codeTextView.setVisibility(View.GONE);**

**访问网络需要加Internet权限：**

**android.permission.INTERNET**

**使用UrlConnection请求一个url地址获取内容：**

**//1.创建一个Url对象**

**URL url = new URL(url\_str);**

**//2.获取一个UrlConnection对象**

**HttpURLConnection connection = (HttpURLConnection)url.openConnection();**

**//3.为UrlConnection对象设置一些请求的参数,请求方式，连接的超时时间**

**connection.setRequestMethod("GET");//设置请求方式**

**connection.setConnectTimeout(1000\*10);//设置超时时间**

**//4.在获取url请求的数据前需要判断响应码，200 ：成功,206:访问部分数据成功 300：跳转或重定向 400：错误 500：服务器异常**

**int code = connection.getResponseCode();**

**if(code == 200){**

**//5.获取有效数据，并将获取的流数据解析成String**

**InputStream inputStream = connection.getInputStream();**

**String result = StreamUtils.streamToString(inputStream);**

**注意事项：**

**1. ANR:application not response 应用无响应; 因为在androoid中耗时的操作（请求网络，大文件的拷贝，数据库的操作）需要在子线程中做。**

**09-02 01:52:40.711: E/ActivityManager(857): ANR in com.itheima.sourcelook (com.itheima.sourcelook/.MainActivity)**

**2. 4.0后网络操作强制在子线程中进行。因为网络访问是耗时的操作，可能会导致ANR异常**

**09-02 01:57:32.879: W/System.err(1789): android.os.NetworkOnMainThreadException**

**3.错误线程调用异常，子线程不能够更新UI(控件的内容)**

**W/System.err(1858): android.view.ViewRootImpl$CalledFromWrongThreadException: Only the original thread that created a view hierarchy can touch its views.**

**4 .Caused by: libcore.io.ErrnoException: getaddrinfo failed: EACCES (Permission denied) 没有加权限异常**

**主线程不能够做耗时的操作，网络请求就是耗时的操作需要放到子线程做。子线程不能更新控件的内容(更新Ui)。所以产生了矛盾，解决办法就是使用Handler.**

## 消息机制的写法（重要） Handler

**使用Handler的步骤：**

**1.主线程中创建一个Handler**

**private Handler handler = new Handler(){**

**public void handleMessage(android.os.Message msg) {**

**};**

**};**

**2.重写handler的handlermessage方法**

**3.子线程中创建一个Message对象，将获取的数据绑定给msg**

**Message msg = new Message();**

**//另一种方式：Message msg = Messge.obtain;**

**msg.obj = result;**

**4.主线程中的handler对象在子线程中将message发送给主线程**

**handler.sendMessage(msg);**

**5.主线程中handlermessage方法接受子线程发来的数据，就可以做更新UI的操作。**

**\*\*\*\*\*\*\*\*\*\*\*主线程**

**//☆☆☆1.在主线程中创建一个Handler对象**

**private Handler handler = new Handler(){**

**//☆☆☆2.重写handler的handlermessage方法,用来接收子线程中发来的消息**

**public void handleMessage(android.os.Message msg) {**

**//☆☆☆5.接收子线程发送的数据，处理数据。**

**Bitmap bitmap = (Bitmap) msg.obj;**

**//☆☆☆6.当前方法属于主线程可以做UI的更新**

**//五.获取服务器返回的内容，显示到textview上**

**img\_pic.setImageBitmap(bitmap);//设置ImageView的图片内容**

**};**

**};**

**\*\*\*\*\*\*\*\*\*\*\*\*子线程**

**if(code == 200){**

**//5.获取有效数据，并将获取的流数据解析成String**

**InputStream inputStream = connection.getInputStream();**

**//将一个读取流转换成一个图片 Drawable , Btimap:位图**

**Bitmap bitmap = BitmapFactory.decodeStream(inputStream);**

**//☆☆☆3.子线中创建一个Message对象，为了携带子线程中获取的数据给主线程。**

**Message msg = Message.obtain();//获取一个Message对象，内部实现是：如果之前的Message存在直接返回，不存在创建新的Message返回**

**msg.obj = bitmap;//将获取的数据封装到msg中。**

**//☆☆☆4.使用handler对象将message发送到主线程。**

**handler.sendMessage(msg);**

**}**

# 消息机制原理（重要）

**有几个主要元素：**

**1.Message:用来携带子线程中的数据。**

**2.MessageQueue:用来存放所有子线程发来的Message.**

**3.Handler:用来在子线程中发送Message，在主线程中接受Message，处理结果**

**4.Looper:是一个消息循环器，一直循环遍历MessageQueue，从MessageQueue中取一个Message，派发给Handler处理。**

## 网络图片查看器

**adb shell+ input text 内容；可以通过将内容输入到手机上的输入框。**

**将一个读取流转换成bitmap对象：位图**

**BitmapFactory：可以将文件，读取流，字节数组转换成一个Bitmap对象。**

**Bitmap bitmap = BitmapFactory.decodeStream(InputStream in);**

**imageView.setImageBitmap(bitmap);//设置图片内容**

## 常见消息处理api（重要）

**面试：子线程一定不能更新UI？ SurfaceView ：多媒体视频播放 ,可以在子线程中更新UI；**

**Progress（进度）相关的控件：也是可以在子线程中更新Ui;**

**审计机制：activity完全显示的时候审计机制才会去检测子线程有没有更新Ui.**

**当Aictivity 执行onResume 方法的时候，界面完全显示出来，才能去检测子线程有没有更新Ui.**

**1.使用activity的runOnUiThread方法更新ui,无论当前线程是否是主线程，都将在主线程执行.**

**runOnUiThread(new Runnable() {**

**@Override**

**public void run() {**

**tv\_simple.setText("我被更新了");**

**}**

**});**

**2.使用handler直接post到主线程，handler需要在主线程创建**

**//延迟多少毫米执行runnable。**

**mHandler.postDelayed(new Runnable() {**

**@Override**

**public void run() {**

**tv\_simple.setText("我被更新了");**

**}**

**}, 1000\*5);**

**应用场景:广告展示后，做页面跳转。**

# 新闻客户端案例(重要)项目和产品

**第一次进入新闻客户端需要请求服务器获取新闻数据，做listview的展示，**

**为了第二次再次打开新闻客户端时能快速显示新闻，需要将数据缓存到数据库中，下次打开可以直接去数据库中获取新闻直接做展示。**

**1.写布局listview ok**

**2.找到listview，设置条目的点击事件。 ok**

**3.获取数据提供给listview做展示。**

**3.1：获取本地数据库缓存的新闻数据,让listview显示。如果没有网络不至于显示空界面。**

**3.2：请求服务器获取新闻数据，是一个json字符串，需要解析json,封装到list集合中。提供给listview展示。**

**public static String newsPath\_url = "http://192.168.13.83:8080/itheima74/servlet/GetNewsServlet";**

**//封装新闻的假数据到list中返回**

**public static ArrayList<NewsBean> getAllNewsForNetWork(Context context) {**

**ArrayList<NewsBean> arrayList = new ArrayList<NewsBean>();**

**try{**

**//1.请求服务器获取新闻数据**

**//获取一个url对象，通过url对象得到一个urlconnnection对象**

**URL url = new URL(newsPath\_url);**

**HttpURLConnection connection = (HttpURLConnection) url.openConnection();**

**//设置连接的方式和超时时间**

**connection.setRequestMethod("GET");**

**connection.setConnectTimeout(10\*1000);**

**//获取请求响应码**

**int code = connection.getResponseCode();**

**if(code == 200){**

**//获取请求到的流信息**

**InputStream inputStream = connection.getInputStream();**

**String result = StreamUtils.streamToString(inputStream);**

**//2.解析获取的新闻数据到List集合中。**

**JSONObject root\_json = new JSONObject(result);//将一个字符串封装成一个json对象。**

**JSONArray jsonArray = root\_json.getJSONArray("newss");//获取root\_json中的newss作为jsonArray对象**

**for (int i = 0 ;i < jsonArray.length();i++){//循环遍历jsonArray**

**JSONObject news\_json = jsonArray.getJSONObject(i);//获取一条新闻的json**

**NewsBean newsBean = new NewsBean();**

**newsBean. id = news\_json.getInt("id");**

**newsBean. comment = news\_json.getInt("comment");//评论数**

**newsBean. type = news\_json.getInt("type");//新闻的类型，0 ：头条 1 ：娱乐 2.体育**

**newsBean. time = news\_json.getString("time");**

**newsBean. des = news\_json.getString("des");**

**newsBean. title = news\_json.getString("title");**

**newsBean. news\_url = news\_json.getString("news\_url");**

**newsBean. icon\_url = news\_json.getString("icon\_url");**

**arrayList.add(newsBean);**

**}**

**//3.清楚数据库中旧的数据，将新的数据缓存到数据库中**

**new NewsDaoUtils(context).delete();**

**new NewsDaoUtils(context).saveNews(arrayList);**

**}**

**}catch (Exception e) {**

**e.printStackTrace();**

**}**

**return arrayList;**

**}**

**3.3： 获取服务端数据成功后，需要缓存到本地数据库，缓存前需要清空本地数据库。**

**4.创建一个Adapter继承BaseAdapter,封装4个方法，需要接收获取的新闻数据**

**5.将adapter设置给listview。**

**<http://192.168.160.2:8080/itheima74/servlet/GetNewsServlet>**

**缓存到本地文件中**

**通过Context.getExternalCacheDir()方法可以获取到mnt/ SdCard/Android/data/你的应用包名/cache/目录，一般存放临时缓存数据**

**private Handler handler = new Handler() {**

**public void handleMessage(android.os.Message msg) {**

**switch (msg.what) {**

**case *MSG\_WHAT\_NEWS*:**

**String result = (String) msg.obj;**

**ArrayList<newBean> list = ParseHttpUtils.*parseNewsjson*(result);**

**if (list != null && list.size() > 0) {**

**setListAdapter(list);**

**// 将数据写入到本地(缓存到本地)**

**FileUtils.*saveNewsToExternalCacheDir*(*mContext*, result);**

**}**

**break;**

**}**

**};**

**};**

**private NewsAdapter newsAdapter;**

**private void setListAdapter(ArrayList<newBean> list){**

**if(adapter == null){**

**adapter = new NesAdapter(list, *mContext*);**

**lv\_news.setAdapter(adapter);**

**}else {**

**adapter.setNewData(list);**

**//通知列表已经改变　刷新列表 不需要重新创建对象**

**adapter.notifyDataSetChanged();**

**}**

**}**

**@Override**

**protected void onCreate(Bundle savedInstanceState) {**

**super.onCreate(savedInstanceState);**

**setContentView(R.layout.*activity\_main*);**

***mContext* = this;**

**lv\_news = (ListView) findViewById(R.id.*lv\_news*);**

**lv\_news.setOnItemClickListener(this);**

**// 读取本地数据 显示到listview中**

**ArrayList<newBean> arrayList = *parseNewsForFile*();**

**if (arrayList != null && arrayList.size() > 0) {**

**setListAdapter(arrayList);**

**}**

**new Thread(new Runnable() {**

**@Override**

**public void run() {**

**try {**

**Thread.*sleep*(1000 \* 5);**

**} catch (InterruptedException e) {**

**e.printStackTrace();**

**}**

**// 1、获取 网络数据 将数据发送到Handler**

**String result = NewsInFo.*getNewsFromServe*();**

**Message msg = Message.*obtain*();**

**msg.what = *MSG\_WHAT\_NEWS*;**

**msg.obj = result;**

**handler.sendMessage(msg);**

**}**

**}).start();**

**}**

## smartimageview 显示(开源的小项目)

**自定义的控件在布局文件中的引用都需要指定类的完整路径**

**1.自定义了一个MyImageview类继承了Imageview，添加三个构造方法**

**2.添加一个setImageUrl方法接受一个图片url**

**3.新建一个子线程去请求url获取图片资源**

**4.将获取的图片Bitmap通过handler发送给主线程,主线程设置给当前view.**

**5.在布局中引用MyImageview，需要指定完成的包名路径。**

# get post方式提交数据到服务器

**get方式和post方式的区别：**

**1.请求的URL地址不同：**

**post："http://192.168.13.83:8080/itheima74/servlet/LoginServlet"**

**get：http://192.168.13.83:8080/itheima74/servlet/LoginServlet?username=root&pwd=123**

**2.请求头不同：**

**\*\*\*\*post方式多了几个请求头：Content-Length Cache-Control Origin**

**openConnection.setRequestProperty("Content-Length", body.length()+"");**

**openConnection.setRequestProperty("Cache-Control", "max-age=0");**

**openConnection.setRequestProperty("Origin", "http://192.168.13.83:8080");**

**\*\*\*\*post方式还多了请求的内容：username=root&pwd=123**

**//设置UrlConnection可以写请求的内容**

**openConnection.setDoOutput(true);**

**//获取一个outputstream，并将内容写入该流**

**openConnection.getOutputStream().write(body.getBytes());**

**3.请求时携带的内容大小不同**

**get:1k**

**post:理论无限制**

**查询数据一般用get 提交数据一般用post**

**总结：**

**在http 协议中，get 请求协议没有请求体，只有请求头和请求行，因此如果想使用get 方式提交数据，只能通过在url 后面加上要提交的参数。这也意味着get 方式只能提交比较小的参数。**

**如果get 方式提交的参数有特殊字符或者中文字符那么必须对其进行URL 编码，不然会导致服务器接收到的数据乱码。**

**对于post 请求，提交的数据位于请求体中，因此需要设置connection.setDoOutput(true);参数，该参数设置后才允许将提交的数据通过输出流的形式提交。不过需要说明的是虽然采用post 方式提交，依然需要将参数进行URL 编码。**

**其实当我们使用浏览器进行form 表单提交的时候，浏览器内部已经帮我们实现了URL 编码。因为我们这里是使用代码模拟浏览器的提交数据过程，因此需要使用代码特殊处理。**

## post方式提交数据乱码的解决

**一般在公司开发客户端和服务端的编码要保持一致。**

**android端的默认编码是utf-8;**

**做url请求时需要对参数进行URLEncode编码.**

**URL url = new URL("http://192.168.13.83:8080/itheima74/servlet/LoginServlet?username="+URLEncoder.encode(username)+"&pwd="+URLEncoder.encode(password));**

**connection.setDoOutput(true);**

**connection.getOutputStream().write(parmes.getBytes());**

**# 2 get方式提交数据乱码解决**

**URLEncode**

**# 3 httpclient方式提交数据到服务器**

**HttpClient:**

**get方式:**

**//使用HttpClient请求服务器将用户密码发送服务器验证**

**try{**

**String path = "http://192.168.13.83:8080/itheima74/servlet/LoginServlet?username="+URLEncoder.encode(username,"utf-8")+"&pwd="+URLEncoder.encode(password,"utf-8");**

**//1.创建一个httpClient对象**

**HttpClient httpclient = new DefaultHttpClient();**

**//2.设置请求的方式**

**HttpGet httpget = new HttpGet(path);**

**//3.执行一个http请求**

**HttpResponse response = httpclient.execute(httpget);**

**//4.获取请求的状态码，**

**StatusLine statusLine = response.getStatusLine();**

**int code = statusLine.getStatusCode();**

**//5.判断状态码后获取内容**

**if(code == 200){**

**HttpEntity entity = response.getEntity();//获取实体内容，中封装的有http请求返回的流信息**

**InputStream inputStream = entity.getContent();**

**//将流信息转换成字符串**

**String result = StreamUtils.streamToString(inputStream);**

**Message msg = Message.obtain();**

**msg.what = 1;**

**msg.obj = result;**

**handler.sendMessage(msg);**

**}**

**}catch (Exception e) {**

**e.printStackTrace();**

**}**

**post方式：**

**//使用UrlConncetion请求服务器将用户密码发送服务器验证**

**try{**

**String path = "http://192.168.13.83:8080/itheima74/servlet/LoginServlet";**

**//1.创建一个httpclient对象**

**HttpClient httpclient = new DefaultHttpClient();**

**//2.创建一个请求方式**

**HttpPost httppost = new HttpPost(path);**

**//创建集合封装数据**

**ArrayList<BasicNameValuePair> arrayList = new ArrayList<BasicNameValuePair>();**

**BasicNameValuePair nameValuePair = new BasicNameValuePair("username",username);**

**arrayList.add(nameValuePair);**

**BasicNameValuePair nameValuePair1 = new BasicNameValuePair("pwd",password);**

**arrayList.add(nameValuePair1);**

**//创建一个Entity**

**UrlEncodedFormEntity entity = new UrlEncodedFormEntity(arrayList, "utf-8");**

**//设置请求时的内容**

**httppost.setEntity(entity);**

**//3.执行一个请求,返回一个response对象**

**HttpResponse response = httpclient.execute(httppost);**

**//4.获取状态码**

**int code = response.getStatusLine().getStatusCode();**

**//5.判断并获取内容**

**if(code == 200){**

**HttpEntity entity1 = response.getEntity();//获取实体内容，中封装的有http请求返回的流信息**

**InputStream inputStream = entity1.getContent();**

**//将流信息转换成字符串**

**String result = StreamUtils.streamToString(inputStream);**

**Message msg = Message.obtain();**

**msg.what = 2;**

**msg.obj = result;**

**handler.sendMessage(msg);**

**}**

**}catch (Exception e) {**

**e.printStackTrace();**

**}**

## 开源项目(asyncHttpClient) get post 方式提交

**get方式：**

**public static void requestNetForGetLogin(final Context context,final Handler handler ,final String username, final String password) {**

**//使用HttpClient请求服务器将用户密码发送服务器验证**

**try{**

**String path = "http://192.168.13.83:8080/itheima74/servlet/LoginServlet?username="+URLEncoder.encode(username,"utf-8")+"&pwd="+URLEncoder.encode(password,"utf-8");**

**//创建一个AsyncHttpClient对象**

**AsyncHttpClient asyncHttpClient = new AsyncHttpClient();**

**asyncHttpClient.get(path, new AsyncHttpResponseHandler() {**

**@Override**

**public void onSuccess(int statusCode, Header[] headers, byte[] responseBody) {**

**//statusCode:状态码 headers：头信息 responseBody：返回的内容，返回的实体**

**//判断状态码**

**if(statusCode == 200){**

**//获取结果**

**try {**

**String result = new String(responseBody,"utf-8");**

**Toast.makeText(context, result, 0).show();**

**} catch (UnsupportedEncodingException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**}**

**}**

**@Override**

**public void onFailure(int statusCode, Header[] headers,**

**byte[] responseBody, Throwable error) {**

**System.out.println("...............onFailure");**

**}**

**});**

**}catch (Exception e) {**

**e.printStackTrace();**

**}**

**}**

**post方式：**

**String path = "http://192.168.13.83:8080/itheima74/servlet/LoginServlet";**

**AsyncHttpClient asyncHttpClient = new AsyncHttpClient();**

**RequestParams params = new RequestParams();**

**params.put("username", username);**

**params.put("pwd", password);**

**//url: parmas：请求时携带的参数信息 responseHandler：是一个匿名内部类接受成功过失败**

**asyncHttpClient.post(path, params, new AsyncHttpResponseHandler() {**

**@Override**

**public void onSuccess(int statusCode, Header[] headers, byte[] responseBody) {**

**//statusCode:状态码 headers：头信息 responseBody：返回的内容，返回的实体**

**//判断状态码**

**if(statusCode == 200){**

**//获取结果**

**try {**

**String result = new String(responseBody,"utf-8");**

**Toast.makeText(context, result, 0).show();**

**} catch (UnsupportedEncodingException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**}**

**}**

**@Override**

**public void onFailure(int statusCode, Header[] headers,**

**byte[] responseBody, Throwable error) {**

**}**

**});**

# 文件上传的操作

**使用第三方utils做文件上传。**

**public void fileupload(View v){**

**try{**

**EditText et\_filepath = (EditText) findViewById(R.id.et\_filepath);**

**//获取输入的文件地址**

**String filepath = et\_filepath.getText().toString().trim();**

**//使用开源Utils做上传操作**

**AsyncHttpClient asyncHttpClient = new AsyncHttpClient();**

**RequestParams params = new RequestParams();**

**params.put("filename", new File(filepath)); 手机中的文件**

**//url : 请求服务器的url**

**asyncHttpClient.post("http://192.168.13.83:8080/itheima74/servlet/UploaderServlet", params, new AsyncHttpResponseHandler() {**

**@Override**

**public void onSuccess(int statusCode, Header[] headers, byte[] responseBody) {**

**if(statusCode == 200){**

**Toast.makeText(MainActivity.this, "上传成功", 0).show();**

**}**

**}**

**@Override**

**public void onFailure(int statusCode, Header[] headers,**

**byte[] responseBody, Throwable error) {**

**}**

**});**

**}catch (Exception e) {**

**e.printStackTrace();**

**}**

**}**

# 多线程断点下载

**多线程下载的步骤：**

**1.要知道服务端资源的大小。**

**通过URLConnection请求服务器url获取。**

**UrlConnection.getContentLength();//资源的大小**

**2.在本地创建一个与服务端资源同样大小的一个文件（占位）**

**//file : 文件； mode:文件的模式，rwd：直接写到底层设备，硬盘**

**RandomAccessFile randomfile =new RandomAccessFile(File file,String mode)**

**randomfile.setLength(long size);//创建一个文件和服务器资源一样大小**

**3.要分配每个线程下载文件的开始位置和结束位置。**

**计算每个线程下载的起始位置和结束位置**

**我们可以把原始文件当成一个字节数组，每个线程只下载该“数组”的指定起始位置和指定结束位置之间的部分。**

**在第一步中我们已经知道了“数组”的总长度。因此只要再知道总共开启的线程的个数就好计算每个线程要下载的范围了。**

**每个线程需要下载的字节个数（blockSize）=总字节数（totalSize）/线程数（threadCount）。**

**假设给线程按照0,1,2,3...n 的方式依次进行编号，那么第n 个线程下载文件的范围为：**

**起始脚标startIndex=n\*blockSize。**

**结束脚标endIndex=(n+1)\*blockSize-1。**

**考虑到totalSize/threadCount 不一定能整除，因此对已最后一个线程应该特殊处理，最后一个线程的起始脚标计算公式不变，但是结束脚标endIndex=totalSize-1;即可。**

***blockSize* = fileLength / *threadCount*;// 计算出每个线程理论下载大小**

**// 3.分配每个线程下载文件的开始位置和结束位置。**

**for (int threadID = 0; threadID < *threadCount*; threadID++) {**

**// 每个线程开始下载的位置**

**int startIndex = threadID \* *blockSize*;**

**// 每个线程结束下载的位置**

**int endIndex = (threadID + 1) \* *blockSize* - 1;**

**// 计算最后一个线程下载的结束位置**

**if (threadID == *threadCount* - 1) {**

**endIndex = fileLength - 1;**

**}**

**为每一个单独的线程创建一个临时文件，用于记录该线程下载的进度。**

**对于单独的一个线程，每下载一部分数据就在本地文件中记录下当前下载的字节数**

**这样子如果下载任务异常终止了，那么下次重新开始下载时就可以接着上次的进度下载。**

**4.开启线程去执行下载 分段请求网络 分段保存文件到本地**

**通过UrlConnection下载部分资源。**

**4.1读取上次下载结束的位置,本次从这个位置开始直接下载。**

**4.1.1如果文件存在**

**File file = new File(threadID + ".txt");**

**if (file.exists()) {**

**BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(new FileInputStream(threadI+ ".txt")));**

**String lastPostion\_str = bufferedReader.readLine();**

**lastPostion = Integer.*parseInt*(lastPostion\_str);// 读取文件获取上次下载的位置**

**设置分段下载的头信息。 Range:做分段数据请求用的。// bytes:0-500:请求服务器资源中下载的范围**

**connection.setRequestProperty("Range", "bytes:"+ lastPostion + "-" + endIndex);**

**4.1.2 文件不存在 重新设置开始下载的位置**

**lastPostion = startIndex;**

**connection.setRequestProperty("Range", "bytes:"+ lastPostion + "-" + endIndex);**

**4.3判断部分资源请求成功**

**connection.getResponseCode() == 206**

**4.4 请求成功将流写入已经创建的占位文件中**

**InputStream inputStream = connection.getInputStream();**

**byte[] bytes = new byte[1024];**

**int length =0;**

**int total=0;//设置线程下载的总大小变量**

**while ((length = inputStream.read(bytes))!=-1) {**

**randomAccessFile.write(bytes, 0, length);**

**total = total+length;**

**//当前线程开始下载的位置**

**int currentThreadlocation = startIndex+total;**

**//根据线程ID将当前下载的位置保存到文件中**

**RandomAccessFile accessFile = new RandomAccessFile(threadID+".txt", "rw");**

**accessFile.write(String.*valueOf*(currentThreadlocation).getBytes());**

**accessFile.close();**

**注意：**

**参数1：Range头作用 做分段数据请求 参数2：设置请求实体的范围**

**1.需要Range头，key：Range value：bytes:0-499**

**urlconnection.setRequestPropety("Range","bytes:0-499")**

**1.1 判断部分资源请求成功**

**1.2 将流中的数据写入占位的文件中 并设置开始写入的位置**

**2.需要设置每个线程在本地文件的保存的开始位置**

**RandomAccessFile randomfile =new RandomAccessFile(File file,String mode)**

**randomfile.seek(int startPostion);//本次线程下载保存的开始位置。**

**5.要知道每个线程下载完毕。**

**进入run方法后设定变量 开启一线程，变量加1**

**下载结束后 变量-1 当变量值＝0 标志着一个线程下载结束。**

**6.删除临时文件**

**当多个线程都下载完成之后，最后一个下载完的线程将所有的临时文件删除。**

**当进入run方法后 设置变量记录当前线程 +1；**

**下载完成后 当前变量 +1 ；**

**如果 当前设置的变量 ＝＝ 0 证明文件下载完成。 可以删除临时文件**

**if (*runningTrheadCount* == 0) {**

**// 删除文件**

**for (int i = 0; i < *countThread*; i++) {**

**File file2 = new File(i + ".txt");**

**file2.delete();**

**}**

## 开源项目实现多线程下载 (xutils)

**public void download(View v){**

**EditText et\_url = (EditText) findViewById(R.id.et\_url);**

**String url = et\_url.getText().toString().trim();**

**//1.创建httpUtils对象**

**HttpUtils httpUtils = new HttpUtils();**

**//2.调用download方法 url:下载的地址 target：下载的目录 callback：回调**

**httpUtils.download(url, "/sdcard/feiqiu/feiq.exe", new RequestCallBack<File>() {**

**@Override**

**public void onLoading(long total, long current, boolean isUploading) {**

**System.out.println("total:"+total+";current:"+current);**

**super.onLoading(total, current, isUploading);**

**}**

**@Override**

**public void onSuccess(ResponseInfo<File> responseInfo) {**

**System.out.println(responseInfo.result);**

**}**

**@Override**

**public void onFailure(HttpException error, String msg) {**

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

**}**

**});**

**}**

# Android中四大组件　Activity

**activity 　　广播接收者(broadCastReceiver)  服务(Service) 　　　内容提供者(contentprovider)**

**四大组件都需要在清单文件里面配置一下**

1. **<intent-filter>　意图过滤器用于配置intent的属性**
2. **</intent-filter>**

**Activity节点下的icon和label 可以和Application节点的icon和label不一样**

**Application 表示当前应用程序**

**隐式意图**

**通过指定一组动作或者数据 开启activity**

**隐式调用需要设置action、data、category、type，再有Android解析你的意图**

**对于本应用中的组件要实现隐式调用，需要在AndroidManifest.xml中配置<intent-filter>(意图过滤器)**

**Data与type同时使用的时候要使用setDateAndType()**

**显示意图**

**通过指定具体包名和类名 开启activity**

**显示调用是直接指定类名，Android不用再去解析intent**

**总结**

**(1)显示意图更加安全一些**

**(2)开启自己应用的界面用显示意图(不需要配置意图过滤器)**

**(3)隐式意图一般开启系统应用(电话拨号器 短信的发送器等等)的界面**

**人品计算器小案例**

**RadioButton存在于RadioGroup组中**

**RadioGroup.getCheckedRadioButtonId();获取RadioGroup组选中的单选框**

**传递数据 底层是Ｍap结构**

**intent.putExtra("name", name);**

**获取开启此Activity的意图对象**

**Intent intent = getIntent();**

**获取携带过来的数据 传递过来什么数据　取出就是什么数据**

**String name = intent.getStringExtra("name");**

**.附加数据 标准传递数据方式 ： setData ：Uri**

**传递数据**

**1.直接附加 intent.putExtra(key,value)**

**获取数据　intent.getStringExtra("book");**

**2.通过Bundle 附加**

**Bundle bundle=new Bundle();**

**bundle.putString("product", "phone");**

**获取数据**

**Bundle bundle=intent.getExtras();**

**String product=bundle.getString("product");**

**3.附加一个自定义的对象 ,对象需要实现序列化 对象实现implements Serializable**

**ChatMessage chatMessage=new ChatMessage();**

**chatMessage.id=5;**

**chatMessage.message="你好";**

**intent.putExtra("chatMessage", chatMessage);**

**startActivity(intent);**

**获取对象**

**Intent intent=getIntent();**

**ChatMessage chatMessage=(ChatMessage) intent.getSerializableExtra("message");**

## Intent意图

**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。**

**Intent表现形式**

**\* 调用Activity ： startActivity(intent) startActivityForResult(Intent)**

**\* 调用Service：服务 startService（intent） \bindService(intent)绑定服务**

**\* 发送广播 ：sendBroadcast(intent)发送无序广播**

**sendOrderedBroadcast(Intent)发送有序广播**

**Intnet 属性设置**

**\* setAction： 设置动作**

**setData : 设置数据 传递给目标组件的数据 ，它是Uri格式的数据**

**- setType ： 设置要访问的数据的MIME类型 ：（\*/image ,\*/video , text/plain）**

**- addCategory: 添加类别： 设置要访问的组件的类型 可以添加多个类别**

**类型：**

**category\_launcher (表示该Activity在应用程序顶级列表会展示)**

**category\_default :缺省类别　调用本应用组件　默认添加该类别**

**category\_home ： 系统启动的时候可以加载该Activity**

**category\_browsable :该Activity能够在浏览器中展示**

## Activity特点介绍

**活动，四大组件之一，也是最重要的组件，它通过内置的Window对象来展示一个与用户交互的界面，它的操作都与用户有关，一般一个布局针对于一个Activity界面**

**View控件捕获事件信息，WindowManagerService传递消息，再由Android框架回调Activity的相应监听方法实现与用户的交互**

**Application:表示一个应用程序　生命周期与整个应用一致**

**android:icon="@drawable/ic\_launcher"　表示应用程序的图示**

**android:label="@string/app\_name"　　　 表示应用程序的名称**

**android:theme="@style/AppTheme" >　 表示应用程序的主题**

**开启Activity　的两种方式**

**1.startActivity(Intent) :开启一个新的Activity，它将放到活动栈的栈顶**

**2.如果一个页面开启另外一个页面 并且当开启的这个页面关闭的时候 还要另外一个页面的数据**

**使用这个方法开启Activity　：startActivityForResult(Intent,int)**

**通过setResult(int,Intent)把结果回传给调用者**

**关闭当前Activity ：　finish();**

**调用者的onActivityResult方法就会执行　然后通过此方法处理回传结果**

**resultCode 结果码　requestCode请求码**

## Activity生命周期

**应用中的活动是由活动栈进行管理的，当一个新的Activity启动后，它会处于栈顶的位置，与用户进行交互，处于运行的状态;先前的Activity会移到它的下方,处于暂停或者停止状态**

**\* Activity生命周期**

**- 完全生命周期**

**自第一次调用onCreate()开始，直至调用onDestroy()为止。Activity在onCreate()中设置所有“全局”状态以完成初始化，而在onDestroy()中释放所有系统资源。**

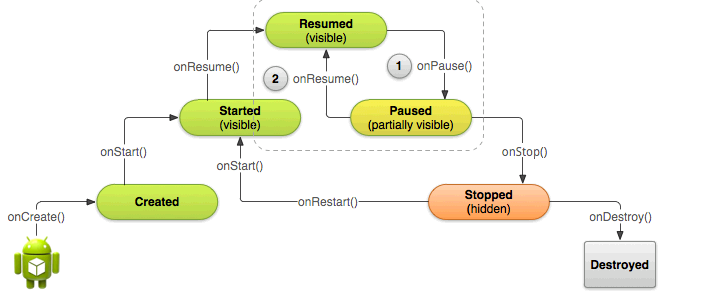
**- 可见生命周期**

**自onStart()调用开始直到相应的onStop()调用结束。在此期间，用户可以在屏幕上看到Activity，尽管它也许并不是位于前台或者也不与用户进行交互。**

**在这两个方法之间，我们可以保留用来向用户显示这个Activity所需的资源。**

**- 可交互的生命周期**

**自onResume()调用起，至相应的onPause()调用为止。在此期间，Activity位于前台最上面并与用户进行交互。Activity会经常在暂停和恢复之间进行状态转换。**

****

**\*Activity生命周期状态**

**Activity有四个生命周期状态，状态之间改变需要回调相应的生命周期方法来完成**

**- 运行状态 ：可以交互，看得见，该Activity跑到前台**

**- 暂停状态 ： 回调了onPause()方法，看得见部分界面，但是不能交互**

**- 停止状态 ： 调用了onStop，完全看不到界面**

**- 销毁状态 : onDestory()方法 (killed 状态) ：被系统杀死**

**Activity的生命周期 (必须掌握)**

**Activity生命周期状态会互相状态的，当回调了相应的生命周期方法，则进入到相应的状态中**

**oncreate 方法 当Activity创建的时候调用**

**onDestroy 方法 当Activity销毁的时候调用**

**onStrat() 方法 当Activity变成可见的时候调用**

**onStop() 方法 当activity 不可见的时候调用**

**onResume()方法 当activity可以获取焦点的时候  当界面的按钮可以被点击了**

**onPause()方法 当失去焦点的时候调用 当按钮不了可以被点击的时候调用**

**onRestart()当界面重新启动的时候调用**

**1. 当用户通过应用列表进入Activity或者激活某个Activity**

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

**2. 用户按下返回键 ，回调的生命周期方法**

**onPause-->onStop-->onDestory**

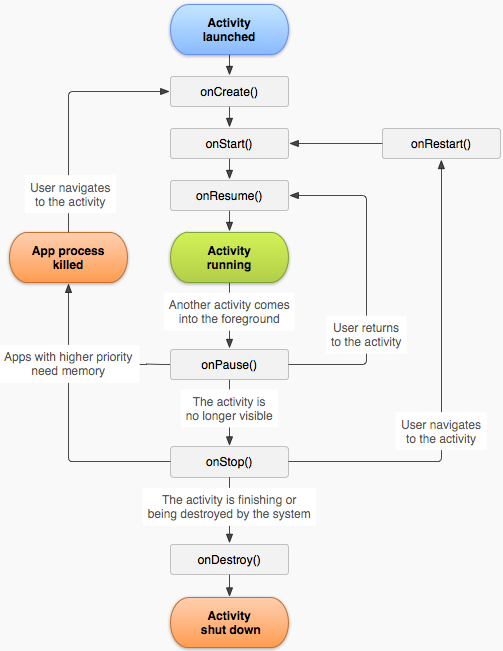
**3. 用户按下home 键**

**onPause-->onStop**

**4. 从停止状态-到运行状态**

**onRestart-->onStart-->onResume**

**执行onRestart() 方法之后一定会执行onStrat()方法　　开发中一般重写onStrat()方法**

****

**应用场景：**

**- onCreate(): 创建Activity时调用，一般在这里做初始化的操作，比如控件的初始化，绑定数据到列表中等**

**- onDestroy()：销毁，在这里释放资源 MediaPlayer.release()， 关闭数据库连接**

**- onStart() 开始方法 ：一般注册广播监听 ，绑定服务 onStop()注销监听 ，取消服务绑定**

**- onpause() 与onResume() ，这两个方法会经常调用，这里不易写耗时的操作，**

**一般在这里对持久化数据的保存 ，停止动画以及耗CPU的事情；onResume则相反**

**默认横竖屏切换 会销毁Activity再重新创建Activity**

**横竖屏切换Activity的生命周期两种方法**

**(1)为了防止横竖屏切换 生命周期会发生变化 所以把Activity配置如下**

**android:screenOrientation="portrait"**

**(2)为了声明周期不变化**

**android:configChanges="orientation|keyboardHidden|screenSize"**

**任务栈的概念**

**栈:先进后出**

**队列:先进先出**

**Task 打开一个Activity叫进栈  关闭一个activit出栈**

**任务栈是用来维护Activity的 是用来维护Activity或维护用户的操作体验**

**我们操作的Activity永远是任务栈的栈顶的Activity**

**说应用程序退出了 实际上任务栈清空了**

# Activity的四种启动模式

**对Activity实例的管理： 通过任务栈的形式来管理（TaskStack）**

**standard ：标准模式（每次激活，都会创建一个Activity实例）默认**

**singletop启动模式**

**singleTop :栈顶模式 :假如要激活的Activity它的实例刚好在栈顶的话，就不用再重新创建**

**singletop 单一顶部模式 在activity的配置文件中设置android:launchMode="singleTop"  
    如果任务栈的栈顶存在这个要开启的activity，不会重新的创建activity，而是复用已经存在的activity。保证栈顶如果存在，不会重复创建。  
    应用场景：浏览器的书签  
  
      
　singletask启动模式**

**singleTask: 单任务模式 ： 假如任务栈有该Activity的实例，就不需要重新创建，**

**因为该Activity要与用户交互，所有在该Activity实例前面的Activity都会弹出堆载**

**singetask 单一任务栈，在当前任务栈里面只能有一个实例存在  
    　当开启activity的时候，就去检查在任务栈里面是否有实例已经存在**

**如果有实例存在就复用这个已经存在的activity，并且把这个activity上面的所有的别的activity都清空，复用这个已经存在的activity。**

**保证整个任务栈里面只有一个实例存在  
      
    应用场景：浏览器的activity　回到主页面  
    如果一个activity的创建需要占用大量的系统资源（cpu，内存）一般配置这个activity为singletask的启动模式。**

**singleinstance启动模式**

**\*singleInstance： 单实例模式 ：主要用于实例共享，会单独开辟一个任务栈来存放该实例，  
    singleInstance启动模式非常特殊， activity会运行在自己的任务栈里面，并且这个任务栈里面只有一个实例存在  
    如果你要保证一个activity在整个手机操作系统里面只有一个实例存在，使用singleInstance  
    应用场景： 来电页面**

**onSaveInstanceState()和 onRestoreInstanceState()**

**Activity的 onSaveInstanceState() 和 onRestoreInstanceState()并不是生命周期方法，它们不同于 onCreate()、onPause()等生命周期方法，它们并不一定会被触发,当Activity容易被销毁时（如：内存不足、用户直接按Home键、跳转到下一个Activity、屏幕切换（横竖屏））onSaveInstanceState()才会被调用。但是当用户主动去销毁一个Activity时，例如在应用中按返回键，onSaveInstanceState()就不会被调用。因为在这种情况下，用户的行为决定了不需要保存Activity的状态。通常onSaveInstanceState()只适合用于保存一些临时性的状态，而onPause()适合用于数据的持久化保存。**

**需要注意的是，onSaveInstanceState方法和onRestoreInstanceState方法“不一定”是成对的被调用的，onRestoreInstanceState被调用的前提是，activity “确实”被系统销毁了**

## 应用程序的响应性(ANR异常)

**在Android中，应用的响应性被活动管理器（Activity Manager）和窗口管理器（Window Manager）这两个系统服务所监视。当用户触发了输入事件（如键盘输入，点击按钮等）,如果应用5秒内没有响应用户的输入事件，**

**那么，Android会认为该应用无响应，便弹出ANR（Application No Response）对话框**

**\* 为什么要弹出ANR异常**

**- 提升用户的体验**

**\* 原理**

**android启动应用程序，开启一个进程，再开启一个单一的线程，该线程也称为主线程，**

**主线程主要用于创建视图，显示视图，与用户交互（对视图控件的监听），对主线程也称为UI线程，对于耗时的操作，不应该在主线程运行，会影响与用户的交互**

**\* 如何解决**

**对于耗时的操作，比如访问网络、访问数据库等操作，需要开辟子线程，在子线程处理耗时的操作，主线程主要实现UI的操作（创建、显示、交互）**

**问题： 子线程不能直接操作UI控件，UI控件的相应方法，为了提升操作速度，都是线程不安全，**

**需要： 子线程与主线程通信**

**- Thread-Handler-Message**

**- AsyncTask**

**菜单设计**

**选择菜单**

**一个Activity最多只能拥有一个选择菜单，当选择菜单第一次访问的时候创建，该Activity销毁没有销毁的情况下，只创建一次**

**\* 创建选择菜单**

**onCreateOptionsMenu(Menu menu)**

**\* 监听选择菜单**

**onOptionsItemSelected(MenuItem item)**

**上下文菜单**

**一个组件可以绑定一个上下文菜单，上下文菜单每次弹出，都会重新创建一次**

**\* 注册上下文菜单**

**注册上下文菜单的两种写法**

**//registerForContextMenu(bookList);**

**// bookList.setOnCreateContextMenuListener(this);**

**\* 创建上下文菜单**

**& onCreateContextMenu(ContextMenu menu, View v,ContextMenuInfo menuInfo)**

**参数：menu：要创建的上下文菜单**

**v： 选择项的视图(ListView的Item)**

**menuInfo: 额外的菜单信息，依赖于选择的视图（V）**

**& 获取弹出上下文菜单的选项视图信息**

**//通过适配器上下文菜单信息类来获取 被选中的菜单项的数据信息**

**AdapterContextMenuInfo adapterInfo=(AdapterContextMenuInfo) menuInfo;**

**\* 监听上下文菜单**

**onContextItemSelected(MenuItem item)**

**//通过适配器上下文菜单信息类来获取 被选中的菜单项的数据信息**

**AdapterContextMenuInfo adapterInfo=(AdapterContextMenuInfo) item.getMenuInfo();**

**操作条(ActionBar)**

**\* 1.Action Bar特点**

**操作栏在Android 3.0版本以后出现，是一个窗口功能用于确定应用程序和用户的位置，并提供给用户操作和导航模式。Action Bar被认为是新版Android系统中重要的交互元素**

**\* 2、Action Bar分成四个区域**

**- App Icon：可显示软件icon，也可用其他图标代替。当软件不在最高级页面时，图标左侧会显示一个左箭头，用户可以通过这个箭头向上导航。  
 - 视图切换：如果你的应用要在不同的View中显示数据，这部分允许用户来切换View。一般的作法是用一个drop-down菜单或者是Tab Controls。如果只有一个界面，那这里可以显示App Title或者更长点的商标信息  
 - Action Buttons：这个放最重要的软件功能，放不下的按钮就自动进入Action overflow了。  
 - Action overflow：把不常用的Actions移到Action overflow**

**、Action Bar常用的交互操作**

**- 选择操作项**

**与选择菜单一致，都是使用相同的api来创建和监听操作项**

**- 操作导航选项标签**

**选项标签要与Fragment相结合，实现界面切换的效果**

**1. 添加标签 ： ActionBar.Tab**

**2. 监听标签 ： ActionBar.TabListener**

# android中的动画

**(1)属性动画**

**(2)补间动画**

**(3)帧动画**

**就是加载一系列的图片资源**

## 帧动画

**在rec目录下新建一个drawable目录(Folder) 然后在drawable目录下新建一个Android XML File 文件**

**在新建的文件中配置: <animation-list xmlns:android="http://schemas.android.com/apk/res/android"**

**android:oneshot="true">(表示动画执行一次)**

**<item android:drawable="@drawable/(在drawable下复制的图片id)" android:duration="200" (图片持续的时长)/>**

**setContentView(R.layout.*activity\_main*);**

**//1.找到控件，用来显示动画**

**ImageView lv = (ImageView) findViewById(R.id.*lv*);**

**//2.设置背景资源**

**lv.setBackgroundResource(R.drawable.*my\_anim*);**

**//3.获取AnimationDrawable 类型**

**AnimationDrawable background = (AnimationDrawable) lv.getBackground();**

**//4.开始执行动画**

**background.start();**

## **补间动画(渐变动画)**

**渐变动画在代码中使用的是AlphaAnimation 类来定义，在XML 文件中使用<alpha>节点来定义。下面分别演示使用XML 文件和Java 代码的方式实现Alpha 渐变动画。**

1. **使用XML 文件实现Alpha 动画**
2. **在res 目录下创建anim 文件夹。**
3. **在anim 文件夹中创建alpha\_anim.xml 文件，文件名可以自定义，这里叫alpha\_anim.xml。**

**编辑alpha\_anim.xml 文件**

**例：**

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

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

**android:fromAlpha=*"0.0"***

**android:toAlpha=*"1.0"***

**android:duration=*"2000"***

**android:repeatCount=*"infinite"***

**android:repeatMode=*"reverse"* >**

**</alpha>**

**属性解释如下：**

* **duration 动画时长**
* **fromAlpha 起始透明度，1 为完全不透明，0 为完全透明**
* **repeatCount 重复次数**
* **toAlpha 目标透明度**
* **repeatMode 重复模式，restart 为重新开始**

1. **添加Java 逻辑代码:使用AnimationUtils 工具类加载xml 文件，获取Animation 对象；调用startAnimation 让ImageView 执行此动画。**
2. **使用编码方式实现Alpha 动画**

**使用编码方式可以实现跟xml 文件同样的效果，上面的效果用java 代码实现如下：**

**/\*\***

**\* 透明度渐变的动画**

**public void alpha(View view){**

**fromAlpha 起始透明度，1 为完全不透明，0 为完全透明**

**AlphaAnimation aa = new AlphaAnimation(0.0f, 1.0f);**

**//动画播放的时间长度**

**aa.setDuration(2000);**

**//设置重复播放的次数 -1代表无限次播放**

**//Animation.INFINITE 无限次播放**

**aa.setRepeatCount(-1);**

**//设置重复播放的模式**

**aa.setRepeatMode(Animation.*REVERSE*);**

**//让iv播放aa动画**

**iv.startAnimation(aa);**

**}**

### **Scale 伸缩动画**

**布局方式参考Aplah渐变动画，这里只给出代码实现的方式：**

**/\*\***

**\* 缩放动画**

**\*/**

**public void scale(View view){**

**/\***

**\* 第一个参数fromX x 轴起始大小(这个大小指倍数，它内部会用这个倍数去乘实际像素)**

**\* 第二参数toX 轴截止大小(若起始大小=截止大小就是指x 轴不伸缩)**

**\* 第三个参数fromY Y 轴的起始大小**

**\* 第四个参数toY 轴的截止大小**

**\* 第五个参数pivotXType X 轴的原点的类型（相对于自己而言还是相对于父容器而言）**

**\* 第六个参数pivotXValue 开始伸缩时的X 轴的原点**

**\*(例:0.5 就是指以图片宽度的二分之一的位置作为X 轴的原点)**

**\* 第七个参数pivotYType Y 轴的原点的类型**

**\* 第八个参数pivotYValue 开始伸缩时的Y 轴的原点**

**\*/**

**ScaleAnimation sa = new ScaleAnimation(0.2f, 2.0f, 0.2f, 2.0f,Animation.*RELATIVE\_TO\_SELF*, 0.5f, Animation.*RELATIVE\_TO\_SELF*, 0.5f);**

**//动画播放的时间长度**

**sa.setDuration(2000);**

**//设置重复播放的次数**

**sa.setRepeatCount(Animation.*INFINITE*);**

**//设置重复播放的模式**

**sa.setRepeatMode(Animation.*REVERSE*);**

**//让iv播放aa动画**

**iv.startAnimation(sa);**

**}**

### **Translate 位移动画**

**Translate 动画代码片段：**

**/\*\***

**\* 平移动画**

**public void trans(View view){**

**/\***

**\* 第一个参数fromXType 位移的x 轴起始坐标的类型(相对于自己还是相对父容器)**

**\* 第二参数fromXValue x 轴起点(0:自身最左边的x 坐标1：最右边的x 坐标)**

**\* 第三个参数toXType X 轴终点坐标的类型**

**\* 第四个参数toXValue X 轴的终点**

**\* 第五个参数fromYType Y 轴起始坐标的类型**

**\* 第六个参数fromYValue Y 轴的起始坐标**

**\* 第七个参数toYType Y 轴的终点坐标的类型**

**\* 第八个参数toYValue Y 轴的终点坐标**

**\*/**

**TranslateAnimation ta = new TranslateAnimation(Animation.*RELATIVE\_TO\_SELF*,**

**0,**

**Animation.*RELATIVE\_TO\_PARENT*,**

**-2f,**

**Animation.*RELATIVE\_TO\_SELF*,**

**0,**

**Animation.*RELATIVE\_TO\_SELF*,**

**-2f);**

**//动画播放的时间长度**

**ta.setDuration(2000);**

**//设置重复播放的次数**

**ta.setRepeatCount(Animation.*INFINITE*);**

**//设置重复播放的模式**

**ta.setRepeatMode(Animation.*REVERSE*);**

**//让iv播放aa动画**

**iv.startAnimation(ta);**

**}**

### **Rotate 旋转动画**

**/\*\***

**\* 旋转动画**

**\*/**

**public void rotate(View view){**

**/\***

**\*第一个参数fromDegrees：旋转的起始角度**

**\*第二个参数toDegrees:旋转的结束角度**

**\*第三个参数pivotXType:X 轴原点的类型(相对于自身还是相对于父容器)**

**\*第四个参数pivotXValue:原点的X 轴坐标**

**\*第五个参数pivotYType： Y 轴原点的类型**

**\*第六个参数pivotYValue:原点的Y 轴坐标**

**\*/**

**RotateAnimation ra = new RotateAnimation(0, 360, Animation.*RELATIVE\_TO\_SELF*, 0.5f, Animation.*RELATIVE\_TO\_SELF*, 0.5f);**

**//动画播放的时间长度**

**ra.setDuration(2000);**

**//设置重复播放的次数**

**ra.setRepeatCount(Animation.*INFINITE*);**

**//设置重复播放的模式**

**ra.setRepeatMode(Animation.*REVERSE*);**

**//让iv播放aa动画**

**iv.startAnimation(ra);**

**}**

**总结: 动画效果不会改变控件真实的坐标**

## 属性动画

**通过控制控件的属性可以实现动画效果，属性动画比补间动画要更灵活、强大的多。**

**需要注意的是属性动画内部其实并没有区分位移、缩放、透明、旋转等动画，其核心思想只是修改一个控件的属性，我们可以通过修改不同的属性来实现补间动画的4 种效果而已。**

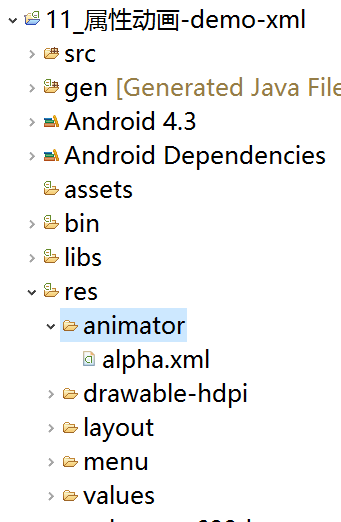
**通过xml 文件实现属性动画**

**和补间动画类似，属性动画也可以通过xml 文件来定义。**

**实现步骤如下：**

**1) 在res 下创建属性动画文件**

**如下图所示，在res 目录下创建animator 文件夹，然后创建一个objectAnimator 资源文件。资源名称自定义即可。**

****

**2) 编写属性动画文件**

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

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

**android:propertyName=*"alpha"***

**android:duration=*"3000"***

**android:repeatCount=*"3"***

**android:repeatMode=*"reverse"***

**android:valueFrom=*"0.0"***

**android:valueTo=*"1.0"***

**>**

**</objectAnimator>**

**3) 编写代码使用属性动画文件**

**/\*\***

**\* 透明度渐变的动画**

**public void alpha(View view) {**

**//加载动画资源**

**Animator animator = AnimatorInflater.*loadAnimator*(this,R.animator.*alpha*);**

**//指定要显示动画的控件(设置执行目标)**

**animator.setTarget(iv);**

**//开启动画**

**animator.start();**

**}**

**创建属性动画(代码执行)**

**\* target 执行的目标**

**\* propertyName 属性名字 The name of the property being animated.**

**\* float... values 可变参数**

**ObjectAnimator oa = ObjectAnimator.ofFloat(iv, "translationX", 10, 50,20,100);**

**挨个执行动画**

**as.playSequentially(oa, oa2, oa3, oa4);**

**一起执行动画**

**as.playTogether(oa, oa2, oa3, oa4);**

**oa.setDuration(2000);**

**oa.start(); 开始动画**

**注意：**

**属性动画跟补间动画比，属性动画是真正改变了控件的属性，因此当控件的位置改变后只要点击到了控件“身上”就能触发onClick 事件。**

**而补间动画则并没用改变控件的真实属性，因此不管属性动画执行后将控件移动到了哪个位置，只能通过点击该控件的原始位置才能触发onClick 事件。**

# **Android广播机制BroadcastReceiver**

**当遇到这些问题：电池低、网络断网、下载完成或某个图片被捕获，想通知你的用户，那该怎么办?**

**在Android平台中， 通过广播机制能很好解决这些问题,当遇到某些系统事件时，比如收到短信、电量偏低等，就会以广播的形式发送事件消息，并且Android框架会自动唤醒订阅了该广播事件类型的广播接收器来处理，处理完毕后，Receiver马上退出。**

**Android广播机制的特点：**

**1. 发送者不必关心接受者的情况，发送者与接收者完全的解耦**

**2. 广播事件是系统级别的事件，而我们开发的应用的事件（比如控件的点击事件），它是属于应用级别的事件**

**3. 系统内置了大量的广播事件，订阅系统的广播事件可以丰富我们应用的功能**

**4. 广播机制主要解决的是应用程序之间通信的功能**

**Android广播接收器**

**\* 特点：**

**1. Android四大组件之一，是唯一可以动态注册的组件　是一种应用在程序之间传输信息的机制**

**2. 没有图形界面，注册广播接收者只要事件响应　应用程序能够自动启动，在主线程运行**

**3. BroadcastReceiver组件生命周期比较短,10秒左右，执行完组件的onReceive()方法后后会立即销毁，所以广播接收器组件应该扮演一个网关的角色，只作简单的信息处理，比如发通知或者Toast;对于复杂的业务应该开启一个Service或者在另一个应用中去完成业务**

**4. 不适合在广播接收器开启子线程来处理耗时的操作，因为它的生命周期只有10秒左右，超过时间则框架会自动销毁该组件**

**下载文件 50分钟 应用程序 ： receiver 10 秒**

**\* 运行流程**

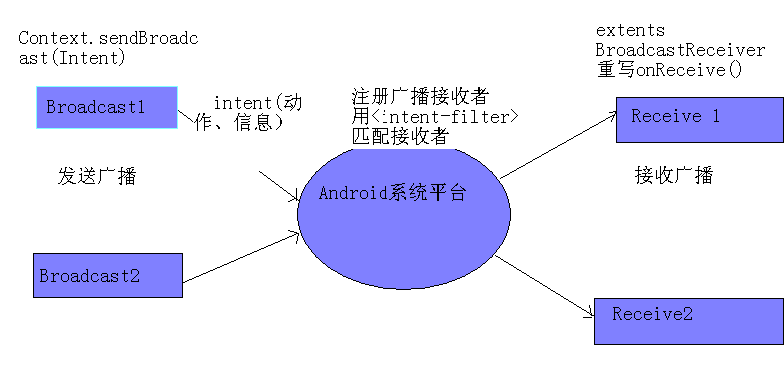
**首先把要发送的信息及动作封装在Intent中，由系统事件机制触发广播(比如：日期改变、电池电量低、网络断网、收到短信等)或者自己通过Content.sendBroadcast(Intent)发送广播。当Intent发送以后，系统依据发送的Intent信息与已经注册的广播接收器组件进行匹配，若匹配成功，则实例化相应的BroadcastReceiver组件，调用onReceive(Context curContext, Intent broadcastMsg)方法，运行完毕，销毁广播组件。**

**广播的三要素**

**- 广播的发送者 ，一般是系统，还有用户**

**- 广播的接受者，广播接收器**

**- 媒介 ： Intent**

****

**广播发送方就是调用Context.sendBroadcast()的程序，而广播接收者就是继承BroadcastReceiver的程序；通常广播发送方都是通过隐式意图**

**Android广播分为两个方面：广播发送者和广播接收者，通常情况下，BroadcastReceiver指的就是广播接收者（广播接收器）。广播作为Android组件间的通信方式，可以使用的场景如下：  
1.同一app内部的同一组件内的消息通信（单个或多个线程之间）；**

**2.同一app内部的不同组件之间的消息通信（单个进程）；**

**3.同一app具有多个进程的不同组件之间的消息通信；**

**4.不同app之间的组件之间消息通信；**

**5.Android系统在特定情况下与App之间的消息通信。**

**从实现原理看上，Android中的广播使用了观察者模式，基于消息的发布/订阅事件模型。因此，从实现的角度来看，Android中的广播将广播的发送者和接受者极大程度上解耦，使得系统能够方便集成，更易扩展。具体实现流程要点粗略概括如下：**

**1.广播接收者BroadcastReceiver通过Binder机制向AMS(Activity Manager Service)进行注册；**

**2.广播发送者通过binder机制向AMS发送广播；**

**3.AMS查找符合相应条件（IntentFilter/Permission等）的BroadcastReceiver，将广播发送到BroadcastReceiver（一般情况下是Activity）相应的消息循环队列中；**

**4.消息循环执行拿到此广播，回调BroadcastReceiver中的onReceive()方法。**

**对于不同的广播类型，以及不同的BroadcastReceiver注册方式，具体实现上会有不同。但总体流程大致如上。**

**自定义广播接收器接收系统广播事件**

**实现步骤：**

**1. 编写类，继承BroadcastReciver ，并且要重新onReceive() 方法**

**2. 在AndroidManifest.xml 注册组件**

**3. 在AndroidManifest.xml订阅广播事件**

**onReceive　当接收到xml清单文件注册的事件的时候会执行这个方法**

**getResultData();　获取xml清单文件注册的事件内容**

**intent.getAction();　获取到当前广播的事件类型**

**SD卡事件配置**

**<action android:name="android.intent.action.MEDIA\_UNMOUNTED"/>**

**<action android:name="android.intent.action.MEDIA\_MOUNTED"/>**

**<action android:name="android.intent.action."/>**

**<!--想让上面的这2个SD卡事件生效 必须的加上这样的一个data -->**

**<data android:scheme="file"/>**

**短信到来事件配置**

**<action android:name="android.provider.Telephony.SMS\_RECEIVED" />**

**获取发短信送的号码 和内容**

**Object[] objects = (Object[]) intent.getExtras().get("pdus");**

**for (Object pdu : objects) {**

**获取smsmessage实例**

**SmsMessage smsMessage = SmsMessage.createFromPdu((byte[]) pdu);**

**获取发送短信的内容**

**String body = smsMessage.getMessageBody();**

**获取发送者**

**String address = smsMessage.getOriginatingAddress();**

**程序的安装与卸载广播事件配置**

**<intent-filter >**

**<action android:name="android.intent.action.PACKAGE\_INSTALL"/>**

**<action android:name="android.intent.action.PACKAGE\_ADDED"/>**

**<action android:name="android.intent.action.PACKAGE\_REMOVED"/>**

**<data android:scheme="package"/>**

**当按手机后退按键的时候 会执行 onBackPressed方法**

**手机重启需添加android.permission.RECEIVE\_BOOT\_COMPLETED　权限**

**广播里面开启一个Activity**

**Intent intent2 = new Intent(context, MainActivity.class);**

**如果在广播里面开启Activity 要设置一个任务栈环境**

**intent2.setFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);**

**context.startActivity(intent2);**

**如果在广播中开启Activity没有添加任务栈环境会抛出此异常**

**java.lang.RuntimeException: Unable to start receiver com.itheima.bootreceiver.**

**BootReceiver: android.util.AndroidRuntimeException: Calling startActivity() from outside of an Activity**

**context requires the FLAG\_ACTIVITY\_NEW\_TASK flag. Is this really what you want?**

**四大组件都运行在进程中　进程是应用程序的载体　　BroadcastReceiver只要这个事件到来，系统就会把当前应用的进程开启，就会自动生效**

**无序广播与有序广播(自定义广播)**

**\* 无序广播**

**所有的广播接收器运行在一个未定义的顺序，不能屏蔽结果，不能修改广播, 无法终止广播 ，可通过Context.sendBroadcast发送无序广播**

**发送广播时还需设置action(intent.setAction) 接收广播　需在清单文件中配置相同action**

**\* 有序广播:**

**有序广播在同一时间传送到一个接收器。由于每个接收器依次执行，它可以结果传播到下一个接收器，或者它可以完全中止的广播，需定义优先级优 先级别在intent-filter 中的 priority 中声明 ,-1000 到1000 之间 , 值越大 , 优先级越高 . 接收者可以截断广播等，可通过Context.sendOrderBroadcast发送有序广播。**

**android:priority ：设定广播接收器的优先级**

**abortBroadcast() ：截断广播、终止广播传递**

**<intent-filter android:priority=*"1000"* >//配置优先级**

**发送一条有序的广播**

**Intent intent = new Intent();**

**intent.setAction("com.itheima.rice");**

**receiverPermission:接收的权限,**

**resultReceiver:最终的receiver不需要在清单文件中配置(无论是否停止广播 ，最终都能接受到广播),**

**scheduler: Handler消息机制**

**initialCode:初始化的code,**

**initialData:初始化的数据,**

**initialExtras:额外的数据**

**sendOrderedBroadcast(intent, null, new FinalReceiver(), null, 1, "给米",null);**

**获取到发送有序广播的数据**

**String resultData = getResultData();**

**Toast.*makeText*(context, "省" + resultData, Toast.*LENGTH\_SHORT*).show();**

**终止广播**

**abortBroadcast();**

**修改数据**

**setResultData("发了500大米");**

## 特殊广播接收者

**在Android 中一些操作比较频繁的事件，比如锁屏解屏和电量的变化，也会发送特定的广播。**

**但是此类广播的注册是无法注册在AndroidManifest.xml 中，只能在代码中进行注册。**

**BroadCastReceiver 的注册方式有两种：1、静态注册（就是通过AndroidManifest.xml 注册）2、动态注册（就是通过代码注册）。**

**静态注册的不足： 一旦接收器组件通过静态方式（AndroidManifest.xml）,则假如有相应的广播事件产生，则该组件会自行启动，除非把该应用卸载掉**

**在本文中前面使用到的BroadCastReceiver 全部都使用的是静态注册方式，其实也可以使用动态注册，但是对于锁屏解屏和电量变化的监听只能通过动态注册。**

**广播接收器也是唯一可以动态注册的Android组件**

**[1]动态的去注册屏幕解锁和锁屏的广播**

**screenReceiver = new ScreenReceiver();**

**[2]创建intent-filter对象**

**IntentFilter filter = new IntentFilter();**

**[3]添加要注册的action**

**filter.addAction("android.intent.action.SCREEN\_OFF");**

**filter.addAction("android.intent.action.SCREEN\_ON");**

**[4]注册广播接收者**

**this.registerReceiver(screenReceiver, filter);**

**注意：必须在Activity中onDestory 方法或者其他地方（必须保证Activity 销毁前）将接收者给注销**

**onDestroy()**

**当activity销毁的时候 取消注册广播接收者**

**unregisterReceiver(screenReceiver);**

**Android4.0以后版本的广播机制问题**

**在3.1之后，系统的package manager增加了对处于“stopped state”应用的管理，**

**指的是安装后从来没有启动过，与此同时系统增加了2个Flag：FLAG\_INCLUDE\_STOPPED\_PACKAGES和FLAG\_EXCLUDE\_STOPPED\_PACKAGES ，**

**来标识一个intent是否激活处于“stopped state”的应用。**

**Android默认给所有的广播intent加上FLAG\_EXCLUDE\_STOPPED\_PACKAGES，能在一定程度上避免流氓软件、病毒，还能提高效率，如果用户没有运行过应用，就不会响应了。**

**不过对于自定义的广播接收器我们可以通过setFlags(Intent.FLAG\_INCLUDE\_STOPPED\_PACKAGES);这个方法来唤醒处于“stopped state”的程序，对于系统发的广播，则无能为力**

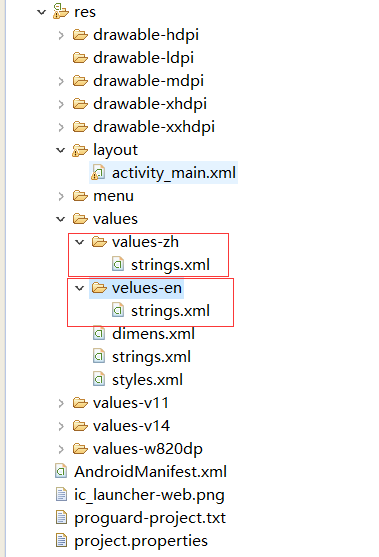
**解决办法：**

1. **在你的广播接收器应用中添加一个Activity，让用户运行一次你的应用**
2. **通过其他应用发一个自定义广播事件，intent设置为intent.setFlags(Intent.FLAG\_INCLUDE\_STOPPED\_PACKAGES)，即包含停止状态的包也能够被激活，你的广播接收器应用订阅了系统的广播事件，同时也订阅了自定义的广播事件，这样就能够激活处于停止状态的广播接收器组件**

# 国际化 (i18n)

**一般是对字符串做国际化 国际化就是在res下的values中新建Folder然后把 values目录下的strings.xml文件复制到刚新建的**

**对应国家环境目录 values-en(当前国家的语言简称)..然后将对应的String改成对应国家的语言**

****

**Service服务**

**四大组件之一，需要注册、没有图形界面****，生命周期长、不能自启动，需要startService或者bindService来启动服务，默认在主线程运行**

**主要用于实现生命周期较长的操作,比如：打印服务、下载服务、音乐服务、更新服务**

**四大组件都是运行在主线程**

## Android进程等级

**Android系统会尽量维持一个进程的生命，直到最终需要为新的更重要的进程腾出内存空间。**

**为了决定哪个进程该终止，系统会跟据运行于进程内的组件的和组件的状态把进程置于不同的重要性等级。**

**Android进程共分为五个等级，分别是：前台进程(Foreground Process),可视进程( Visible Process), 服务进程(Service Process), 后台进程(Background Process), 空进程(Empty Process)**

**前台进程等级最高，空进程等级最低，系统需要内存资源时，会优先回收等级低的进程，比如空进程、后台进程等**

**Service 并不影响我们使用Thread，而且很多情形下，我们都是在Service 中开启子Thread的混合使用方式。**

****1. 前台进程** Foreground process**

**用户正在交互 如果满足下面的条件，一个进程就被认为是前台进程：**

**1)这个进程拥有一个正在与用户交互的Activity**

1. **这个进程拥有一个绑定到正在与用户交互的activity上的Service。  
   3)这个进程拥有一个前台运行的Service　—　service调用了方法startForeground().  
   4)这个进程拥有一个正在执行其任何一个生命周期回调方法（onCreate(),onStart(或onDestroy()）的Service。  
    5)这个进程拥有正在执行其onReceive()方法的BroadcastReceiver。**

****2. 可视进程** Visible process 用户没有在交互 但用户还一直能看得见页面**

**一个进程不拥有运行于前台的组件，但是依然能影响用户所见。满足下列条件时，进程即为可见：  
     1)这个进程拥有一个不在前台但仍可见的Activity(它的onPause()方法被调用)。  
     2)这个进程拥有一个绑定在前台（或者可见）Activity的服务。  
    一个可见的进程是极其重要的，通常不会被终止，除非内存不够，需要释放内存以便前台进程运行。**

****3.服务进程** Service Process   通过startService()开启了一个服务**

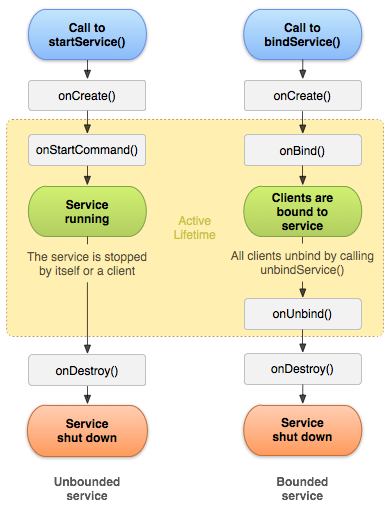
**一个进程不在上述两种之内，但它运行着一个被startService()所启动的service。  
 尽管一个服务进程不直接影响用户所见，但是它们通常做一些用户关心的事情（比如播放音乐或下载数据），所以除非系统没有足够的空间运行前台进程和可见进程时才会终止一个服务进程。**

1. ****后台进程** 当前用户看不见页面 相当于Activity执行了onStop方法**

**这样的进程们不会直接影响到用户体验，所以系统可以在任意时刻杀了它们从而为前台、可见、以及服务进程们提供存储空间。**

****5.空进程****

**没有运行任何component的进程，保留这个进程主要是为了缓存的需要，这种进程存在的唯一原因是做为缓存以改善组件再次于其中运行时的启动时间**

****

**服务生命周期图**

## 全局服务 (没有约束的服务) startservice

**start方式开启服务的特点**

**服务是在后台运行 可以理解成是没有界面的activity**

**定义四大组件的方式都是一样的**

**定义一个类继承Service　清单文件中配置**

**特点:**

**(1)服务通过startservice方式开启 第一次点击按钮开启服务 会执行服务的onCreate 和 onStartCommand方法**

**(2)如果第二次开始在点击按钮开启服务 服务之后执行onStratCommand方法**

**(3)服务被开启后 会在设置页面里面的 running里面找得到这个服务**

**(4)startservice 方式开启服务 服务就会在后台长期运行 直到用户手工停止 或者调用StopService方法 服务才会被销毁**

**启动context.startService(intent)-->onCreate(服务的回调方法，无服务实例则回调) -->onStartCommand(intent ，服务的生命周期方法)-->服务运行**

**交互： context.startService(intent)-->onStartCommand(intent )**

**停止服务 context.stopService(intent)-->onDestory(服务的生命周期方法)**

**第一次 启动服务时，运行 onCreate (服务第一次被开启的时候调用)-->onStartCommand**

**后面在启动服务时，服务只执行onStartCommand**

**在实际使用过程中，通过Intent 传递数据，在OnStartCommand中执行**

**在onCreate 中执行监听电话的状态**

**TelephoneManager类电话管理者　通过getSystemService(*TELEPHONY\_SERVICE*);**

**注册电话监听：**

**Listen()　(参数1：PhoneStateListener(抽象类)　参数2：样式)**

**定义电话监听类(PhoneStateListener的继承类)：**

**onCallStateChanged　当设备的状态发生改变的时候会调用此方法**

**实现录音的功能获取获取MediaRecorder类的实例**

**2]设置音频的来源**

**recorder.setAudioSource(MediaRecorder.AudioSource.MIC);**

**MediaRecorder.AudioSource.*MIC　只支持单方录音***

**MediaRecorder.AudioSource.*VOICE\_CALL　支持双方录音***

**[3]设置音频的输出格式**

**recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE\_GPP);**

**[4]设置音频的编码方式**

**recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR\_NB);**

**[5]保存的文件路径**

**recorder.setOutputFile("/mnt/sdcard/luyin.3gp");**

**recorder.prepare();　　开始录音**

**recorder.reset();　重置MediaRecorder实例**

**recorder.release();　释放资源**

**权限**

**<uses-permission android:name="android.permission.READ\_PHONE\_STATE"/>**

**<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />**

**<uses-permission android:name="android.permission.RECORD\_AUDIO" />**

**<uses-permission android:name="android.permission.RECEIVE\_BOOT\_COMPLETED"/>**

## 绑定服务 bindService

**为什么要引入bindService目的为了调用服务里面的方法**

**一旦服务开启，那么调用者和服务就有密切关系了，如果所有客户端被destroy了**

**那么服务也就会被destroy。且可以调用服务内部中的方法。调用者被销毁前，需要对绑定的服务解绑。**

**\* 启动：bindService(Activity方法) -->onCreate()-->onBind()**

**\* 交互：直接在Activity中拿到Service的实例或者Service的代理对象进行操作**

**\* 关闭：unBindService(Activity方法)-->OnUnbind()-->ondestroy()**

**(1)当点击按钮第一次开启服务 会执行服务的onCreate方法 和 onBind()方法**

**(2) 当第二次点击按钮在调用bindservice  服务没有响应**

**\*\*(3) 当activity销毁的时候服务也销毁(activity销毁的时候取消绑定服务 onDestroy()方法中写unbindService(传递参数：监听服务状态类))**

**不求同时生但求同时死**

**(4)通过bind方式开启服务  服务不能再设置页面里面找到  相当于是一个隐形的服务**

**(5)bindservice不能多次解绑 多次解绑会报错**

**Bindservice(参数1：连接服务的intent 参数2：接口(ServiceConnction 用于监听服务的状态) 参数3：如何连接到服务(BTND\_AUTO\_CREATE :自动连接))**

**ServiceConnction (用于监听服务的状态 有二个回调方法 )**

**1.onServiceConnected:服务连接成功后调用 2.onServiceDisConnected:失去连接调用**

**通过bindService获取服务中的方法**

**0.用bindservice开启服务**

**1.(IBinder) 继承Binder 获取中间对象**

**2.在onBind()将定义的中间对象返回**

**3.在Activity的onCreate()调用bindService()方法**

**4.通过接口ServiceConnction 实现类中的onServiceConnected()获取中间对象 通过中间对象调用服务中的方法**

**5.在Activity的onDestroy()方法中解绑服务 unbindService(参数传递：ServiceConnction 接口的实现类);**

**让服务在后台长期运行又调用服务里面的方法 混合方式开启服务**

**(1)先调用startService()方法 保证服务在后台长期运行**

**(2)调用bindservice()目的获取我们定义的中间对象 调用服务里面的方法**

**(3)unbindservice() 解绑服务**

**(4)最后调用stopservice() 停止服务**

## IntentService 通过startService 调用

**这是一个[Service](http://docs.eoeandroid.com/reference/android/app/Service.html)的子类,该子类使用线程处理所有启动请求,.你需要做的只是实现[onHandleIntent()](http://docs.eoeandroid.com/reference/android/app/IntentService.html" \l "onHandleIntent(android.content.Intent))方法即可.依据Intent的请求指令执行相应的业务。**

**特点：**

**1. IntentServcie启动只能通过startService方式，通过Intent传指令，在onHandleIntent（Intent）处理请求命令，且该方法在子线程运行**

**2. 它是Service的子类，会自动开启子线程处理业务请求**

**3. 当请求的业务处理完毕后，服务自动停止**

**4. 服务会开启一个子线程处理每批次的请求，当该批次的请求处理完毕，服务自动结束**

**\* 流程介绍:**

**1. 从应用的主线程当中创建一个默认的线程执行所有的intents发送给onStartCommand()方法,该线程从主线程分离.**

**2. 创建工作队列,每次传递一个intent 给onHandleIntent()方法实现,不必担心多线程.**

**3. 所有的请求被处理后服务停止，所以你永远都不必调用stopSelf()函数.**

**4. 默认实现onBind()方法返回null.**

**IntentService ：startService 调用**

**public class MyIntentServiceActivity extends Activity {**

**@Override**

**protected void onCreate(Bundle savedInstanceState) {**

**super.onCreate(savedInstanceState);**

**setContentView(R.layout.activity\_my\_intent\_service);**

**}**

**public void upload(View v){**

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

**intent.putExtra("cmd", 1);//代表上传**

**startService(intent);**

**}**

**public void download(View v){**

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

**intent.putExtra("cmd", 2);//代表下载**

**startService(intent);**

**public class FileService extends IntentService {**

**public FileService() {**

**//name是该Service的标记，一般不用**

**super("myFileService");**

**}**

**@Override**

**public void onCreate() {**

**System.out.println("onCreate");**

**super.onCreate();**

**}**

**@Override**

**public int onStartCommand(Intent intent, int flags, int startId) {**

**System.out.println("onStartCommand");**

**return super.onStartCommand(intent, flags, startId);**

**}**

**protected void onHandleIntent(Intent intent) {**

**//该方法在子线程运行，处理耗时的业务**

**int cmd =intent.getIntExtra("cmd", 0);**

**switch (cmd) {**

**case 1://上传指令**

**System.out.println("onHandleIntent+上传");**

**SystemClock.sleep(2000);**

**break;**

**case 2://下载指令**

**System.out.println("onHandleIntent+下载");**

**break;**

## AIDL介绍

**(1)远程服务 运行在其他应用里面的服务**

**(2)本地服务 运行在自己应用里面的服务**

**(3)进行进程间通信(IPC)**

**(4)aidl Android interface Defination Language Android接口定义语言 专门是用来解决进程间通信的**

**aidl 实现步骤和之前调用服务里面的方法的区别**

**(0)用隐式意图开启服务**

**(1)先把Iservice.java文件变成aidl文件改后缀名**

**(2)adil 不认识public 把public 给我去掉**

**(3)会自动生成一个Stub类 实现ipc**

**(4)我们定义的中间人对象 直接继承stub**

**(5)想要保证2个应用程序的aidl文件是同一个包名 要求aidl文件所在包名相同**

**(6)获取中间人对象Stub.asinterface(Ibinder obj)**

**AIDL是线性安全的，由框架来维护其线性安全**

# 内容提供者contentprovider

## 内容提供者特点介绍

**Android四大组件之一，没有图形界面**

**共享数据给第三方应用程序，统一数据访问方式**

**线性安全，Android系统为每个 ContentProvider提供一个实例（单列模式），通过ContentResolver访问ContentProvider**

**ContentProvider实现流程**

**[1]定义内容提供者 定义一个类继承contentProvider 在清单文件中需配置authorities 属性**

**[2]在清单文件里面配置一下**

**[3] 定义一个urimatcher(uri路径匹配器)**

**private static final UriMatcher *sURIMatcher* = new UriMatcher(UriMatcher.*NO\_MATCH*);**

**[4]写一个静态代码块 添加匹配规则**

**sURIMatcher.addURI(authority：与清单文件中authorities 一致, path:需要暴露的名称, code：匹配码)**

**[5]按照我们添加的匹配规则 暴露想暴露的方法**

**[6]在onCreate 初始化数据库帮助类对象**

**[7]如果你发现如下log日志 就说明内容提供者写的没有问题**

09-11 02:02:31.142: I/ActivityThread(16636): Pub com.itheima.provider: com.itheima.db.AccountProvider

**注意：只要是通过内容提供者暴露出来的数据 其他应用访问的方式都是一样的 就是通过内容解析者ContentResolver**

**注意：访问内容提供者 访问路径的时候 需加上协议 "content://**

**例：Uri uri = Uri.*parse*("content://com.itheima.provider/query");**

**通过content对象获取内容解析者 然后通过内容解析者(ContentResolver)可以获取内容提供者(ContentProvider)暴露出来的方法**

**ContentResolver contentResolver = getContentResolver();**

## 内容提供者Uri 的书写规范如下所示：



* **schema，用来说明一个ContentProvider 控制这些数据。"content://"**

**这是个固定写法，注意别写错。**

* **主机名或授权Authority：它定义了是哪个ContentProvider 提供这些数据。**
* **path：路径，URI 下的某一个Item。**
* **ID：通常定义Uri 时使用”#”号占位符代替, 使用时替换成对应的数字**
* **content://com.itheima.provider/person/#：#表示数据id（#代表任意数字）**
* **content://com.itheima.provider/person/\*：\*来匹配任意文本**

## 查询手机中的短信数据

**短信存放在data/data/com.android.providers.telephony/databases/mmssms.db 数据库中**

**Sms表中 address: 号码 date：时间 body：短信内容**

**Uri uri = Uri.parse("content://sms/"); sms通过查看系统短信的源码 smsProvider**

**短信的数据库已经通过内容提供者暴露出来 所以我们直接通过内容解析者查询**

**Cursor cursor = getContentResolver().query(uri, new String[]{"address","date","body"}, null, null, null);**

**插入短信到手机**

**ContentValues values = new ContentValues();**

**values.put("address", "18632525");**

**values.put("body", "过来一趟");**

**values.put("date", System.currentTimeMillis());**

**getContentResolver().insert(uri, values)**

## 查询手机联系人

**联系人存放在data/data/com.android.providers.contacts/databases/contacts2.db 数据库中**

**data表  data1列里面存的是所有联系人的信息**

**raw\_contacts 表中的 contact\_id 列是用来区分一共有几条联系人信息 raw\_contacts 表中的 contact\_id 列对应data表中的raw\_contact\_id**

**mimetype\_id 列是用来区分数据类型  1：表示联系人邮箱 5：表示联系人电话号码 7：表示联系人姓名 8表示：联系人地址**

**row\_contacts表 中contact\_id就是data表的 raw\_contact\_id**

**查询联系人的步骤**

**[1]先查询row\_contacts表 的contact\_id列 知道一共有几条联系人**

**Uri.parse("content://com.android.contacts/raw\_contacts");**

**系统源码Uri命名规则字段与表名相同**

**例：Cursor cursor = context.getContentResolver().query(uri,new String[] { "contact\_id" }, null, null, null);**

**[2]根据contact\_id去查询data表  查询data表的data1和mimetype\_id(区分数据类型)**

**查询data表 其实查询的是view\_data 的视图 而view\_data表中 mimetype\_id 指的是mimetype**

**view\_data 是由data表和mimetype表的组合**

**例：contactsBean.id = contact\_id;**

**// 根据contact\_id去查询data表  查询data1列和mimetype(区分数据类型)**

**Cursor cursorData = context.getContentResolver().**

**query(uriData, new String[] { "data1", "mimetype" },"raw\_contact\_id=?", newString[] { contact\_id },null);**

**注意：删除联系人在数据库的data表中还会存在联系人数据 只是把raw\_contacts表中的 contact\_id设置为null**

## 插入手机联系人

**插入联系人的步骤**

**Uri uri = Uri.*parse*("content://com.android.contacts/raw\_contacts");**

**Uri uriData = Uri.*parse*("content://com.android.contacts/data");**

**1.在插入联系人id的之前查询一下 row\_contact 一共有几条数据 加+1就是插入联系人的id**

**Cursor cursor = getContentResolver().query(uri, null, null, null, null);**

**int count = cursor.getCount();同**

**2.先往row\_contacts表插入数据   contact\_id**

**values.put("contact\_id", contact\_id);**

**3.在往data表里面插入数据  data1**

**4.插入的数据之前要告诉数据库 属于第几条联系人 和 数据类型**

**ContentValues nameValues = new ContentValues();**

**nameValues.put("raw\_contact\_id", contact\_id);**

**nameValues.put("data1", name);**

**nameValues.put("mimetype", "vnd.android.cursor.item/name");**

**getContentResolver().insert(uriData, nameValues);**

**常用API介绍**

**\* Uri：（统一资源标识符）表示要操作的数据**

**由两部分组成：内容提供者的标识、具体要访问什么，也就是路径**

**content://cn.itcast.sqlite.provider.books/student/**

**\* UriMatch： 用来匹配Uri的 工具类**

**UriMatch.addUri(Uri);//注册Uri**

**UriMatch.match(Uri);匹配Uri**

**\* ContentUris： Uri内容解析**

**ContentUris.withAppendedId(Uri,id) : 在Uri尾部添加id**

**ContentUris.parseIdUuri) :解析Uri，获取Uri的id部分**

## 内容观察者

**内容观察不是四大组件不需要在清单文件里配置**

**定义内容观察者**

**1.注册一个内容观察者**

**Uri uri = Uri.parse("content://sms/");**

**getContentResolver().registerContentObserver(uri, true, new MyContentObserver(new Handler()));**

**定义一个类继承ContentObserver**

**private class MyContentObserver extends ContentObserver{**

**public MyContentObserver(Handler handler) {**

**super(handler);**

**}**

**//此方法当观察的内容提供者发生改变的时候调用**

**public void onChange(boolean selfChange) {**

**super.onChange(selfChange);**

**2.在内容提供者中发生改变时，发送消息**

**发送一条消息 说明说明数据库被操作了**

**getContext().getContentResolver().notifyChange(uri, null);**

# 计算机表示图形的几种方式

**多媒体:(包含文字 图片 音频 视频)**

**图形的大小 = 图片的总像素 \* 每个像素的大小**

**单色  每个像素最多可以表示2种颜色  只需要使用长度为1的二进制位来表示  那么每个像素占1/8byte**

**16色  每个像素最多可以表示16种颜色 0000 - 1111 那么只需要使用长度为4的二进制表示 那么每个像素占1/2个byte**

**256色 每个像素最多可以表示256种颜色  0000 0000 - 1111 1111  那么只需要使用长度8的二进制位表示 那么每个像素占1byte**

**24位 rgb**

**r 1byte  0-255**

**g 1byte  0-255**

**b 1byte  0-255    一个像素占3byte**

**Android采用的是png格式**

**三种像素如下：**

* **ARGB\_4444 : 2bytes 每个像素占据2 个字节**
* **ARGB\_8888 : 4bytes 每个像素占据4 个字节**
* **RGB\_565 : 4bytes 每个像素占据4 个字节**

# 缩放加载加载大图片

**在Android下采用ARGB来表示颜色  每个像素占4byte**

**宽缩放: 图片的宽/手机的宽**

**高缩放: 图片的高/手机的高**

**需要考虑的问题 (1)动态获取图片的分辨率**

**(2)动态获取手机分辨率**

**1.通过WindowManager 对象获取手机的分辩率**

**WindowManager wm = (WindowManager) getSystemService(*WINDOW\_SERVICE*);**

**获取手机的宽和高 第一种方法 已过时**

**int width = wm.getDefaultDisplay().getWidth();//得到宽**

**int height = wm.getDefaultDisplay().getHeight();//得到高**

**获取手机的宽和高 第二种方法**

**Point point = new Point();**

**wm.getDefaultDisplay().getSize(point);**

**screen\_width = point.x;**

**screen\_height = point.y;**

**2.获取图片的宽和高**

**创建bitmap工厂类的 配置参数**

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

**返回值如果设置为true 解码器 则返回null(则没有bitmap对象，但是可以查询bitmap的信息 宽和高)**

**options.inJustDecodeBounds=true;**

**Bitmap bitmap = BitmapFactory.*decodeFile*("/sdcard/dog.jpg",options);**

**获取图片的宽和高**

**int img\_width = options.outWidth;**

**int img\_height = options.outHeight;**

**3计算缩放比**

**int scale = 1;//定义默认的缩放比**

**int scale\_x = img\_width/screen\_width;**

**int scale\_y= img\_height/screen\_height;**

**if(scale\_x>scale\_y && scale\_x>scale){**

**scale= scale\_x;**

**}**

**if(scale\_y>scale\_x && scale\_y>scale){**

**scale=scale\_y;**

**}**

**4.按照缩放比显示图片 解析位图 显示图片**

**options.inSampleSize=scale;**

**开始解析位图**

**options.inJustDecodeBounds=false;**

**Bitmap bitmap\_scale = BitmapFactory.*decodeFile*("/sdcard/dog.jpg",options);**

# 创建原图的副本(**图形处理**)

**在Android 中 原图是不可以修改 需修改图片 必须是原图的副本**

**创建模板**

**width:宽, height:高, config:配置 需要和原图一样**

**Bitmap bitmap\_copy = Bitmap.*createBitmap*(bitmap\_src.getWidth(), bitmap\_src.getHeight(), bitmap\_src.getConfig());**

**2设置一个画布 以bitmap\_copy为模板**

**Canvas canvas = new Canvas(bitmap\_copy);**

**3 创建一个画笔**

**Paint paint = new Paint();**

**4 开始作画 bitmap：参考原图作画 matrix:矩阵(几行几列)**

**canvas.drawBitmap(bitmap\_src, new Matrix(), paint);**

**5.对图片进行旋转**

**Matrix matrix = new Matrix();**

**degrees:角度, px：X坐标 , py：Y坐标**

**matrix.setRotate(degrees, bitmap\_src.getWidth()/2, bitmap\_src.getHeight()/2);**

**canvas.drawBitmap(bitmap\_src, matrix, paint);**

**不能在子线程更新ui (这个方法中的逻辑一定在主线程运行)**

**runOnUiThread(new Runnable() {**

**public void run() {**

**//4.将bitmap\_copy 显示到 lv\_copy 上**

**lv\_copy.setImageBitmap(bitmap\_copy)**

**});**

**SystemClock.*sleep*(1000);//停一秒**

**缩放**

**matrix.setScale(0.5f, 0.5f);**

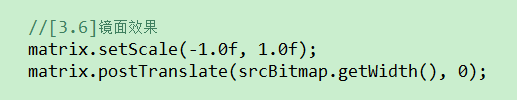
**位移**

**matrix.setTranslate(20, 0);**

**倒影效果(Android 中坐标系 向右为正 向下为正)**

****

**镜面效果**

****

## 画画板小案例

1. **创建原图模板**

**2.控件设置触摸事件(实现不停画线功能)**

**lv.setOnTouchListener(new OnTouchListener() {**

**int startX = 0;**

**int startY = 0;**

**// 当触摸控件时执行该方法**

**@Override**

**public boolean onTouch(View v, MotionEvent event) {**

**3获取手指触摸类型**

**int action = event.getAction();**

**4.判断是什么触摸类型**

**switch (action) {**

**case MotionEvent.*ACTION\_DOWN*://触摸按下**

**5.获取触摸按下坐标**

**startX = (int) event.getX();**

**startY = (int) event.getY();**

**break;**

**case MotionEvent.*ACTION\_MOVE*://触摸移动**

**6.获取获取触摸停止坐标**

**int stopX = (int) event.getX();**

**int stopY = (int) event.getY();**

**7.开始画线**

**// startX:开始X点坐标, startY:开始Y点坐标, stopX:结束X点坐标, stopY:结束Y点坐标, paint**

**canvas.drawLine(startX, startY, stopX, stopY, paint);//画线**

**8.更新坐标**

**startX=stopX;**

**startY=stopY;**

**System.*out*.println("更新坐标");**

**//11.更新UI**

**lv.setImageBitmap(bitmap\_copy);**

**break;**

**case MotionEvent.*ACTION\_UP*://触摸抬起**

**return true;//true 触摸事件全部执行**

**设置画笔颜色**

**paint.setColor(Color.RED);**

**设置画笔的宽度**

**paint.setStrokeWidth(15);**

**保存**

**format 保存图片的格式**

**quality 保存照片的质量**

**copyBitmap.compress(CompressFormat.PNG, 100, fos);**

## 撕衣服游戏

**实现原理**

**使用帧布局叠加2 个ImageView，每个ImageView 负责显示一张图片，一张图片有衣服，一张图片没有衣服，没有衣服的图片放置在下面，有衣服的图片放置在上面。给上面的ImageView 设置触摸的事件，当手指触摸到图片上时，将手指触摸到的点周边的图片的像素点设置为透明的，这样下面的图片就一点一**

**点显示出来了，从而有一种“撕衣服”的感觉。**

**在编写该案例的时候应该注意的事项如下：**

**1.触摸事件onTouch 的返回值必须设置为true，否则触摸的事件将不被处理**

**2.使用BitmapFactory 的decodeResouces 方法得到的图片是没有透明度的，即图片格式为RGB\_565,所以若想能够修改透明度，需要使用Canvas 对象对图片进行重绘，重新绘制的图片格式采用ARGB。**

**3.加载图片时需要对其进行一下压缩，防止图片与控件大小不匹配，导致触摸时点对不上，达不到触摸哪里就设置哪里的像素点透明的效果。一般图片宽高不超过屏幕宽高即可。**

1. **将操作的图片转换成Bitmap**

**Bitmap bitmap = BitmapFactory.*decodeResource*(getResources(), R.drawable.*pre19*);**

1. **创建原图的副本**
2. **将副本显示到控件上**

**4.控件设置触摸事件**

**lv.setOnTouchListener(new OnTouchListener() {**

**//每触摸一次都会执行该方法以**

**@Override**

**public boolean onTouch(View v, MotionEvent event){**

**int action = event.getAction();**

**switch (action) {**

**case MotionEvent.*ACTION\_MOVE*://触摸移动**

**for (int i = -7; i < 7; i++) {**

**for (int j = -7; j <7; j++) {**

**//一次撕一个圆形**

**if(Math.*sqrt*(i\*i+j\*j)<7){**

**//5.触摸变透明(一次改变一个像素)**

**try {**

**bitmap\_copy.setPixel((int)event.getX()+i, (int)event.getY()+j,Color.*TRANSPARENT*);**

**} catch (Exception e) {**

**//6.更新LV**

**lv.setImageBitmap(bitmap\_copy);**

## 音乐播放器(**MediaPlayer** )

**Android 官方提供了MediaPlayer 核心类，用于播放音乐.视频 其状态流程如下图所示。MediaPlayer 必须严格按照状态图操作，否则就会出现错误，这些错误都是底层抛出，严格按照状态图操作的话一般就不会出问题。**

**使用MediaPlayer 播放音乐的核心方法如下所示：**

**1. MediaPlayer player = new MediaPlayer(); 创建对象**

**2. player.reset(); 重置为初始状态**

**3. player.setAudioStreamType(AudioManager.STREAM\_MUSIC);声音流类型**

**4. player.setDataSource(“/mnt/sdcard/test.mp3”); 设置音频源**

**5. player.prepare(); 准备**

**6. player.start(); 开始或恢复播放**

**7. player.pause(); 暂停播放**

**9. player.stop(); 停止播放**

**10. player.release(); 释放资源**

**11.seekTo 设置播方音乐指定位置**

**12.getDuration()获取总时长**

**13.getCurrentPosition() 获取当前播放的位置**

**14.getProgress() 获取拖动结束之后的位置**

**15.prepareAsync() 异步 播放网络音乐用此方法**

**16.setDisplay() 设置播放视频的内容 SurfaceHolder 是用来维护视频播放的内容**

**17.isPlaying() 正在播放**

**18.setLooping 是否循环**

**setOnCompletionListener 播放完成监听事件**

**onCompletion()播放完成会执行此方法**

**setOnPreparedListener 资源准备完成监听**

**onPrepared() 当播放资源准备完成时候执行此方法 player.start(); 开始播放可以在此方法中执行**

**流程图如下：**



**Timer计时器**

**[cancel](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/util/Timer.html" \l "cancel())()终止任务。**

**[schedule](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/util/Timer.html" \l "schedule(java.util.TimerTask, java.util.Date))([TimerTask](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/util/TimerTask.html" \o "java.util 中的类) task, [Date](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/util/Date.html" \o "java.util 中的类) time) 安排在指定的时间执行指定的任务。**

**[schedule](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/util/Timer.html" \l "schedule(java.util.TimerTask, java.util.Date, long))([TimerTask](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/util/TimerTask.html" \o "java.util 中的类) task, [Date](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/util/Date.html" \o "java.util 中的类) firstTime, long period) 安排指定的任务在指定的时间开始进行重复的*固定延迟执行***

**SeekBar(播放音乐控件)**

**setMax() 设置SeekBar最大进度**

**setProgress() 设置SeekBar当前进度**

**setOnSeekBarChangeListener (SeekBar 设置进度条改变监听)**

**onStopTrackingTouch() 当停止托动执行此方法**

**onStartTrackingTouch() 当开始托动执行此方法**

**onProgressChanged() 当进度条发生改变执行此方法**

## SurfaceView(显示视频的View)

**surfaceview 控件是一个重量级控件**

**内部维护了2个线程**

**A 获取数据  负责显示 (相互转换)**

**B 负责显示  获取数据**

**可以直接在子线程更新ui   与进度相关的控件可以直接在子线程更新ui**

**整个过程：**

1. **继承SurfaceView并实现SurfaceHolder.Callback接口 ---->**
2. **SurfaceView.getHolder()获得SurfaceHolder对象 ---->**
3. **SurfaceHolder.addCallback(callback)添加回调函数---->**
4. **SurfaceHolder.lockCanvas()获得Canvas对象并锁定画布---->**
5. **Canvas绘画 ---->**
6. **SurfaceHolder.unlockCanvasAndPost(Canvas canvas)结束锁定画图，并提交改变，将图形显示。**

### SurfaceHolder

**这里用到了一个类SurfaceHolder,可以把它当成surface的控制器，用来操纵surface。处理它的Canvas上画的效果和动画，控制表面，大小，像素等。**

**几个需要注意的方法：**

**(1)、abstract void addCallback(SurfaceHolder.Callback callback); 播放状态监听**

**// 给SurfaceView当前的持有者一个回调对象。**

**(2)、abstract Canvas lockCanvas();**

**// 锁定画布，一般在锁定后就可以通过其返回的画布对象Canvas，在其上面画图等操作了。**

**(3)、abstract Canvas lockCanvas(Rect dirty);**

**// 锁定画布的某个区域进行画图等..因为画完图后，会调用下面的unlockCanvasAndPost来改变显示内容。**

**// 相对部分内存要求比较高的游戏来说，可以不用重画dirty外的其它区域的像素，可以提高速度。**

**(4)、abstract void unlockCanvasAndPost(Canvas canvas);**

**// 结束锁定画图，并提交改变。**

**surfaceDestroyed() 当Surfaceview销毁的时候调用**

**surfaceCreated() 当Surfaceview 初始化调用**

**surfaceChanged() 当Surfaceview 表面发生改变调用**

**播放视频 － 设置SurfaceView控件 － 通过SurfaceView.getHolder() 获取SurfaceHolder(用于维护播放视频) － SurfaceHolder.addCallback()设置监听**

**在surfaceDestroyed()－获取播放的当前进度 － 在surfaceCreated() 中seekTo() 从上次位置开始播放**

## Vitamio框架的使用

**注意： VideoView**

**1.这个控件就是对surfaceview 和 meidiaplayer进行封装**

**2.meidiaplayer 播放视频他只支持 3gp MP4格式**

**1 引入vitamio框架 以library、**

**2 在布局中定义VideoView**

**<io.vov.vitamio.widget.VideoView**

**android:id="@+id/vv"**

**android:layout\_width="match\_parent"**

**android:layout\_height="match\_parent"**

**/>**

**3 mainactivity代码**

**插件vitamio框架检查是否可用**

**if (!LibsChecker.checkVitamioLibs(this)) {**

**return;**

**}**

**final VideoView vv = (VideoView) findViewById(R.id.vv);**

**vv.setVideoPath("http://192.168.1.2:8080/haha.avi");**

**vv.setOnPreparedListener(new OnPreparedListener() { // 准备监听**

**@Override**

**public void onPrepared(MediaPlayer mp) {**

**vv.start();**

**}**

**});**

**//设置video的控制器 进度条**

**vv.setMediaController(new MediaController(this));**

**4 一定要在清单文件初始化InitActivity**

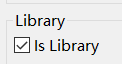
**<activity android:name="io.vov.vitamio.activity.InitActivity"></activity>**

**注意 ：vitamio框架**

**meidiaplayer**

**videoview (底层是meidiaplayer )**

**Ffmpeg 开源项目 (vitamio 视频解码的原理Ffmpeg C语言编写)**

**如何将一个项目变成一个内库： 程序项目右键 —— Properties —— 对话框中选择 Android   OK 就可以了**

# S**oundPool(播放密集 短暂的音效)**

**如果应用程序经常播放密集、急促而又短暂的音效（如游戏音效）那么使用MediaPlayer显得有些不太适合了。因为MediaPlayer存在如下缺点：**

**1) 延时时间较长，且资源占用率高。**

**2) 不支持多个音频同时播放。**

**Android中除了MediaPlayer播放音频之外还提供了SoundPool来播放音效，SoundPool使用音效池的概念来管理多个短促的音效，例如它可以开始就加载20个音效，以后在程序中按音效的ID进行播放。**

**SoundPool主要用于播放一些较短的声音片段，与MediaPlayer相比，SoundPool的优势在于CPU资源占用量低和反应延迟小。另外，SoundPool还支持自行设置声音的品质、音量、 播放比率等参数。**

**SoundPool提供了一个构造器，该构造器可以指定它总共支持多少个声音（也就是池的大小）、声音的品质等。构造器如下：**

**SoundPool(int maxStreams, int streamType, int srcQuality)：第一个参数指定支持多少个声音；第二个参数指定声音类型：第三个参数指定声音品质。一旦得到了SoundPool对象之后，接下来就可调用SoundPool的多个重载的load方法来加载声音了。**

**SoundPool提供了如下4个load方法：**

* **int load(Context context, int resld, int priority)：从 resld 所对应的资源加载声音。**
* **int load(FileDescriptor fd, long offset, long length, int priority)：加载 fd 所对应的文件的offset开始、长度为length的声音。**
* **int load(AssetFileDescriptor afd, int priority)：从afd 所对应的文件中加载声音。**
* **int load(String path, int priority)：从path 对应的文件去加载声音。**

**上面4个方法中都有一个priority参数，该参数目前还没有任何作用，Android建议将该 参数设为1，保持和未来的兼容性。**

**上面4个方法加载声音之后，都会返回该声音的的ID，以后程序就可以通过该声音的ID 来播放指定声音。**

**SoundPool提供的播放指定声音的方法：**

**int play(int soundID, float leftVolume, float rightVolume, int priority, int loop, float rate)：该方法的第一个参数指定播放哪个声音；leftVolume、rightVolume指定左、右的音量：priority指定播放声音的优先级，数值越大，优先级越高；**

**loop指定是否循环，0为不循环，-1为循环；**

**rate指定播放的比率，数值可从0.5到2， 1为正常比率。**

**为了更好地管理SoundPool所加载的每个声音的1D，程序一般会使用一个HashMap对象来管理声音。**

**归纳起来，使用SoundPool播放声音的步骤如下：**

**1) 调用SoundPool的构造器创建SoundPool的对象。**

**2) 调用SoundPool对象的load()方法从指定资源、文件中加载声音。最好使用HashMap< Integer, Integer>来管理所加载的声音。**

**3) 调用SoundPool的play方法播放声音。**

# 传感器的使用(**SensorManager**)

**1. 获取SensorManager管理器**

**SensorManager sm = (SensorManager) getSystemService(SENSOR\_SERVICE);**

**2. 设置需要获取传感器的type**

**SENSOR\_TYPE\_ACCELEROMETER       1 //加速度**

**SENSOR\_TYPE\_MAGNETIC\_FIELD      2 //磁力**

**SENSOR\_TYPE\_ORIENTATION          3 //方向**

**SENSOR\_TYPE\_GYROSCOPE           4 //陀螺仪**

**SENSOR\_TYPE\_LIGHT               5 //光线感应**

**SENSOR\_TYPE\_PRESSURE             6 //压力**

**SENSOR\_TYPE\_TEMPERATURE          7 //温度**

**SENSOR\_TYPE\_PROXIMITY           8 //接近**

**SENSOR\_TYPE\_GRAVITY             9 //重力**

**SENSOR\_TYPE\_LINEAR\_ACCELERATION 10//线性加速度**

**Sensor sensor = sm.getDefaultSensor(type);**

**3. 注册监听**

**sm.registerListener(listener, sensor, SensorManager.SENSOR\_DELAY\_NORMAL);**

**private class MyLintener implements SensorEventListener{**

**//当传感器数据变化的调用的方法**

**@Override**

**public void onSensorChanged(SensorEvent event) {**

**//1 如果是光线传感器，event.values[0] 代表的是光感度**

**//2 如果是方向传感器，event.values[0]**

**}**

**//当传感器精度发生变化的时候调用的方法**

**@Override**

**public void onAccuracyChanged(Sensor sensor, int accuracy) {**

**}**

**}**

**4. 在退出的时候要记得取消监听**

**sm.unregisterListener(listener);**

# **调用系统照相机和摄像机功能**

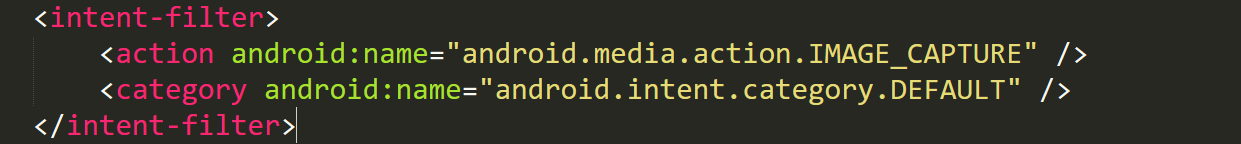
**调用系统摄像头进行拍照和摄像是通过隐式启动系统Activity 实现的，无需给自己的工程添加权限，直接调用即可。因此我们只需知道系统照相机和摄像机Activity 的action 和category 就可以了。**

**步骤：**

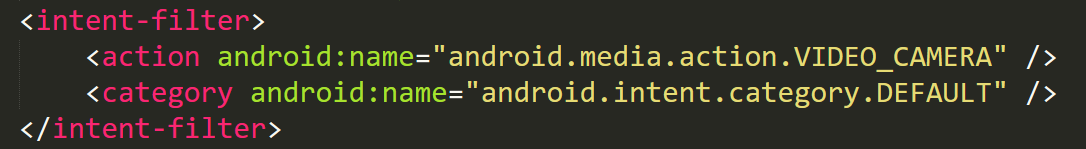
**1. 打开Android 源码，查看”\packages\apps\”文件文件目录下的Camera 应用，即系统摄像头的应用程序。打开其清单文件文件，查看其Activity 的action 和category 信息。**

**2. Camera 类的action 和category 如下：**

**照相机的意图过滤器**

****

**摄像机的意图过滤器**

****

**照相 创建意图**

**Intent intent = new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);**

**存放图片的路径**

**File file = new File(Environment.getExternalStorageDirectory().getPath(), "haha.png");**

**intent.putExtra(MediaStore.EXTRA\_OUTPUT, Uri.fromFile(file));**

**开启一个activity 并获取结果**

**startActivityForResult(intent, 1);**

**录像 创建意图**

**Intent intent = new Intent(MediaStore.ACTION\_VIDEO\_CAPTURE);**

**存放图片的路径**

**File file = new File(Environment.getExternalStorageDirectory().getPath(), "haha.3gp");**

**intent.putExtra(MediaStore.EXTRA\_OUTPUT, Uri.fromFile(file));**

**开启一个activity 并获取结果**

**startActivityForResult(intent, 2);**

# Fragment

**Fragment :碎片、片段**

**引入fragments主要目的是用在大屏幕设备上支持更加动态和灵活的UI设计，**

**一个Fragment代表一个用户接口的行为或者部分行为,把几个Fragment混合到一个Activity中,可以创建一个多个页面的UI并可以在多个Activity中复用一个Fragment，碎片必须总是嵌入到一个活动（activity）中，并且它的生命周期直接受到Activity的生命周期的影响。**

**把fragment 想象成一个activity的模块化区域, 有它自己的生命周期, 接收属于它的输入事件, 并且可以在activity运行期间添加和删除**

**\* DialogFragment ：对话框碎片**

**用于显示一个对话框窗口的Fragment，它浮于Activity窗口上面，该对话框的显示依赖于Fragment的生命状态**

**\* 实现步骤**

**- 创建一个类，继承DialogFragment，重写onCreateDialog方法，在该方法中完成对话框的创建**

**- 在Activity中创建DialogFragment实例，并调用show()方法完成对话框的显示**

**可以通过DialogFragment.setArguments()\getArguments()传递和接收参数**

**布局中添加Fragment 需放在 ViewGroup 中 LinearLayout FrameLayout RelativeLayout 这几个布局都继承 ViewGroup**

**所有的控件都继承 View**

## Fragment 的创建

**创建Fragment 的步骤如下：**

**1. 为Fragment 定义一个布局**

**2. 定义类MyFragment 继承Fragment**

**3. 重写MyFragment 类中的onCreateView 方法，该方法返回一个View 对象作为当前Fragment 的布局。**

**例： View view = inflater.inflate(R.layout.*fragment\_contacts*, null);**

**4.创建Fragment 实例 开启事物**

**FragmentManager manager = getFragmentManager();**

**开启事物**

**FragmentTransaction transaction = manager.beginTransaction();**

**添加fragment**

**transaction.replace(R.id.*ll*, new FragmentContacts());**

**提交**

**transaction.commit();**

**fragment兼容低版本的写法**

**[1]定义fragment继承V4包中的Fragment (导入 import android.support.v4.app.Fragment;)**

**[2]定义的activity要继承v4包中的FragmentActivity (MainActivity extends FragmentActivity)**

**[3]通过这个方法getSupportFragmentManager  获取Fragment的管理者**

**通过getsupportFragmentManager();获取Fragment的管理者**

**FragmentManager supportFragmentManager = getSupportFragmentManager();**

**Fragment之间通信**

**事物中的transaction 添加fragment 增加标记**

**transaction.replace(R.id.ll1, new Fragment1(),"f1");**

**两个Fragment之间公共的桥梁是 Activity 通过标记获取别一个 Fragment实例**

**Fragment2 f2 = (Fragment2) getActivity().getFragmentManager().findFragmentByTag("f2");**

## Fragment的生命周期



**1. onAttach：绑定到activity**

**2. onCreate：创建fragment**

**3. onCreateView： 创建fragment 的布局**

**4. onActivityCreated： activity 创建完成后**

**5. onStart： 可见, 不可交互**

**6. onResume： 可见, 可交互**

**7. onPause： 部分可见, 不可交互**

**8. onStop：不可见**

**9. onDestroyView： 销毁fragment 的view 对象**

**10. onDestroy： fragment 销毁了**

**11. onDetach： 从activity 解绑了**

**由于Fragment 是跟Activity 绑定的，因此Activity 的生命周期是“严重”影响Fragment 的生命周期的**

**Fragment比Activity多了几个额外的生命周期回调方法：  
**onAttach(Activity)**当Fragment与Activity发生关联时调用。  
**onCreateView(LayoutInflater, ViewGroup,Bundle)**创建该Fragment的视图  
**onActivityCreated(Bundle)**当Activity的onCreate方法返回时调用  
**onDestoryView()**与onCreateView想对应，当该Fragment的视图被移除时调用  
**onDetach()**与onAttach相对应，当Fragment与Activity关联被取消时调用  
注意：除了onCreateView，其他的所有方法如果你重写了，必须调用父类对于该方法的实现，**

**[1]使用Fragment必须重写onCreateView方法**

**[2]还可以重写一个ondestroy方法 做一些收尾的工作**

# menu菜单

**[1]添加菜单方式 通过一个布局 在res下 meun目录下创建一个布局**

**在onCreateOptionsMenu( )方法中**

**getMenuInflater().inflate(R.menu.main, menu);**

**[2]动态的添加**

**添加菜单**

**menu.add(0, 1, 0, "后退");**

**menu.add(0, 2, 0, "前进");**

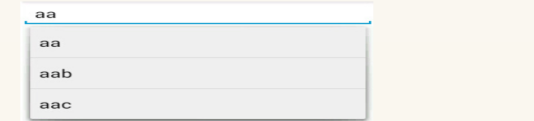
**具体点击的是menu的哪个条目 需要重写这个方法 onOptionsItemSelected 方法**

**[3]点击menu弹出一个对话框 如果你想实现自己定义的view 效果重写onMeunOpened方法**

**当点击menu按键的时候会先执行这个方法 onMenuOpened(int featureId, Menu menu) 默认返回true**

**此方法的返回值如果是true 会执行 onCreateOptionsMenu( ) 如果返回false 则不会执行 onCreateOptionsMenu( )**

# AutoCompleteTextView控件



1. **布局文件中写 AutoCompleteTextView**

**android:completionThreshold="1" 设置按多少个字显示提示**

**2.找到控件**

**AutoCompleteTextView actv = (AutoCompleteTextView) findViewById(R.id.actv);**

**3.创建数据适配器**

**ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,android.R.layout.simple\_dropdown\_item\_1line, COUNTRIES);**

**4.设置数据适配器**

**actv.setAdapter(adapter);**

# 应用程序的反编译

**Apktools :apk反编泽工具**

**dex2jar :Android 所有类文件最终打包成dex文件 dex2jar将dex 转换成jar包  jd.exe：就看jar包源码(程序源码)**

**Android逆向助手 (只能查看android apk格式源码)**

# 总结学过的manager

**[1]smsManager 发短信**

**[2]telephoneManager 电话的管理者**

**[3]windowManager 窗口的管理者 (获取屏幕分辨率)**

**[4]fragmenManager fragment的管理者**

**[5]NotificationManager 通知的管理者**

**[6] SensorManager 传感器的管理者**

# SVN

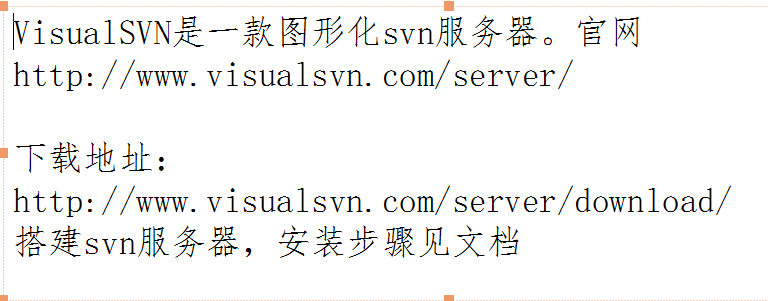
## 为什么需要版本控制软件

1. **代码的冻结**
2. **需求频繁的操作**
3. **权限设置**

**常见的版本控制软件**

**Cvs svn (集中式 需要服务器) ClearCase (IBM 开发) vss (微软) git(分步式 不需要服务器 支持离线工作)**

1. **虚拟机的安装**

****

****

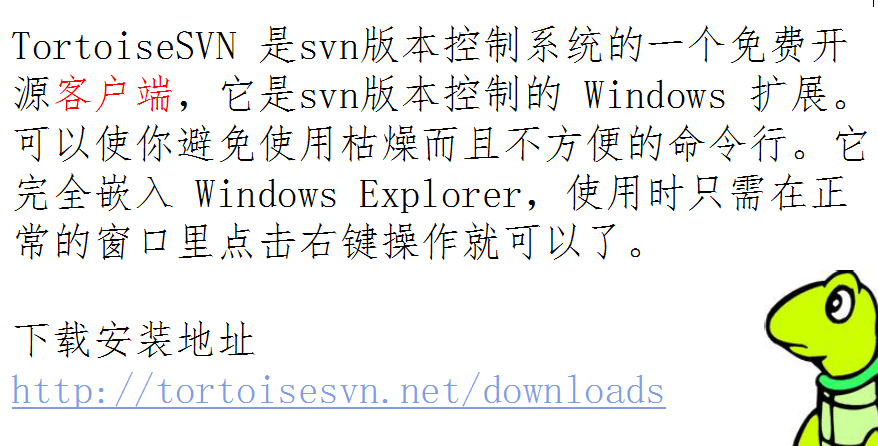
**Svn服务端软件的安装**

****

**[1]验证服务器安装成功 在浏览器敲 https://127.0.0.1**

**[2]其他电脑通过访问 https://192.168.13.250 (虚拟机IP地址设为 192.168.13.250也可自定义设置)**

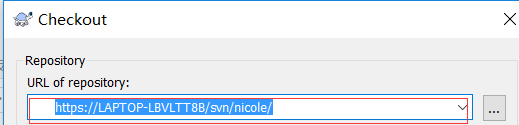
## Svn客服端软件的安装

****

****

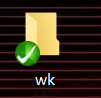
## 单用记代码的控制

**[1]让当前工作控件和svn服务器建立关联**

**(在服务端建好的目录下右击 选择Copy URL to Clipboard 复制当前URL地址 然后再客服端右键选择 SVN Checkout... 在弹出的对话框中**

**<https://后面修改ip地址> ok )**

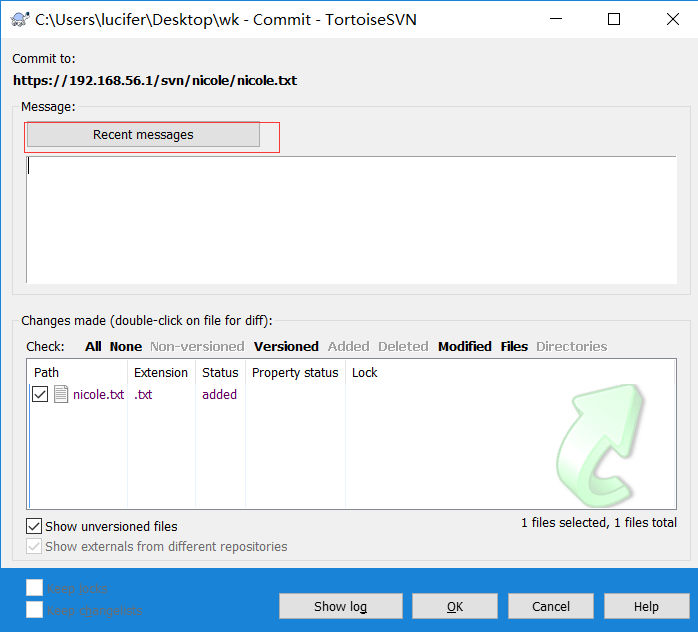
**[2]如果发现当前工作控件有一个.svn目录就说明当前工作空间和仓库建立关联**

**[3]当前的这个工作控件会有一个绿色对勾 (如果没绿色对勾 可以把当前目录册到回收站然后还原 或 把桌面进程杀死 (explorer.exe )然后点任务管理器 文件 新键任务)**

**[4]点击add 会变成一个蓝色+ 计划把文件加入到版本控制库中**

**(写好的文件 右键选择TortoiseSVN - Add )**

**[5]把文件提交到版本控制库中**

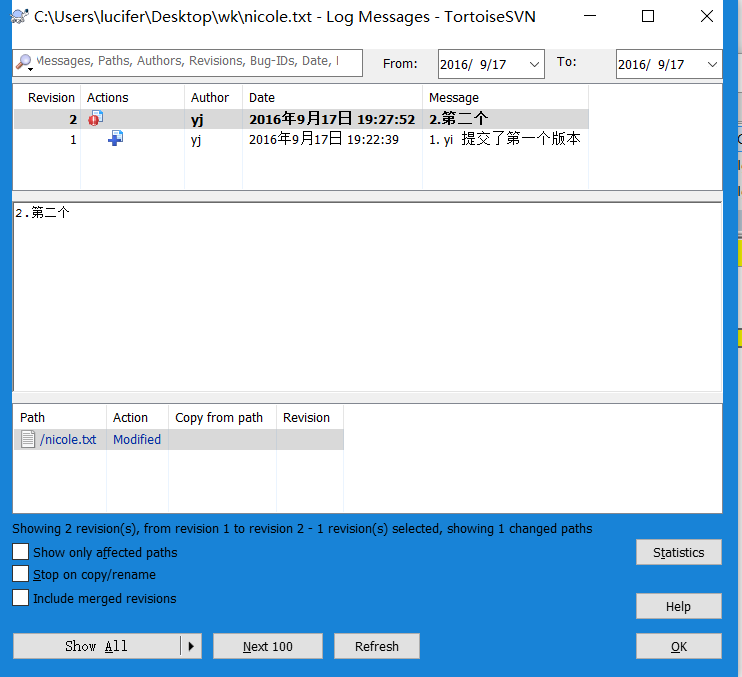
**(然后在文件上右击 选择 SVN Commit.... 在弹出的对话框中  )**

**注释**

**修改代码后提交 原文件会变成  在修改后的文件上右击 选择**

**服务器查看是否提交 在刷新下**

**服务器查看原来版本代码：在代码文件右击 选择 TortoiseSVN 选择 show log 对话框中会出现提交的所有版本代码**

**代码导出：在 在文件上右击 选择 Svae revision to... 选择保存的路径**

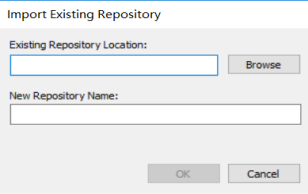
**需要导出的文件**

**代码丢失找回 文件上右击选择SVN Checkout... 输入ip地起址 就能查看所有代码版本代码**

**清除SVN 用户名数据 右击选择TortoiseSVN —— Settings ——对话框中选择——点击右边所有的——确定**

1. **版本库的备份与还原**

**服务器(仓库) 存放在 Repositories 文件中 可以备份**

**服务器丢失可以在 Repositories 右击选择 所有任务 —— Import Existing Repository... ——在对话框中 ——点击 选择存取目录**

**多用户代码的控制**

**提交代码到服务器 要先更新下(SVN Update)**

**解决冲突:文件右击 ——TortoiseSVN ——Edit conflicts ——对话框中解决冲突后——Save(保存)**

**SVN 不是专门用来解决冲突的 (如何避免 分模块开发)**

## svn常见图标

**[1]绿色对勾 代表和服务已经建立关联 (2)提交成功 更新成功等**

**[2]蓝色? 代表服务器不知道有这个文件**

**[3]蓝色+ 代表计划加入到版本控制库中**

**[4]黄色感叹号 代表遇到了冲突**

**[5]红色感叹号 代表修改了这个文件 需要提交到版本控制库中**

**[6]灰色对勾 代表文件只读(将文件变成只读的权限(添加属性)——文件右击 ——TortoiseSVN——点击Properties——对话框中选择——选择Needs-Lock**

**——选择——OK)**

**操作(修改)该文件需要右击——选择SVN Get Lock...**

**[7]锁在开发中慎重**

1. **svn提交代码的原则**

**[1]先更新在提交**

**[2]多提交**

**[3]不要提交不能通过编译的代码**

**[4]每次提交必须书写明晰的标注**

**[5]提交时注意不要提交本地自动生成的文件**

**[6]不要提交自己不明白的代码**

**[7]慎用锁定功能**

****

## 使用svn管理Android项目

**使用ignore 忽略你不想提交的目录：文件右击——TortoiseSVN——选择Unversion and add to ignore list——选择第一个(忽略的条目)**

**[1]使用客户端软件**

**[2]使用eclipse插件**

## svn的标准目录结构

**Trunk**

**这是SVN目录的主分支，表示日常开发中的项目，任何时候Trunk里包含的都是最新的开发代码。 这里的代码将会工作到你的下一个主要发布版本。**

**Trunk应该只被用来开发将会成为你的下一个重要版本的代码。**

**Branches 分支**

**Experimental branches**

**有时你想将某个新技术引进项目。这很好，但是你当然不想赌上你的整个项目。**

**Bug fix branches**

**分支也可以用于处理trunk或release branches里发现的严重的Bug。**

**Tags**

**一般情况下，tag，是用来做一个milestone的，不管是不是release，都是一个可用的版本。**

**Tags，一般是只读的。(项目备份——TortoiseSVN——Repo-browser——对话框中选择Trunk右击——选择Copy to ——对话框添加tags/1.0(命名))**

**项目合并：文件右击——TortoiseSVN——Merge....——Branches目录下文件选择合并**

## 使用eclipse演示企业开发流程(重点)

**[1]安装snv插件**

**[2]提供的feature 里面的jar包放到你自己eclipse的feature里面**

**[3]提供的plugin 里面的jar 放到你自己eclipse的plugin里面**

**[4]如果你点击你自己eclipse的windowpreference 下有一个svn说明安装成功**

**(创建帐户——右击创建帐号 帐户赋权限——项目文件右击——所有任务——Manage Security——添加)**

**使用eclipse 插件管理SVN 项目 ： 项目右击选择——对话框中选择Next ——输入URL地址**

**输入用户名密码 —— 选中项目 Finish**

**项目提交——右击选择 Team(所有操作都在该目录下)**

**新项目提交——Team——添加至版本控制(A)——提交**

1. **git简介**

**采用分步式版本库**

**msysgit是Windows版的Git，提供了命令行操作**

**下载地址：http://msysgit.github.io/**

**cywing --- jni 是在windows下完美模拟linux的指令**

**2.TortoiseGIT 是Git版本控制系统的一个免费开源客户端，它是git版本控制的 Windows 扩展。可以使你避免使用枯燥而且不方便的命令行。它完全嵌入 Windows Explorer，使用时只需在正常的窗口里点击右键操作就可以了。**

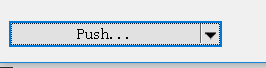
**下载地址：**

**https://code.google.com/p/tortoisegit/**

****

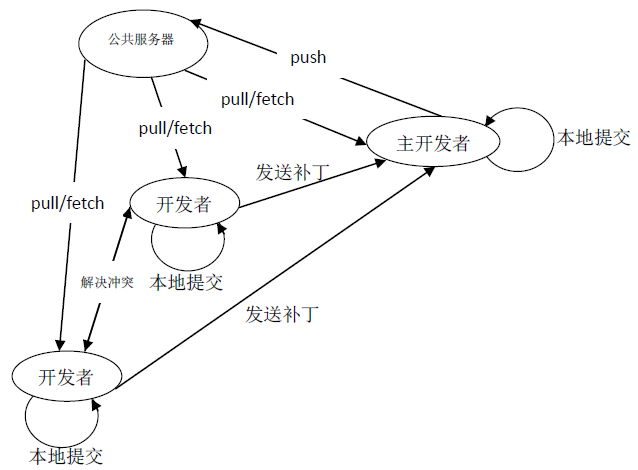
# git用户操作(用于开源项目)

**git 创建仓库 ——右击选择——Git Create repository here.... ——然后在项目文件右击——Git Commit>”master”.... 弹出界面(写注释提交)**

**可以提供到远端**

**与git 相关的操作都在 TortoiseGit Settings 中**

**创建补丁：文件右击——TortoiseGit——Create Patch Serial...(创建补丁)——Apply patch Serial...(应用补丁)**

****

**开源项目获取网站：**

**http://www.github.com git**

**http://code.google.com svn**