

Android 开发技巧合集

0、ANDROID常用类库说明	6
1、ANDROID文件系统与应用程序架构	7
1.1、ANDROID 文件系统	7
1.2、ANDROID应用程序架构	9
2、ANDROID应用程序结构	11
2.1、ACTIVITY	12
2.1.1、概述	12
2.1.2、Activity的生命周期	15
2.1.3、Activity 的创建	16
2.1.4、Activity 的跳转（含Bundle传值）	17
2.1.5、Activity 堆栈	18
2.1.6、Intent对象调用Activity实例	19
2.1.7、Activity透明	21
2.1.8、一次性关闭所有的Activity	22
2.1.9、PreferenceActivity 用法	22
2.1.10、Dialog风格的Activity	23
2.1.11、横竖屏切换不销毁当前Activity	23
2.2、INTENT RECEIVER	25
2.3、SERVICE	26
2.3.1、什么是Service	26
2.3.2、如何使用Service	27
2.3.3、Service的生命周期	32
2.3.4、判断服务开启状态	33
2.3.5、获取启动的服务	34
2.4、CONTENT PROVIDER	35
3、ANDROID UI LAYOUT	35
3.1、概述	35
3.2、线性布局（LINEAR LAYOUT）	36
3.3、相对布局（RELATIVE LAYOUT）	39
3.4、TABLELAYOUT	40
3.5、ABSOLUTE LAYOUT	47
4、ANDROID UI 控件	48
4.1、IMAGEBUTTON	48
4.1.1、图案填充问题	48
4.2、TEXTVIEW	49
4.2.1、动态滚动	49
4.3、EDITTEXT	49
4.3.1、光标选择	49

4.4、TITLEBAR	50
4.4.1、非全屏状态下不显示title标题栏.....	50
4.4.2、标题栏进度指示器.....	50
4.4.3、titleBar 高级实现方法(更美观).....	51
4.4.4、获取标题栏和状态栏高度.....	57
4.4.5、标题栏显示简单的进度框.....	57
4.5、MENU	58
4.5.1、简单的代码.....	58
4.5.2、menu实现的两种方法.....	58
4.5.3、自定义MENU背景	62
4.5.4、触发menu	64
4.5.5、Context Menu和Options Menu菜单的区别.....	64
4.5.6、Context menus for expandable lists.....	64
4.6、LISTVIEW	66
4.6.1、ListView自定义分割线.....	66
4.6.2、LIST例一.....	66
4.6.3、LIST例二.....	76
4.6.4、LIST例三.....	80
4.6.5、ListView 被选中item的背景颜色	82
4.6.6、ListView自定义背景颜色.....	83
4.6.7、List长按与短按消息映射.....	84
4.6.8、点击ListView改变背景色.....	87
4.6.9、自动滚动ListView	88
4.6.10、BaseExpandableListAdapter例.....	88
4.6.11、列表视图 (List View)	96
4.6.12、NoteList	99
4.7、TAB与TABHOST	106
4.8、RATINGBAR	110
4.8.1、例一.....	110
4.8.2、例二.....	112
4.9、DATE/TIME SET	115
4.9.1、DatePicker/TimePicker	115
4.9.2、DatePickerDialog/TimePickerDialog	119
4.10、WEBVIEW	120
4.10.1、WebView的使用.....	120
4.11、SCROLLVIEW	121
4.11.1、ScrollView的使用.....	121
4.12、GRIDVIEW	124
4.12.1、GridView的使用	124
4.13、GAMEVIEW	127
4.13.1、显示到一个布局中.....	127
4.14、TOASTE	128
4.14.1、短时间显示.....	128
4.14.2、长时间显示.....	128

4.15、对话框.....	128
4.15.1、简单的对话框.....	128
4.15.2、包含两个按钮的对话框.....	128
4.15.3、三个按钮的提示框.....	129
4.15.4、包含输入的dlg.....	131
4.15.5、圆形进度框.....	133
4.15.6、AlertDialog.Builder.....	133
4.15.7、模式对话框.....	134
4.16、拖动BUTTON获得位置.....	135
5、ANDROID UI 美化	137
5.1、简单美化BUTTON、IMAGEBUTTON、TEXTVIEW等控件	137
5.2、BUTTON美化案例☆.....	139
5.3、IMAGEBUTTON 按下时的动画效果	142
5.4、滚动条显示与隐藏.....	143
5.5、LISTVIEW 与 SCROLLVIEW 解决办法	144
方法一：（重写ListView）	144
方法二：	150
5.6、3D魔方	151
6、ANDROID UI 动画	160
6.1、四种 2D动画	160
6.1.1、透明度控制动画效果 alpha.....	160
6.1.2、旋转动画效果 rotate	161
6.1.3、尺寸伸缩动画效果 scale.....	162
6.1.4、位置转移动画效果 translate.....	163
6.1.5、四种动画效果的调用.....	164
7、异步调用.....	167
开辟一个线程：	167
THREAD:	168
HANDER.....	170
TIMER	173
ANDROID 界面刷新	174
MESSAGE HANDER.....	175
用法:	175
线程与子线程调用MessageHandler.....	177
Messagehandler实例.....	177
8、数据存储与读取.....	179
1. PREFERENCES.....	179
2. FILES	180
3. DATABASES	180
4. NETWORK	183
5. CONTENTPROVIDER	183

6、执行SQL语句进行查询	188
用法1.....	188
其它:	188
详解:	189
查看SQLITE表格内容	192
9、常用功能的实现.....	193
9.1、获取手机型号以及系统版本号	193
9.2、更改应用程序图标	194
9.3、迎合不同的手机分辨率	194
9.4、ANDROID屏幕适应的四个原则	195
9.5、ANDROID常用单位	196
9.6、取得屏幕信息	197
9.7、横竖屏	197
9.8、程序完全全屏	200
9.8.1 锁屏锁键盘.....	200
9.9、程序的开机启动	201
9.10、动态START页面	208
9.11、彻底退出当前程序	212
9.12、获取应用程序的名称，包名，版本号和图标	212
9.13、调用ANDROID INSTALLER 安装和卸载程序	215
9.14、后台监控应用程序包的安装&卸载	216
9.15、显示应用详细列表	224
9.16、寻找应用	224
9.17、注册一个BROADCASTRECEIVER	225
9.18、打开另一程序	225
9.19、播放默认铃声	225
9.20、设置默认来电铃声	226
9.21、位图旋转	227
9.22、手机震动控制	228
9.23、SENSOR2D感应实例	228
9.24、运用JAVA MAIL包实现发GMAIL邮件	230
9.26、ANDROID键盘响应.....	236
9.27、后台监听某个按键	238
9.28、VECTOR用法.....	239
9.29、CURSOR	242
9.30、把一个字符串写进文件	244
9.31、把文件内容读出到一个字符串	245
9.32、扫描WIFI热点演示实例教程	246
9.33、调用GOOGLE搜索	249
9.34、调用浏览器 载入某网址	249
9.35、获取 IP地址	249
9.36、从输入流中获取数据并以字节数组返回	250
9.37、通过ANDROID 客户端上传数据到服务器	251
9.38、文件下载类	255

9.39、下载文件的进度条提示	263
9.40、通过HTTPCLIENT从指定SERVER获取数据	265
9.41、通过FTP传输文件，关闭UI获得返回码	266
9.42、激活JAVASCRIPT打开内部链接	266
9.43、清空手机COOKIES	267
9.44、检查SD卡是否存在并且可以写入	267
9.45、获取SD卡的路径和存储空间	268
9.46、将程序安装到SD卡	268
9.47、创建一个SD映像	269
9.48、查看手机内存存储	269
9.49、在模拟器上调试GOOGLE MAPS	271
9.50、建立GPRS连接	273
9.51、获取手机位置	274
9.5* 获得经纬度，地名标注在地图上	274
9.52、获得两个GPS坐标之间的距离	276
9.53、通过经纬度显示地图	277
9.54、路径规划	277
9.55、将坐标传递到GOOGLE MAP并显示	277
9.56、获取本机电话号码	280
9.57、获得手机联系人	280
9.58、2.0 以上版本查询联系人详细信息	282
9.59、2.0 以上版本添加联系人	285
9.60、拨打电话	287
9.61、发送SMS、MMS	287
9.62、监听电话被呼叫状态	288
9.63、监听要拨打的电话(可以后台进行修改号码)	290
9.64、后台监听短信内容	291
9.65、删除最近收到的一条短信	292
9.66、调用发短信的程序	293
9.67、后台发送短信	293
9.68、调用发送彩信程序	294
9.69、发送EMAIL	294
9.70、播放多媒体	295
9.71、控制音量	296
9.72、定义CONTENTOBSERVER，监听某个数据表	302
9.73、打开照相机	303
9.74、从GALLERY选取图片	303
9.75、打开录音机	303
9.76、语音朗读	303
9.77、手机获取视频流显示在电脑上	305
9.78、蓝牙的使用	313
9.79、一个很好的加密解密字符串	316
9.80、DRAWABLE、BITMAP、BYTE[]之间的转换	318
9.81、高循环效率的代码	320

9.82、给模拟器打电话发短信	321
9.83、加快模拟器速度	321
9.83.1、模拟器“尚未注册网络”	322
9.84、EMULATOR命令行参数	322
9.85、如何进行单元测试	323
9.86、ANDROID自动化测试初探	324
9.86.1、捕获Activity上的Element	324
9.86.2、Hierarchyviewer 捕获Element的	328
9.86.3、架构实现	330
9.86.4、模拟键盘鼠标事件 (Socket+Instrumentation实现)	332
9.86.5、再述模拟键盘鼠标事件 (adb shell 实现)	334
9.87、反编译APK	344
9.88、更换APK图标(签名打包)	348
9.89、利用ANDROID MARKET赚钱	363
9.90、ANDROID-MARKET 使用	365
9.91、传感器	369
9.91.1、获取手机上的传感器	369
9.91.2、	371
9.92、时间类	372
* 获得日期或时间字符串	372
* num天前的日期	373
* num天后的日期	373
* 判断 thingdate 的 dotime 天后是否在今天之后	374
* 判断testDate+testTime 是否在两个时间之内	375
附录:	378
附录 1、XML布局中的常用属性	378
1. 通用属性	378
2. Edit Text部分属性	381
3. layout_alignParentRight android:paddingRight	384
附录 2、INTENT ACTION	385
附录 3、ANDROID的动作、广播、类别等标志	387
★★★附带工具包说明	393
1. APK反编译工具.rar	393
2. APK安装工具.rar	393

0、Android常用类库说明

在 Android 中，各种包写成 android.*的方式，重要包的描述如下所示：

android.app：提供高层的程序模型、提供基本的运行环境

android.content 包含各种的对设备上的数据进行访问和发布的类

android.database：通过内容提供者浏览和操作数据库



android.graphics : 底层的图形库, 包含画布, 颜色过滤, 点, 矩形, 可以将他们直接绘制到屏幕上.

android.location : 定位和相关服务的类

android.media : 提供一些类管理多种音频、视频的媒体接口

android.net : 提供帮助网络访问的类, 超过通常的 java.net.* 接口

android.os : 提供了系统服务、消息传输、IPC 机制

android.opengl : 提供 OpenGL 的工具, 3D 加速

android.provider : 提供类访问 Android 的内容提供者

android.telephony : 提供与拨打电话相关的 API 交互

android.view : 提供基础的用户界面接口框架

android.util : 涉及工具性的方法, 例如时间日期的操作

android.webkit: 默认浏览器操作接口

android.widget: 包含各种 UI 元素 (大部分是可见的) 在应用程序的屏幕中使用

1、Android 文件系统与应用程序架构

1.1、Android 文件系统

```
# pwd && ls -a -l
```

```
/
```

```
drwxrwxrwt root root 2009-06-10 09:53 sqlite_stmt_journals
```

```
drwxrwx--- system cache 2008-09-06 22:51 cache
```

```
d---rwxrwx system system 1970-01-01 08:00 sdcard
```

```
lrwxrwxrwx root root 2009-06-09 22:11 etc -> /system/etc
```

```
drwxr-xr-x root root 2008-09-06 22:45 system
```

```
drwxr-xr-x root root 1970-01-01 08:00 sys
```

```
drwxr-x--- root root 1970-01-01 08:00/sbin
```

```
-rw-r--r-- root root 117 1970-01-01 08:00 runme.sh
```

```
dr-xr-xr-x root root 1970-01-01 08:00 proc
```

```
-rwxr-x--- root root 1704 1970-01-01 08:00 init.trout.rc
```

```
-rwxr-x--- root root 9329 1970-01-01 08:00 init.rc
```

```
-rwxr-x--- root root 1677 1970-01-01 08:00 init.goldfish.rc
```

```
-rwxr-x--- root root 106636 1970-01-01 08:00 init
```

```
-rw-r--r-- root root 118 1970-01-01 08:00 default.prop
```

```
drwxrwx--x system system 2008-09-06 22:51 data
```

```
drwx----- root root 2009-06-07 16:29 root
```

```
drwxr-xr-x root root 2009-06-09 22:11 dev
```

sqlite_stmt_journals: 一个根目录下的 tmpfs 文件系统, 用于存放临时文件数据。

cache : 是缓存临时文件夹, 据说是除了 T-mobile 的 OTA 更新外, 别无用处。

sdcard: 是 SD 卡中的 FAT32 文件系统挂载的目录

etc : 指向 /system/etc , 众所周知的配置文件存放目录

system : 是一个很重要的目录, 系统中的大部分东西都在这里了, 以下是目录结构:

ls -a -l /system

```

drwxr-xr-x root 208 1970-01-01 08:00 xbin
drwxr-xr-x root root 1970-01-01 08:00 modules
drwxr-xr-x root root 2008-08-01 20:00 framework
drwxr-xr-x root root 2008-08-01 20:00 fonts
drwxr-xr-x root root 2008-08-01 20:00 etc
-rw-r--r-- root root 2197 2008-08-01 20:00 build.prop
drwxr-xr-x root root 2008-08-01 20:00 media
drwxr-xr-x root shell 2008-08-01 20:00 bin
drwxr-xr-x root root 2008-08-01 20:00 usr
drwxr-xr-x root root 2008-08-01 20:00 app
drwxr-xr-x root root 2008-09-06 22:45 lost+found
drwxr-xr-x root root 2008-08-01 20:00 lib
drwxr-xr-x root root 2008-08-01 20:00 sd
-rw-r--r-- root root 145 2008-08-01 20:00 init.rc

```

sys : 用於挂载 sysfs 文件系统。在设备模型中,sysfs 文件系统用来表示设备的结构.将设备的层次结构形象的反应到用户空间中.用户空间可以修改 sysfs 中的文件属性来修改设备的属性值

sbin: 只放了一个用於调试的 adbd 程序。

proc : /proc 文件系统下的多种文件提供的系统信息不是针对某个特定进程的,而是能够在整个系统范围的上下文中使用。

data : 存放用户安装的软件以及各种数据。

root : 什么都没有。

dev : 不用多说了,设备节点文件的存放地。

下面介绍非目录的文件:

runme.sh 用於 SD 卡中 EXT2 文件系统的自动挂载动作的脚本。

init.trout.rc, init.rc, init.goldfish.rc 是初始化文件。

init 是系统启动到文件系统的时候第一个运行的程序。

从以上的根目录分析来看,Android 的根文件系统并非标准的 Linux 文件系统,所以以後还得仔细分析一下启动过程,才能认识 Android 系统。

今天要来分析一下 Android 文件系统的/system 目录的结构。

/system 目录是在 Android 文件系统占有及其重要的位置,基本上所有的工具和应用程序都在这个目录下,我看来是一个真正的 rootfs。他在 Android 手机中存放在 nandflash 的 mtd3 中,是一个 yaffs2 文件系统,在启动时被挂载在 root 的/system 目录下,其中包含有:

pwd && ls -a -l

```

/system
drwxr-xr-x root 208 1970-01-01 08:00 xbin

```




```

drwxr-xr-x root root 1970-01-01 08:00 modules
drwxr-xr-x root root 2008-08-01 20:00 framework
drwxr-xr-x root root 2008-08-01 20:00 fonts
drwxr-xr-x root root 2008-08-01 20:00 etc
-rw-r--r-- root root 2197 2008-08-01 20:00 build.prop
drwxr-xr-x root root 2008-08-01 20:00 media
drwxr-xr-x root shell 2008-08-01 20:00 bin
drwxr-xr-x root root 2008-08-01 20:00 usr
drwxr-xr-x root root 2008-08-01 20:00 app
drwxr-xr-x root root 2008-09-06 22:45 lost+found
drwxr-xr-x root root 2008-08-01 20:00 lib
drwxr-xr-x root root 2008-08-01 20:00 sd
-rw-r--r-- root root 145 2008-08-01 20:00 init.rc

```

下面逐个分析其中的目录：

xbin：下放了很多系统管理工具，这些工具不是到 **toolbox** 的链接，每个都是可执行程序。如果你看到这些命令你会发现他们根本不常用，他们都是为系统管理员准备的，是一些系统管理和配置工具。这个文件夹的作用相当于标准 Linux 文件系统中的 **/sbin**。我的手机此目录下有 **busybox**，肯定是改过的，应该是破解者加上的。

modules：使用来存放内核模块（主要是 **fs** 和 **net**）和模块配置文件的地方。

framework：是 JAVA 平台的一些核心文件，属于 JAVA 平台系统框架文件。里面的文件都是 **.jar** 和 **.odex** 文件。

备注：什么是 **odex** 文件？**odex** 是被优化过的 JAVA 程序文件，体积通常是 **.jar** 的 4 倍左右。执行效率比 **.jar** 高。

fonts：很显然，这是字体库文件的存放目录。

etc：这里存放了系统中几乎所有的配置文件，根目录下的 **/etc** 就链接于此。

build.prop：是一个属性文件，在 Android 系统中 **.prop** 文件很重要，记录了系统的设置和改变，类似于 **/etc** 中的文件。

media：里面主要是存放了系统的铃声的，分为 **notifications**（通知）、**ui**（界面）、**alarms**（警告）和 **ringtones**（铃声），里面都是 **.ogg** 音频文件。

bin：众所周知，是存放用户常用的工具程序的，其中大部分是到 **toolbox** 的链接（类似嵌入式 Linux 中的 **busybox**）。**toolbox** 应该是 google 简化版的 **busybox**，我还没深入研究过。

usr：用户的配置文件，如键盘布局、共享、时区文件等等。您可以 **cat** 来看看。

app：顾名思义，存放的是 Android 系统自带的 JAVA 应用程序。

lost+found：**yaffs** 文件系统固有的，类似回收站的文件夹，只有是 **yaffs** 文件系统都会有。

lib：存放几乎所有的共享库（**.so**）文件。

sd：SD 卡中的 **EXT2** 分区的挂载目录

init.rc：一个初始化脚本，用于将 **/system/modules** 和 **/system/xbin** 挂载为 **cramfs**，避免系统被无意破坏。

好了，**/system** 目录的结构就简单分析到这里。

1.2、Android应用程序架构

src/ java 源代码存放目录



gen/ 自动生成目录

gen 目录中存放所有由 Android 开发工具自动生成的文件。目录中最重要的就是 R.java 文件。这个文件由 Android 开发工具自动产生的。Android 开发工具会自动根据你放入 res 目录的 xml 界面文件、图标与常量，同步更新修改 R.java 文件。正因为 R.java 文件是由开发工具自动生成的，所以我们应避免手工修改 R.java。R.java 在应用中起到了字典的作用，它包含了界面、图标、常量等各种资源的 id，通过 R.java，应用可以很方便地找到对应资源。另外编译器也会检查 R.java 列表中的资源是否被使用到，没有被使用到的资源不会编译进软件中，这样可以减少应用在手机占用的空间。

res/ 资源(Resource)目录

在这个目录中我们可以存放应用使用到的各种资源，如 xml 界面文件，图片或数据。具体请看 ppt 下方备注栏。

AndroidManifest.xml 功能清单文件

这个文件列出了应用程序所提供的功能，在这个文件中，你可以指定应用程序使用到的服务(如电话服务、互联网服务、短信服务、GPS 服务等)。另外当你新添加一个 Activity 的时候，也需要在这个文件中进行相应配置，只有配置好后，才能调用此 Activity。

default.properties 项目环境信息，一般是不需要修改此文件

res/drawable 专门存放 png、jpg 等图标文件。在代码中使用 getResources().getDrawable(resourceId)获取该目录下的资源。

res/layout 专门存放 xml 界面文件，xml 界面文件和 HTML 文件一样，主要用于显示用户操作界面。

res/values 专门存放应用使用到的各种类型数据。不同类型的数据存放在不同的文件中，如下：

- strings.xml 定义字符串和数值，在 Activity 中使用 getResources().getString(resourceId) 或 getResources().getText(resourceId)取得资源。它的作用和 struts 中的国际化资源文件一样。

```
<?xml version="1.0" encoding="UTF-8"?>
<resources>
    <string name="android"> android</string>
</resources>
```

- arrays.xml 定义数组。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="colors">
        <item>red</item>
        <item>yellow</item>
        <item>green</item>
        <item>blue</item>
    </string-array>
</resources>
```

- colors.xml 定义颜色和颜色字符串数值，你可以在 Activity 中使用 getResources().getDrawable(resourceId) 以及 getResources().getColor(resourceId)取得这些资源。例子如下：

```

    <?xml version="1.0" encoding="UTF-8"?>
<resources>
    <color name="contents_text">#ff000000</color>
</resources>

```

· `dimens.xml` 定义尺寸数据, 在 Activity 中使用 `getResources().getDimension(resourceId)` 取得这些资源

```

    <?xml version="1.0" encoding="UTF-8"?>
<resources>
    <dimen name="key_height">50dip</dimen>
</resources>

```

· `styles.xml` 定义样式。

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="androidText" parent="@style/Text">
        <item name="android:textSize">18px</item>
        <item name="android:textColor">#008</item>
    </style>
</resources>

```

`res/anim/` 编译成帧动画的 XML 文件。

`res/xml/` 在 Activity 中使用 `getResources().getXML()` 读取该目录下的 XML 资源文件。

`res/raw/` 该目录下的文件将直接被复制到设备上。编译软件时, 这些数据不会被编译, 它们被直接加入到程序安装包里。为了在程序中使用这些资源, 你可以调用 `getResources().openRawResource(ID)`, 参数 ID 形式: `R.raw.somefilename`。

2、Android应用程序结构

Android 应用程序一般由四部分组成:

1. Activity 2. Intent Receiver 3. Service 4. Content Provider

并非所有的应用程序都要包括所有四个部分, 但是您的应用程序是由其中的组合构成的。

一旦您决定使用哪个组件, 您需要把它们列出在名为 `AndroidManifest.xml` 的文件中。这是一个 XML 文件, 其中声明了您的应用程序中所需要的组件以及它们的功能和需求。



2.1、Activity

Activity 是四个 Android 构造块中最基本的组件。一个 activity 通常是一个单独的屏幕。每一个 activity 作为一个独立的类来实现，均继承自 Activity 基类。您的 activity 类将显示一个由若干 Views 控件组成的用户界面并对事件做出响应。大多数应用程序包含多个屏幕。举例来说，一个文本消息应用程序也许会有一个屏幕，用来显示联系人列表，第二个屏幕用来编辑短消息，还有用来浏览历史消息或者用来更改设置的屏幕。每一个这样的屏幕都将作为一个 activity。切换屏幕是通过打开一个新的 activity 来实现的。在一些实例中，一个 activity 会将返回值返回给前一个 activity，比如，一个允许用户选择图片的 activity 将返回选中的图片到调用方。

当一个新的屏幕打开，前一个屏幕将暂停并保存在历史堆栈中。用户在历史堆栈中可以回退到前一个屏幕。当屏幕不再使用时，还可以在历史堆栈中删除。Android 将保留历史堆栈为从主屏幕开始的每一个应用。

2.1.1、概述

一个 Android 应用程序很少会只有一个 Activity 对象，如何在多个 Activity 之间进行跳转，而且能够互相传值是一个很基本的要求。

在前面创建的 MyApp 中，我们通过点击按钮可以更新当前界面上的文本内容。现在我们想换种方式，在点击按钮后，显示一个新的屏幕，在这个屏幕上输入一段话，然后再返回到原先的界面显示刚才输入的那句话。

首先我们新建这个新屏幕的布局文件 input.xml，并添加一个文本输入框和一个按钮（注意，xml 元素的命名不要和其他布局文件中的定义重名，因为所有的资源都在 R 中进行索引，比如 id，如果重名了在使用 R.id.* 的时候就会有问题了）。这个布局文件对应的是一个 Activity，因此我们再新建一个 Input 类（继承自 Activity）用于显示这个布局并响应事件。

```
9 public class Input extends Activity
10 {
11     @Override
12     protected void onCreate(Bundle savedInstanceState)
13     {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.input);
16     }
17 }
```

然后，还有一个重要的工作，那就是在清单文件 AndroidManifest.xml 中告诉程序，我定义了一个新的 Activity，你可以去调用它。



```

1<?xml version="1.0" encoding="utf-8"?>
2<manifest xmlns:android="http://schemas.android.com/apk/res/android"
3    package="wuyf.android.myapplication">
4    <application android:icon="@drawable/icon" android:label="@string/app_name">
5        <activity android:name=".MyApp" android:label="@string/app_name">
6            <intent-filter>
7                <action android:name="android.intent.action.MAIN" />
8                <category android:name="android.intent.category.LAUNCHER" />
9            </intent-filter>
10        </activity>
11        <activity android:name=".Input" android:label="@string/app_name">
12        </activity>
13    </application>
14</manifest>

```

我们希望在以前的那个主界面上点击按钮以后可以跳转到文本输入界面，所以我们需要对按钮的 onClick 事件进行定义：

```

13@
14    @Override
15    public void onCreate(Bundle savedInstanceState)
16    {
17        super.onCreate(savedInstanceState);
18        setContentView(R.layout.main);
19
20        buttonNewInput = (Button) this.findViewById(R.id.buttonNewInput);
21        buttonNewInput.setOnClickListener(new View.OnClickListener()
22        {
23            public void onClick(View view)
24            {
25                Intent intent = new Intent(MyApp.this, Input.class);
26                startActivityForResult(intent, 0);
27            }
28        });

```

在这段代码里出现了一些新东西。首先是 Intent，它是 Android 一个很重要的类。Intent 直译是“意图”，什么是意图呢？比如你想从这个 Activity 跳转到另外一个 Activity，这就是一个意图。它不但可以连接多个 Activity，还可以在它们之间传递数据。在这里，我们就是用 Intent 从 MyApp 对象跳转到了 Input 对象。

再看紧跟着的 startActivityForResult() 方法，顾名思义，它可以从一个定义好的 Intent 对象启动一个新的 Activity，并且，这个 Activity 会返回执行的结果，这些内容下面马上就会提到。

好，这里我们已经可以调出新 Activity 了，我们看一下执行的结果：

你马上可以想到，现在需要对新的 Activity (Input) 进行了处理了。我们在点击“确定”按钮的时候，需要获得上面 EditText 对象中的文本，然后返回到前一个 Activity (MyApp) 中去。看我们的按钮事件处理：

```

22    editTextInput = (EditText) findViewById(R.id.editTextInput);
23    buttonInputOk = (Button) this.findViewById(R.id.buttonInputOk);
24    buttonInputOk.setOnClickListener(new View.OnClickListener()
25    {
26        public void onClick(View view)
27        {
28            String text = editTextInput.getText().toString();
29
30            SharedPreferences preferences = getSharedPreferences("Text", 0);
31            SharedPreferences.Editor editor = preferences.edit();
32            editor.putString("text", text);
33
34            if (editor.commit())
35            {
36                setResult(Activity.RESULT_OK);
37            }
38
39            finish();
40        }
41    });

```



这里的关键是 `SharedPreferences` 对象，这是在多个 `Activity`（同一包中）共享数据的方式，本质上它就是一个可以在包的范围内进行数据共享的文件。

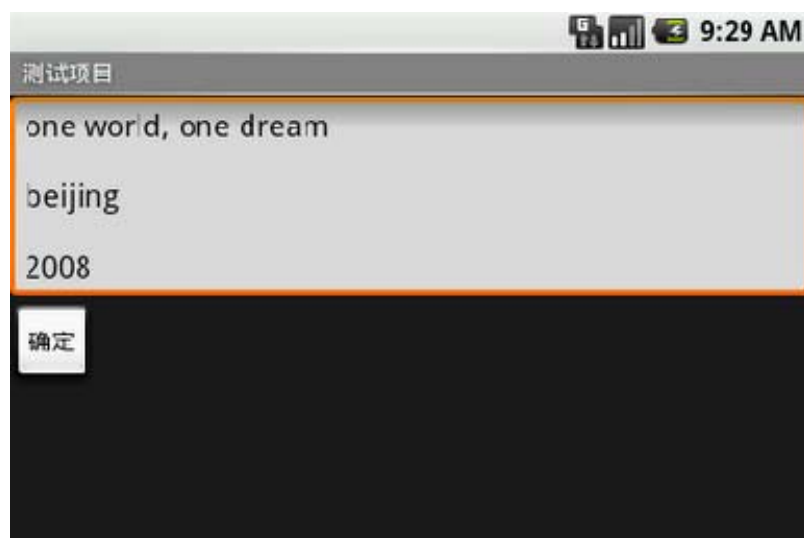
我们通过一个标签“Text”获得了和文本相关的那个 `SharedPreferences` 对象（“Text”仅仅是自己定义的一个标签），然后给它赋予一个“text”对象值为当前文本框中输入的文本。设置完成以后，设置当前 `Activity` 的执行结果为 `RESULT_OK`，再关闭当前的 `Activity`，剩下的事情就可以回到 `MyApp` 这个主界面中去执行了。

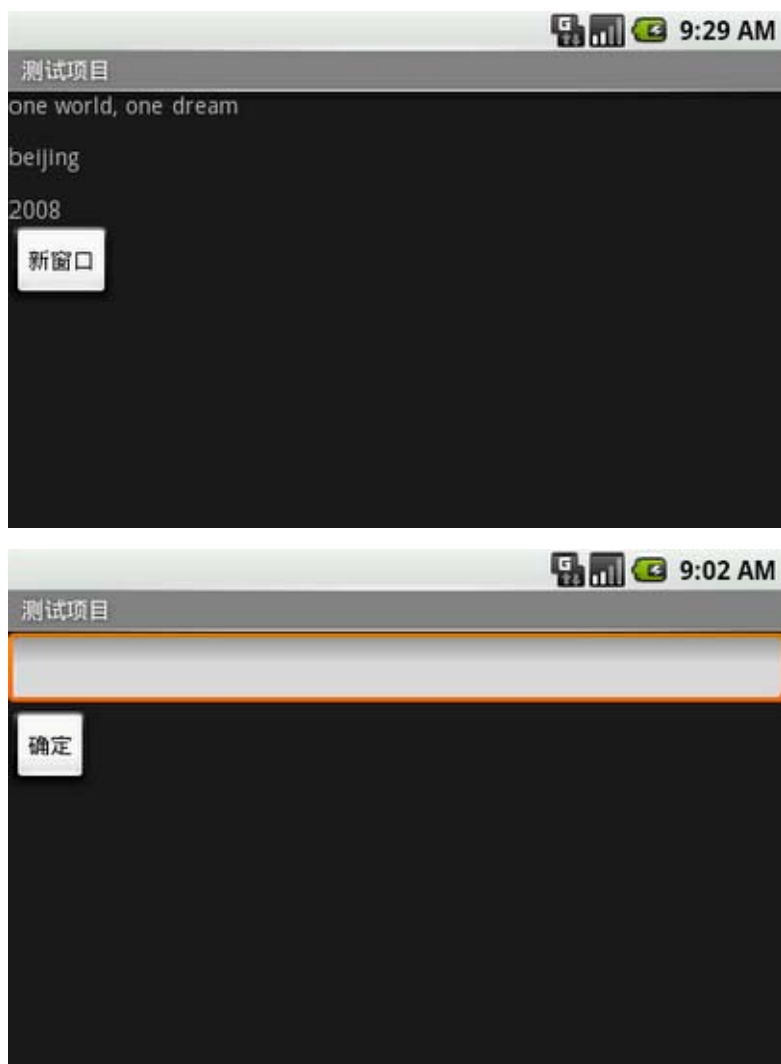
其实剩下的事情也很简单，在 `MyApp` 这个 `Activity` 中，我们需要重写一个函数，`onActivityResult()`。因为我们启动 `Input` 这个 `Activity` 的时候使用的是 `startActivityForResult()` 方法，这个方法会使 `Input` 执行完以后返回给 `MyApp` 一个结果，`MyApp` 接收到返回结果的时候会触发 `onActivityResult` 事件，对于结果的处理就在 `onActivityResult()` 中进行。同样，我们通过“Text”这个标签获得 `SharedPreferences` 对象，再把字符串从“text”对象中取出来并显示到当前屏幕上。

另外说明一下，`requestCode` 是用来标识请求对象，我们刚才在启动 `Activity` 的时候使用的是“`startActivityForResult(intent, 0)`”，这里的 0 就是 `requestCode`，当然，你可以设置成任何你喜欢的值。

```
336 @Override
337 protected void onActivityResult(int requestCode, int resultCode, Intent data)
338 {
339     if (requestCode == 0)
340     {
341         if (resultCode == Activity.RESULT_OK)
342         {
343             SharedPreferences preferences = getSharedPreferences(
344                 "Text", 0);
345             textViewShowInput.setText(preferences.getString("text", null));
346         }
347     }
348 }
349 }
```

我们看一下执行结果：





2.1.2、Activity的生命周期

和其他手机平台的应用程序一样，Android 的应用程序的生命周期是被统一掌控的，也就是说我们写的应用程序命运掌握在别人（系统）的手里，我们不能改变它，只能学习并适应它。

简单地说一下为什么是这样：我们手机在运行一个应用程序的时候，有可能打进来电话 发进来短信，或者没有电了，这时候程序都会被中断，优先去服务电话的基本功能，另外系统也不允许你占用太多资源，至少要保证电话功能吧，所以资源不足的时候也就有可能被干掉。

OPhone Activity 生命周期解析

要为你的 OPhone 应用程序创建用户界面屏幕，就需要继承 Activity 类，并且使用 Views 为你的应用程序提供用户交互。每个 Activity 表示用户界面中的一个屏幕。你的应用程序越复杂，需要的屏就越多，每一屏都是一个新的 Activity。典型的一个应用程序一般至少包括一个屏用来处理用户界面的主要功能，也常常还有其他的屏用来输入用户信息，或者展现不同

的数据并支持更多的功能。大多数 Activity 都是全屏的，但是你也可以创建半透明或者浮动的 Activity。

理解 Activity 的生命周期对应用程序开发来说是至关重要的，这样才能确保您的应用提供了一个很好的用户体验和妥善管理其资源。由于 OPhone 应用程序不控制自己的进程寿命，由 OPhone Runtime 管理每个应用程序进程，但是每个 Activity 的状态反过来会影响到 OPhone Runtime 是否将终止当前 Activity 和还是让它继续运行

2.1.3、Activity 的创建

通过继承 Activity 类，我们可以创建一个新的 Activity。基本的示例代码如下所示：

```
import android.app.Activity;
import android.os.Bundle;

public class MyActivity extends Activity {

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle icle) {
        super.onCreate(icle);
    }
}
```

Activity 基类仅仅是一个封装了一些与窗口显示相关功能的空空的屏幕，本身并不具备太具体用处，因此创建 Activity 以后，第一件事情就是在这个空空的屏幕上摆上你所需要的物件，各种 Views。你可以在 xml 文件中的定义你所需要的 Views，也可以在代码中定义。通过在 Activity onCreate 方法中调用 setContentView，便可将 Activity 与你的 Views 绑定在一起来实现用户交互的功能。

在代码中定义你的 Views

```
@Override
public void onCreate(Bundle icle) {
    super.onCreate(icle);
    MyView myView = new MyView(this);
    setContentView(myView);
}
```

在 layout 文件中定义你的 Views

```
@Override
public void onCreate(Bundle icle) {
    super.onCreate(icle);
    setContentView(R.layout.main);
}
```



为了在你的应用程序中使用 Activity，你还需要将你写的 Activity 类注册到 Manifest 当中，参见以下 XML 片段：

```

1. <activity android:label="@string/app_name"
2. android:name=".MyActivity">
3. <intent-filter>
4. <action android:name="android.intent.action.MAIN" />
5. <category android:name="android.intent.category.LAUNCHER" />
6. </intent-filter>
7. </activity>

```

2.1.4、Activity 的跳转（含Bundle传值）

1、2 个 Activity 的切换,没有数据传递

```

1. //从 A 到 B
2. Intent intent = new Intent();
3. intent.setClass(A.this, B.class);
4. startActivity(intent);
5. A.this.finish();

```

2、2 个 Activity 之间传递数据(简单)

//A数据传给B

//A中代码：“passData” 是自定义的识别标志，可以随便命名~ 还可以添加多个

```

Intent intent = new Intent();
intent.setClass(A.this, B.class);
Bundle mBundle = new Bundle();
mBundle.putString("passData", "ray'blog");//压入数据
intent.putExtras(mBundle);
startActivity(intent);
A.this.finish();

```

//B 中接受数据的代码：

```

//读出数据， 则 data 的值为 ray 'blog
Bundle bundle = getIntent().getExtras();
String data = bundle.getString("passData");

```



3、2 个 Activity 之间传递数据

相关的几个函数

startActivityForResult

```
public final void setResult(int resultCode, String data)
```

回调函数

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

例如 A 到 B, 从 B 得到 A 的数据

//A 到 B

```
static final int RG_REQUEST = 0;
Intent intent = new Intent();
intent.setClass(A.this, B.class);
startActivityForResult(intent, RG_REQUEST);
```

//在 B 中处理

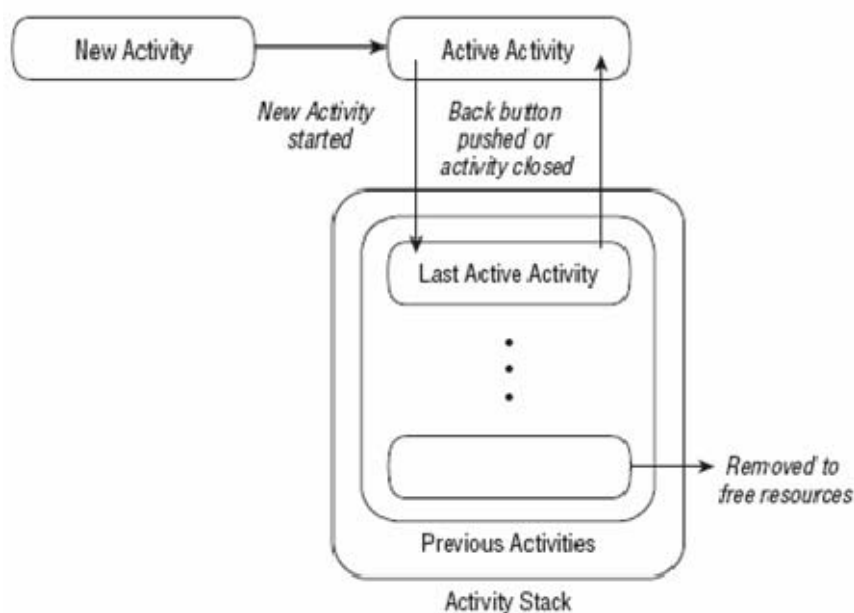
```
Bundle bundle = new Bundle();
bundle.putString("DataKey", editText.getText().toString()); //给 bundle 写入数据
Intent mIntent = new Intent();
mIntent.putExtras(bundle);
setResult(RESULT_OK, mIntent);
finish();
```

//最后在 A 的回调函数里面接收数据

```
if (requestCode == RG_REQUEST) {
    if (resultCode == RESULT_CANCELED)
        setTitle("Canceled...");
    else if (resultCode == RESULT_OK) {
        setTitle((String) data.getCharSequenceExtra("DataKey"));
    }
}
```

2.1.5. Activity 堆栈

每个 Activity 的状态由它所在 Activity 栈中的位置所决定, 所有当前正在运行的 Activity 将遵循照后进先出的原则。当一个新的 Activity 启动, 当前的 Activity 将移至堆栈的顶部, 如果用户使用 Back 按钮, 或在前台 Activity 被关闭, 下一个 Activity 将被激活并且移至到堆栈的顶部。这个过程如下图所示



2.1.6、Intent对象调用Activity实例

Android 中一个 Activity 调用另一个 Activity — Intent 对象的使用

第一步：建立 **Android** 工程： **IntentDemo**。

第二步：编写 **Activity** 的子类别： **IntentDemo**，其程序代码如下：

```
package com.a3gs.intent;
```

```
import android.app.Activity;
```

```
import android.content.Intent;
```

```
import android.os.Bundle;
```

```
import android.view.View;
```

```
import android.widget.Button;
```

```
public class IntentDemo extends Activity {
```

```
    /** Called when the activity is first created. */
```

```
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.main);
```

```
        Button btn1 = (Button) findViewById(R.id.btn1);
```

```
        btn1.setOnClickListener(new Button.OnClickListener(){
```

```
            @Override
```

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



```

        // TODO Auto-generated method stub
        Intent intent = new Intent();
        intent.setClass(IntentDemo.this, Intent2.class);
        /* 调用一个新的 Activity */
        startActivity(intent);
        /* 关闭原本的 Activity */
        IntentDemo.this.finish();
    }
});
}
}

```

第三步：新建一个 Activity 的子类别：Intent2，其程序代码如下：

```

package com.a3gs.intent;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class Intent2 extends Activity {
    /** Called when the activity is first created. */

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.mylayout);
        Button btn2 = (Button) findViewById(R.id.btn2);
        btn2.setOnClickListener(new Button.OnClickListener() {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                Intent intent = new Intent();
                intent.setClass(Intent2.this, IntentDemo.class);
                startActivity(intent);
                Intent2.this.finish();
            }
        });
    }
}

```

第四步：修改 res/layout/main.xml，其代码如下：

```

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

```



```

xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<Button android:id="@+id/btn1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="activity2"
/>
</LinearLayout>

```

第五步：修改 res/layout/ mylayout.xml，其代码如下：

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<Button android:id="@+id/btn2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="activity1"
/>
</LinearLayout>

```

第六步：AndroidManifest.xml 中声明 activity

```
<activity android:name=" Intent2"></activity>
```

OK了！

2.1.7、Activity透明

OnCreate 中不设 Layout

```
this.setTheme(R.style.Theme_Transparent);
```

以下是 Theme_Transparent 的定义（注意 transparent_bg 是一副透明的图片）

8. 屏幕元素设置句柄

使用 Activity.findViewById 来取得屏幕上的元素的句柄。使用该句柄您可以设置或获取任何该对象外露的值。

```
TextView msgTextView = (TextView)findViewById(R.id.msg);
```



```
msgTextView.setText(R.string.push_me);
```

2.1.8、一次性关闭所有的Activity

```
ActivityManager am = (ActivityManager) getSystemService (Context.ACTIVITY_SERVICE);  
am.restartPackage(getPackageName());
```

系统会将，该包下的，所有进程，服务，全部杀掉，就可以杀干净了，要注意加上

```
<uses-permission
```

```
android:name="android.permission.RESTART_PACKAGES"></uses-permission>
```

2.1.9、PreferenceActivity 用法

```
public class Setting extends PreferenceActivity  
{  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        addPreferencesFromResource(R.xml.settings);  
    }  
}
```

Setting.xml:

```
Android:key="seting2"  
android:title="@string/seting2"  
android:summary="@string/seting2"/>  
android:key="seting1"  
android:title="@string/seting1"  
android:summaryOff="@string/seting1summaryOff"  
android:summaryOn="@stringseting1summaryOff"/>
```



2.1.10、Dialog风格的Activity



可以在弹出的 Activity 中添加进度条等控件~~~使用广泛。。

方法:

跟 activity 用法一样, 只是在 manifest 文件中注册 Activity 时多添加一个 dialog 风格的语句。

如:

```
<activity
    android:name=".SelectDlg"
    android:label="@string/app_name"
    android:theme="@android:style/Theme.Dialog" />
```

2.1.11、横竖屏切换不销毁当前Activity

首先在 Manifest.xml 的 Activity 元素中加入

`android:configChanges="orientation|keyboardHidden"` 属性

```
<activity android:name=".FileBrowser"

    android:label="@string/app_name"

    android:configChanges="orientation|keyboardHidden">

    <intent-filter>

        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
```

```

        </intent-filter>

    </activity>

```

加入这条属性的含义是，应用程序将会处理屏幕方向和键盘状态(推出或合上)信息的改动。但对于其他的设备配置信息的改动则会由 Android 系统来处理（销毁当前 Activity，然后重启一个新的 Activity 实例）。

那么，现在还需要在 java 代码的 activity 子类中加入配置信息改动的处理代码。这个也很简单

```

/**
 * onConfigurationChanged
 *
 * the package:android.content.res.Configuration.
 *
 * @param newConfig, The new device configuration.
 *
 * 当设备配置信息有改动（比如屏幕方向的变化，实体键盘的推开或合上等）时，
 *
 * 并且如果此时有 activity 正在运行，系统会调用这个函数。
 *
 * 注意: onConfigurationChanged 只会监测应用程序在 AnroidMainifest.xml 中通过
 *
 * android:configChanges="xxxx"指定的配置类型的改动；
 *
 * 而对于其他配置的更改，则系统会 onDestroy()当前 Activity，然后重启一个新的
 *
 * Activity 实例。

```

```

 */

@Override

public void onConfigurationChanged(Configuration newConfig) {

    super.onConfigurationChanged(newConfig);

    // 检测屏幕的方向：纵向或横向

    if (this.getResources().getConfiguration().orientation

        == Configuration.ORIENTATION_LANDSCAPE) {

```



```
        //当前为横屏， 在此处添加额外的处理代码

    }

    else if (this.getResources().getConfiguration().orientation

        == Configuration.ORIENTATION_PORTRAIT) {

        //当前为竖屏， 在此处添加额外的处理代码

    }

    //检测实体键盘的状态：推出或者合上

    if (newConfig.hardKeyboardHidden

        == Configuration.HARDKEYBOARDHIDDEN_NO){

        //实体键盘处于推出状态，在此处添加额外的处理代码

    }

    else if (newConfig.hardKeyboardHidden

        == Configuration.HARDKEYBOARDHIDDEN_YES){

        //实体键盘处于合上状态，在此处添加额外的处理代码

    }

}
```

别忘了在 java 文件中加上 `import android.content.res.Configuration`。

这样就 OK 了，屏幕方向改变时，应用程序的显示界面也会随着改动，而不是被销毁！

2.2、Intent Receiver

您可以使用 `IntentReceiver` 来使您的应用程序代码能够响应外部事件，如电话呼入、数据网络可用、处于晚上时。尽管 `IntentReceiver` 可以使用 `NotificationManager` 来提醒用户一些感兴趣的事件的发生，但是它并不显示用户界面。`Intent Receiver` 在 `AndroidManifest.xml` 中完成注册，当然您也可以在代码中通过 `Context.registerReceiver()` 方法完成注册。您的应用程序的触发不必调用 `intent receiver`；系统会在需要的时候启动您的应用程序，当 `intent receiver` 被触发时。应用程序也可以通过 `Context.broadcastIntent()` 将自身的 `intent` 广播给其他应用程序。

Intent 以及 Intent Filters

Android 使用一个名为 Intent 的类来完成屏幕间的切换。Intent 类描述了应用程序想要做什么。一个 intent 的数据结构包含两个最重要的部分为 action（动作）和 data（数据）。典型的 action 有 MAIN（activity 的入口）、VIEW、PICK、EDIT 等。data 是以 URI 的形式表示的。例如，要显示一个人的联系方式，您需要创建一个 intent，其中 action 为 VIEW，data 为表示这个人的 URI。

还有一个相关的名叫 IntentFilter 的类。如果说 intent 是一个要做什么事的请求的话，那么 IntentFilter 则是用来描述一个 activity 能够操作哪些 intent。一个能够显示联系人信息的 activity 将声明一个 IntentFilter 来描述如何操作 VIEW 动作和表示这个人的 URI。Activities 在 AndroidManifest.xml 中声明 IntentFilter 类。

屏幕间的切换是通过解析 Intent 实现的。当前向导航时，activity 会自动调用 startActivity(intent myintent)方法。系统会在所有应用程序中定义的 IntentFilter 中查找，选择最匹配 myintent 的 Intent 对应的 activity。新的 activity 收到 intent 的通知后，开始运行。Intents 解析过程在 startActivity()方法被调用时发生，提供了两个好处：

1. Activity 能够通过简单的以 Intent 的形式发送请求来重用其他组件中的功能。
2. Activity 能够在任何时候由一个带有相同 IntentFilter 的 Activity 来替换。

2.3、Service

2.3.1、什么是Service

Service 是具有长生命周期，且没有用户界面的程序。典型的例子是正从播放列表中播放歌曲的媒体播放器。在媒体播放器中，可能将有一个或多个 activities 允许用户选择歌曲并播放它们。然而，音乐回放并不需要 activity 的操纵，因为用户希望音乐播放能够在屏幕切换到新的屏幕时继续播放。在这个例子中，媒体播放器会调用 Context.startService()来启动一个 Service 在后台运行来播放歌曲。系统将会保持音乐回放 service 的运行直到其结束。注意您可以通过 Context.bindService()方法连接一个 service（如果它还没有运行将启动它）。当连接到一个 service 后，我们还可以通过 service 提供的接口与之通讯。正如媒体播放器，我们还可以进行暂停、重播等操作。

Service，看名字就知道跟正常理解的“服务”差不多，后台运行，可交互这样的一个东西。它跟 Activity 的级别差不多，但是他不能自己运行，需要通过某一个 Activity 或者其他 Context 对象来调用， Context.startService() 和 Context.bindService()。

两种启动 Service 的方式有所不同。这里要说明一下的是如果你在 Service 的 onCreate 或者 onStart 做一些很耗时间的东西，最好在 Service 里启动一个线程来完成，因为 Service 是跑在主线程中，会影响到你的 UI 操作或者阻塞主线程中的其他事情。

什么时候需要 Service 呢？比如播放多媒体的时候用户启动了其他 Activity 这个时候程

序要在后台继续播放，比如检测 SD 卡上文件的变化，再或者在后台记录你地理信息位置的改变等等，总之服务嘛，总是藏在后头的。

2.3.2、如何使用Service

那接下来用代码来说明一下怎么使用 Service，这里我们要讲的是 Local Service 也就是你自己的一个 Service，你也可以操作别的应用程序的 service 如果它允许你那么去做的话，这就设计到一个比较麻烦的东西 interprocess communication (IPC)，在不同的进程中通信的机制，这个我自己也还没有用过，等用了以后再跟大伙说说，通常情况下 Local 的就够用啦。

跟 Activity 一样首先你要写一个类继承自 android.app.Service,在这里我叫他 TestService 代码如下：

```
package com.haric.tutorial;

import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.Service;
import android.content.Intent;
import android.os.Binder;
import android.os.IBinder;
import android.util.Log;

public class TestService extends Service {
    private static final String TAG = "TestService";
    private NotificationManager _nm;

    @Override
    public IBinder onBind(Intent i) {
        Log.e(TAG, "=====> TestService.onBind");
        return null;
    }

    public class LocalBinder extends Binder {
        TestService getService() {
            return TestService.this;
        }
    }

    @Override
    public boolean onUnbind(Intent i) {
        Log.e(TAG, "=====> TestService.onUnbind");
    }
}
```



```

return false;
}

@Override
public void onRebind(Intent i) {
    Log.e(TAG, "=====> TestService.onRebind");
}

@Override
public void onCreate() {
    Log.e(TAG, "=====> TestService.onCreate");
    _nm = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
    showNotification();
}

@Override
public void onStart(Intent intent, int startId) {
    Log.e(TAG, "=====> TestService.onStart");
}

@Override
public void onDestroy() {
    _nm.cancel(R.string.service_started);
    Log.e(TAG, "=====> TestService.onDestroy");
}

private void showNotification() {
    Notification notification = new Notification(R.drawable.face_1,
        "Service started", System.currentTimeMillis());

    PendingIntent contentIntent = PendingIntent.getActivity(this, 0,
        new Intent(this, TestServiceHolder.class), 0);

    // must set this for content view, or will throw a exception
    notification.setLatestEventInfo(this, "Test Service",
        "Service started", contentIntent);

    _nm.notify(R.string.service_started, notification);
}
}

```

其中用到 Notification 是为了明显地表明 Service 存活的状态,跟 demo 的 code 学过来的,这样看上去直观一点,更多关于 Notification 的内容以后 UI 部分来写吧,现在就知道怎么使用就好了。



```
@Override
public void onCreate() {
    Log.e(TAG, "=====> TestService.onCreate");
    _nm = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
    showNotification();
}
```

像这样，我在 Service 的几个生命周期函数中加了打印 log 的语句，方便测试。

```
public class LocalBinder extends Binder {
    TestService getService() {
        return TestService.this;
    }
}
```

这个方法是为了让调用者得到这个 Service 并操作它。

Service 本身就这样简单了，你需要做什么就在 onCreate 和 onStart 里做好了，起个线程什么的。

再看一下它的调用者，TestServiceHolder

```
package com.haric.tutorial;

import android.app.Activity;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.IBinder;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

public class TestServiceHolder extends Activity {
    private boolean _isBound;
    private TestService _boundService;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.test_service_holder);
        setTitle("Service Test");
    }
}
```



```
initButtons();
}

private ServiceConnection _connection = new ServiceConnection() {
public void onServiceConnected(ComponentName className, IBinder service) {
    _boundService = ((TestService.LocalBinder)service).getService();

    Toast.makeText(TestServiceHolder.this, "Service connected",
        Toast.LENGTH_SHORT).show();
}

public void onServiceDisconnected(ComponentName className) {
    // unexpectedly disconnected, we should never see this happen.
    _boundService = null;
    Toast.makeText(TestServiceHolder.this, "Service connected",
        Toast.LENGTH_SHORT).show();
}
};

private void initButtons() {
    Button buttonStart = (Button) findViewById(R.id.start_service);
    buttonStart.setOnClickListener(new OnClickListener() {
    public void onClick(View arg0) {
        startService();
    }
    });

    Button buttonStop = (Button) findViewById(R.id.stop_service);
    buttonStop.setOnClickListener(new OnClickListener() {
    public void onClick(View arg0) {
        stopService();
    }
    });

    Button buttonBind = (Button) findViewById(R.id.bind_service);
    buttonBind.setOnClickListener(new OnClickListener() {
    public void onClick(View arg0) {
        bindService();
    }
    });

    Button buttonUnbind = (Button) findViewById(R.id.unbind_service);
    buttonUnbind.setOnClickListener(new OnClickListener() {
```



```
public void onClick(View arg0) {
    unbindService();
}

});

}

private void startService() {
    Intent i = new Intent(this, TestService.class);
    this.startService(i);
}

private void stopService() {
    Intent i = new Intent(this, TestService.class);
    this.stopService(i);
}

private void bindService() {
    Intent i = new Intent(this, TestService.class);
    bindService(i, _connection, Context.BIND_AUTO_CREATE);
    _isBound = true;
}

private void unbindService() {
    if(_isBound) {
        unbindService(_connection);
        _isBound = false;
    }
}
}
```

这里可以看到两种启动方法，start 和 bind，当然也是通过 intent 调用的，在 intent 中指明指定要启动的 Service 的名字，stop 也一样：

```
private void startService() {
    Intent i = new Intent(this, TestService.class);
    this.startService(i);
}

private void stopService() {
    Intent i = new Intent(this, TestService.class);
    this.stopService(i);
}
```

对于 bind 的话，需要一个 ServiceConnection 对象



```

private ServiceConnection _connection = new ServiceConnection() {
public void onServiceConnected(ComponentName className, IBinder service) {
    _boundService = ((TestService.LocalBinder)service).getService();

    Toast.makeText(TestServiceHolder.this, "Service connected",
    Toast.LENGTH_SHORT).show();
}

public void onServiceDisconnected(ComponentName className) {
    // unexpectedly disconnected, we should never see this happen.
    _boundService = null;
    Toast.makeText(TestServiceHolder.this, "Service connected",
    Toast.LENGTH_SHORT).show();
}
};

```

用来把 Activity 和特定的 Service 连接在一起，共同存亡，具体的生命周期细节下一段来讲。

2.3.3、Service 的生命周期

Service 的生命周期方法比 Activity 少一些，只有 onCreate, onStart, onDestroy 我们有两种方式启动一个 Service，他们对 Service 生命周期的影响是不一样的。

1 通过 startService

Service 会经历 onCreate -> onStart
stopService 的时候直接 onDestroy

如果是调用者(TestServiceHolder)自己直接退出而没有调用 stopService 的话，Service 会一直在后台运行。

下次 TestServiceHolder 再起来可以 stopService。

2 通过 bindService

Service 只会运行 onCreate，这个时候 TestServiceHolder 和 TestService 绑定在一起 TestServiceHolder 退出了，Service 就会调用 onUnbind->onDestroyed 所谓绑定在一起就共存亡了。

那有同学问了，要是这几个方法交织在一起的话，会出现什么情况呢？

一个原则是 Service 的 onCreate 的方法只会被调用一次，就是你无论多少次的 startService 又 bindService，Service 只被创建一次。如果先是 bind 了，那么 start 的时候就直接运行 Service 的 onStart 方法，如果先 是 start，那么 bind 的时候就直接运行 onBind 方法。如果你先 bind 上了，就 stop 不掉了，对啊，就是 stopService 不好使了，只能先

UnbindService, 再 StopService,所以是先 start 还是先 bind 行为是有区别的。

2.3.4、判断服务开启状态

```
package com.craining.book.Growth_Need.Alarm;
```

```
import java.util.List;
import android.app.Activity;
import android.app.ActivityManager;
import android.os.Bundle;
import android.widget.TextView;

public class RunningService extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.main);
        TextView mTextView = new TextView(this);
        ActivityManager mActivityManager =
            (ActivityManager) getSystemService(ACTIVITY_SERVICE);

        List<ActivityManager.RunningServiceInfo> mServiceList =
mActivityManager.getRunningServices(30);
        //我要判断的服务名字, 我在 launcher2 里加了一个音乐服务
        final String musicClassName = "com.android.launcher2.MusicService";
        boolean b = MusicServiceIsStart(mServiceList, musicClassName);

        mTextView.setText("你要判断的服务状态为: " + b + "\n" + getServiceClassName(mServiceList));
        setContentView(mTextView);
    }
    //通过 Service 的类名来判断是否启动某个服务
    public static boolean MusicServiceIsStart(List<ActivityManager.RunningServiceInfo> mServiceList,String
className){

        for(int i = 0; i < mServiceList.size(); i++){
            if(className.equals(mServiceList.get(i).service.getClassName())){
                return true;
            }
        }
        return false;
    }
    //获取所有启动的服务的类名
    private String getServiceClassName(List<ActivityManager.RunningServiceInfo> mServiceList){
        String res = "";
```



```

for(int i = 0; i < mServiceList.size(); i++){
    res+=mServiceList.get(i).service.getClassName()+"\n";
}

return res;
}
}

```

2.3.5、获取启动的服务

```

package com.tutor.runningservice;
import java.util.List;
import android.app.Activity;
import android.app.ActivityManager;
import android.os.Bundle;
import android.widget.TextView;
public class RunningService extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.main);
        TextView mTextView = new TextView(this);
        ActivityManager mActivityManager =
            (ActivityManager) getSystemService(ACTIVITY_SERVICE);

        List<ActivityManager.RunningServiceInfo> mServiceList =
mActivityManager.getRunningServices(30);
        //我要判断的服务名字，我在 launcher2 里加了一个音乐服务
        final String musicClassName = "com.android.launcher2.MusicService";

        boolean b = MusicServiceIsStart(mServiceList, musicClassName);

        mTextView.setText("你要判断的服务状态为: " +b+"\n" + getServiceClassName(mServiceList));
        setContentView(mTextView);
    }
    //通过 Service 的类名来判断是否启动某个服务
    private boolean MusicServiceIsStart(List<ActivityManager.RunningServiceInfo> mServiceList,String
className){

        for(int i = 0; i < mServiceList.size(); i++){
            if(className.equals(mServiceList.get(i).service.getClassName())){
                return true;
            }
        }
    }
}

```



```
    }  
    return false;  
}  
//获取所有启动的服务的类名  
private String getServiceClassName(List<ActivityManager.RunningServiceInfo> mServiceList){  
    String res = "";  
    for(int i = 0; i < mServiceList.size(); i++){  
        res+=mServiceList.get(i).service.getClassName()+"\n";  
    }  
  
    return res;  
}
```

2.4、Content Provider

应用程序能够将数据存储存储在文件、SQLite 数据库或者其他有效的机制中。当您想要与其他应用程序共享您的数据时，Content Provider 将会非常有用。Content Provider 也是一个类，其被实现为一组标准的方法，使其他应用程序能够存储和检索此 Content Provider 操作的数据类型。

3、Android UI Layout

3.1、概述

我们对 Android 应用程序运行原理及布局文件可谓有了比较深刻的认识和理解，并且用“Hello World!”程序来实践证明了。在继续深入 [Android 开发之旅](#) 之前，有必要解决前两篇中没有介绍的遗留问题：**View** 的几种布局显示方法，以后就不会在针对布局方面做过多的介绍。**View** 的布局显示方式有下面几种：

线性布局（Linear Layout）、

相对布局（Relative Layout）、

表格布局（Table Layout）、

网格视图（Grid View）、

标签布局（Tab Layout）、

列表视图（List View）、

绝对布局（AbsoluteLayout）。

view 的布局显示概述：



通过前面的学习我们知道：在一个 **Android** 应用程序中，用户界面通过 **View** 和 **ViewGroup** 对象构建。**Android** 中有很多 **View** 和 **ViewGroup**，他们都继承自 **View** 类。**View** 对象是 **Android** 平台上表示用户界面的基本单元。

View 的布局显示方式直接影响用户界面，**View** 的布局方式是指一组 **View** 元素如何布局，准确的说是 **ViewGroup** 中包含的一些 **View** 怎么样布局。**ViewGroup** 类是布局（**layout**）和视图容器（**View container**）的基类，此类也定义了 **ViewGroup.LayoutParams** 类，它作为布局参数的基类，此类告诉父视图其中的子视图想如何显示。例如，XML 布局文件中名为 **layout_something** 的属性（参加 上篇的 4.2 节）。我们要介绍的 **View** 的布局方式的类，都是直接或间接继承自 **ViewGroup** 类，如下图所示：

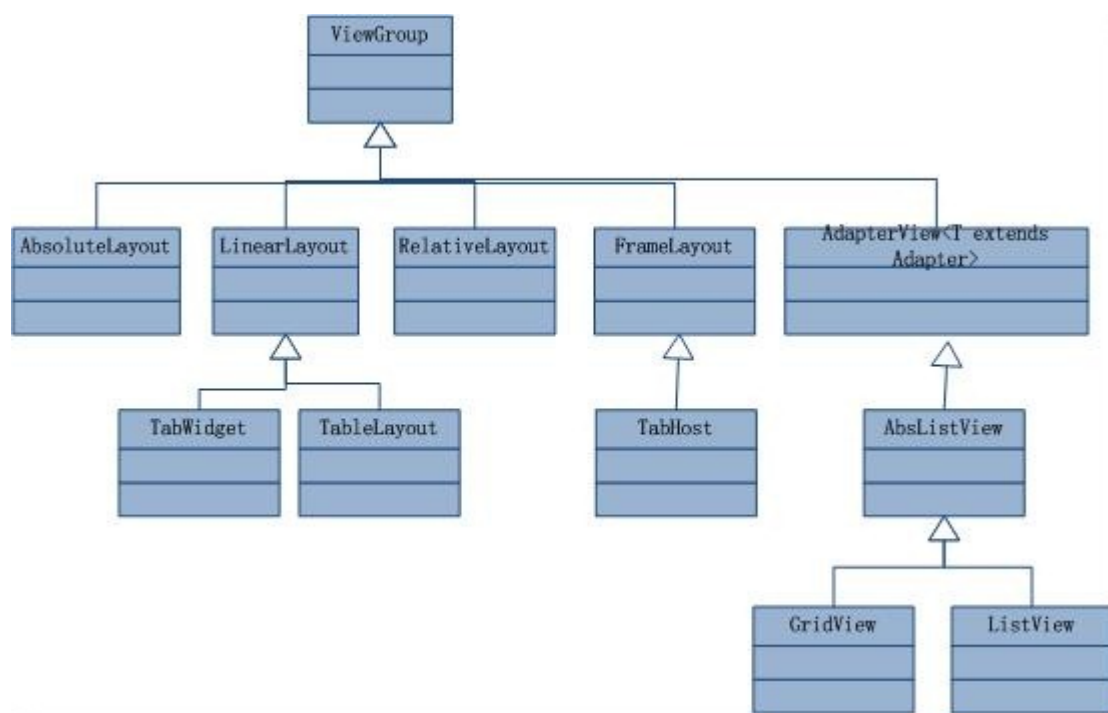


图 1、继承自 ViewGroup 的一些布局类

其实，所有的布局方式都可以归类为 **ViewGroup** 的 5 个类别，即 **ViewGroup** 的 5 个直接子类。其它的一些布局都扩展自这 5 个类。下面分小节分别介绍 **View** 的七种布局显示方式。

3.2、线性布局（Linear Layout）

线性布局：是一个 **ViewGroup** 以线性方向显示它的子视图（**view**）元素，即垂直地或水平地。之前我们的 Hello World! 程序中 **view** 的布局方式就是线性布局的，一定不陌生！如下所示 **res/layout/main.xml**：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal"><!-- have an eye on ! -->
    <Button android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button1"
        android:layout_weight="1"
    />
    <Button android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button2"
        android:layout_weight="1"
    />
    <Button android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button3"
        android:layout_weight="1"
    />
    <Button android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button4"
        android:layout_weight="1"
    />
    <Button android:id="@+id/button5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button5"
        android:layout_weight="1"
    />
</LinearLayout>
```

从上面可以看出根 LinearLayout 视图组 (ViewGroup) 包含 5 个 Button，它的子元素是以线性方式 (horizontal，水平的) 布局，运行效果如下图所示：





图 2、线性布局（水平或者说是横向）

如果你在 `android:orientation="horizontal"` 设置为 `vertical`，则是垂直或者说是纵向的，如下图所示：

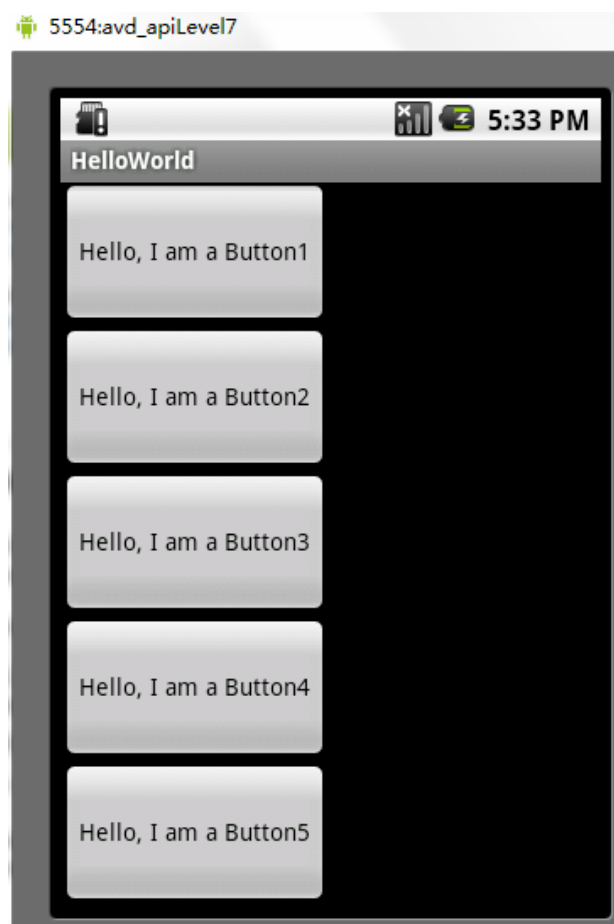


图 3、线性布局（垂直或者说是纵向）

2.1、Tips: `android:layout_weight="1"`

这个属性很关键，如果你没有显示设置它，它默认为 0。把上面布局文件（水平显示的那个）中的这个属性都去掉，运行会得出如下结果：



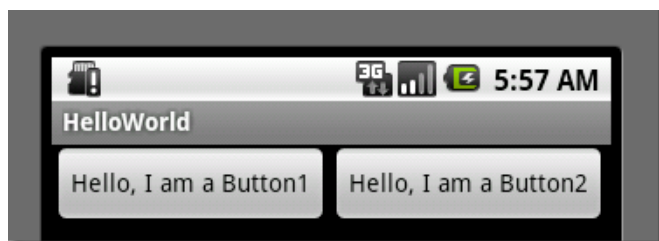


图 4、layout_weight 属性

没有了这个属性，我们本来定义的 5 个 Button 运行后却只显示了 2 个 Button，为什么呢？

"weight"顾名思义是权重的意思，layout_weight 用于给一个线性布局中的诸多视图的重要程度赋值。所有的视图都有一个 layout_weight 值，默认为零，意思是需要显示多大的视图就占据多大的屏幕空间。这就不难解释为什么会造成上面的情况了：Button1~Button5 都设置了 layout_height 和 layout_width 属性为 wrap_content 即包住文字内容，他们都没有设置 layout_weight 属性，即默认为 0，这样 Button1 和 Button2 根据需要的内容占据了整个屏幕，别的就显示不了啦！

若赋一个高于零的值，则将父视图中的可用空间分割，分割大小具体取决于每一个视图的 layout_weight 值以及该值在当前屏幕布局的整体 layout_weight 值和在其其它视图屏幕布局的 layout_weight 值中所占的比率而定。举个例子：比如说我们在 水平方向上有一个文本标签和两个文本编辑元素。该文本标签并无指定 layout_weight 值，所以它将占据需要提供的最少空间。如果两个文本编辑元素每一个的 layout_weight 值都设置为 1，则两者平分在父视图布局剩余的宽度(因为我们声明这两者的重要度相等)。如果两个文本编辑元素其中第一个的 layout_weight 值设置为 1，而第二个的设置 2，则剩余空间的三分之二分给第一个，三分之一分给第二个(数值越小，重要度越高)。

3.3、相对布局 (Relative Layout)

相对布局：是一个 ViewGroup 以相对位置显示它的子视图 (view) 元素，一个视图可以指定相对于它的兄弟视图的位置 (例如在给定视图的左边或者下面) 或相对于 [RelativeLayout](#) 的特定区域的位置 (例如底部对齐，或中间偏左)。

相对布局是设计用户界面的有力工具，因为它消除了嵌套视图组。如果你发现你使用了多个嵌套的 [LinearLayout](#) 视图组后，你可以考虑使用一个 [RelativeLayout](#) 视图组了。看下面的 res/layout/main.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:id="@+id/label"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Type here:"/>
    <EditText
        android:id="@+id/entry"
```

```

        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@android:drawable/editbox_background"
        android:layout_below="@id/label"/><!-- have an eye on ! -->
    <Button
        android:id="@+id/ok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/entry" <!-- have an eye on ! -->
        android:layout_alignParentRight="true" <!-- have an eye on ! -->
        android:layout_marginLeft="10dip"
        android:text="OK" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/ok" <!-- have an eye on ! -->
        android:layout_alignTop="@id/ok" <!-- have an eye on ! -->
        android:text="Cancel" />
</RelativeLayout>

```

从上面的布局文件我们知道，RelativeLayout视图组包含一个TextView、一个EditText、两个Button，注意标记了<!-- have an eye on ! -->的属性，在使用相对布局方式中就是使用这些类似的属性来定位视图到你想要的位置，它们的值是你参照的视图的id。这些属性的意思很简单，就是英文单词的直译，就不多做介绍了。运行之后，得如下结果：

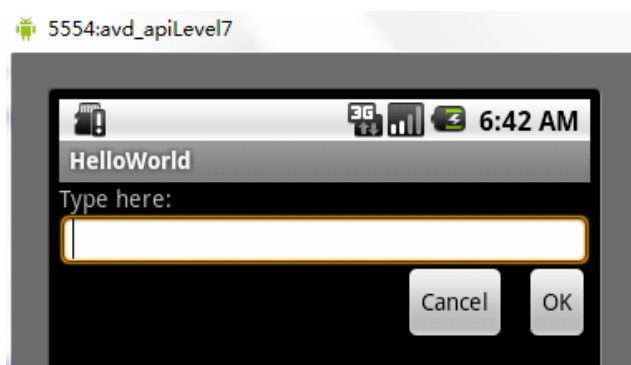


图 5、相对布局

3.4、TableLayout

TableLayout (一)

TableLayout 跟 TableRow 是一组搭配使用的布局，TableLayout 置底，TableRow 在 TableLayout 的上面，而 Button、TextView 等控件就在 TableRow 之上，另外，TableLayout 之

上也可以单独放控件。TableLayout 是一个使用复杂的布局，最简单的用法就仅仅是拖拉控件做出个界面，但实际上，会经常在代码里使用 TableLayout，例如做出表格的效果。本文主要介绍 TableLayout 的基本使用方法。

TableLayout 经常用的属性是：



android:collapseColumns: 以第 0 行为序，隐藏指定的列：

android:collapseColumns 该属性为空时，如下图：



把 android:collapseColumns=0,2-----》意思是把第 0 和第 2 列去掉，如下图：



android:shrinkColumns: 以第 0 行为序，自动延伸指定的列填充可用部分：

当 LayoutRow 里面的控件还没有布满布局时，shrinkColumns 不起作用，如下图：

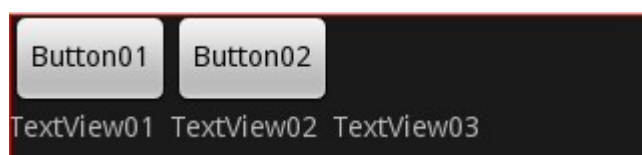


设置了 shrinkColumns=0,1,2，布局完全没有改变，因为 LayoutRow 里面还剩足够的空间。

当 LayoutRow 布满控件时，如下图：

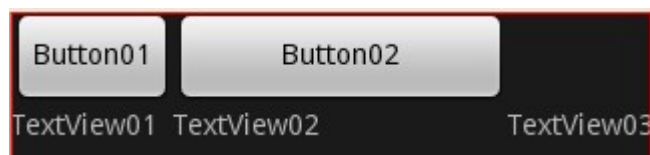


设置设置了 `shrinkColumns=2`，则结果如下图，控件自动向垂直方向填充空间：



`android:stretchColumns`：以第 0 行为序，尽量把指定的列填充空白部分：

设置 `stretchColumns=1`，则结果如下图，第 1 列被尽量填充(Button02 与 TextView02 同时向右填充,直到 TextView03 被压挤到最后边)。



Android 的 `TableLayout` + `TableRow` 虽然使用有点复杂,但是功能很强大。。。。。Android 提供了很多布局属性,但是手机程序的界面没有 PC 那么花俏,所以常用的就那几项而已。。

TableLayout (二)

TableLayout (一) 主要将如何 UI 设计器设计 `TableLayout` + `TableRow`，由于实际应用中，经常需要在代码里往 `TableLayout` 添加数据(9 宫图也可以用 `TableLayout` 做出来)，本文就是介绍这方面的简单使用方法。

`main.xml` 的代码如下，用到 `TableLayout` 的 ID 为 `TableLayout01`：

view plain copy to clipboard print?

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TableLayout
        android:id="@+id/TableLayout01"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
    </TableLayout>
</LinearLayout>
<?xml version="1.0" encoding="utf-8"?>
```



```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TableLayout
        android:id="@+id/TableLayout01"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
    </TableLayout>
</LinearLayout>

```

JAVA代码如下:

```

view plaincopy to clipboardprint?
package com.LayoutDemo;
import com.LayoutDemo.R;
import android.app.Activity;
import android.os.Bundle;
import android.view.ViewGroup;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;
public class LayoutDemo extends Activity {
    /** Called when the activity is first created. */
    private final int WC = ViewGroup.LayoutParams.WRAP_CONTENT;
    private final int FP = ViewGroup.LayoutParams.FILL_PARENT;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //新建TableLayout01 的实例
        TableLayout tableLayout = (TableLayout)findViewById(R.id.TableLayout01);
        //全部列自动填充空白处
        tableLayout.setStretchAllColumns(true);
        //生成 10 行， 8 列的表格
        for(int row=0;row<10;row++)
        {
            TableRow tableRow=new TableRow(this);
            for(int col=0;col<8;col++)
            {
                //tv用于显示
                TextView tv=new TextView(this);

```

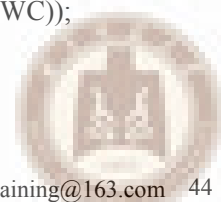


```

        tv.setText(""+col+","+row+"");
        tableRow.addView(tv);
    }
    //新建的TableRow添加到TableLayout
    tableLayout.addView(tableRow, new TableLayout.LayoutParams(FP, WC));
}
}
}
package com.LayoutDemo;
import com.LayoutDemo.R;
import android.app.Activity;
import android.os.Bundle;
import android.view.ViewGroup;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;
public class LayoutDemo extends Activity {
    /** Called when the activity is first created. */
    private final int WC = ViewGroup.LayoutParams.WRAP_CONTENT;
    private final int FP = ViewGroup.LayoutParams.FILL_PARENT;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //新建TableLayout01 的实例
        TableLayout tableLayout = (TableLayout)findViewById(R.id.TableLayout01);
        //全部列自动填充空白处
        tableLayout.setStretchAllColumns(true);
        //生成 10 行， 8 列的表格
        for(int row=0;row<10;row++)
        {
            TableRow tableRow=new TableRow(this);
            for(int col=0;col<8;col++)
            {
                //tv用于显示
                TextView tv=new TextView(this);
                tv.setText(""+col+","+row+"");
                tableRow.addView(tv);
            }
            //新建的TableRow添加到TableLayout
            tableLayout.addView(tableRow, new TableLayout.LayoutParams(FP, WC));
        }
    }
}

```



```
}
```

结果如下图:

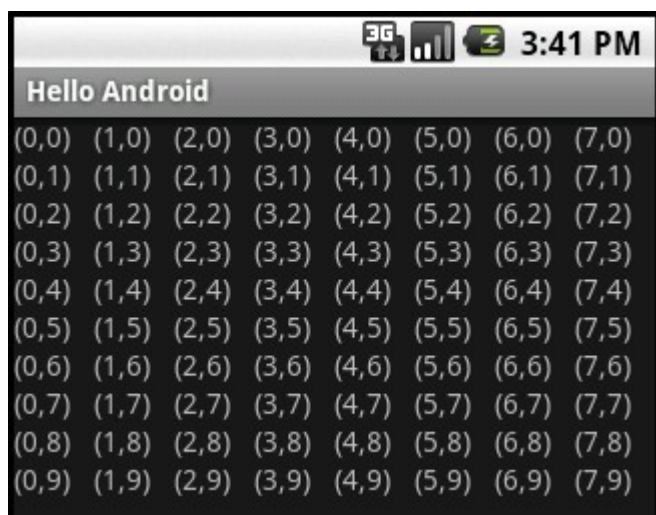


Table Layout (三)

表格布局: 是一个ViewGroup以表格显示它的子视图(view)元素, 即行和列标识一个视图的位置。其实Android的表格布局跟HTML中的表格布局非常类似, TableRow 就像HTML表格的<tr>标记。

用表格布局需要知道以下几点:

- **android:shrinkColumns**, 对应的方法: **setShrinkAllColumns(boolean)**, 作用: 设置表格的列是否收缩(列编号从 0 开始, 下同), 多列用逗号隔开(下同), 如**android:shrinkColumns="0,1,2"**, 即表格的第 1、2、3 列的内容是收缩的以适合屏幕, 不会挤出屏幕。
- **android:collapseColumns**, 对应的方法: **setColumnCollapsed(int,boolean)**, 作用: 设置表格的列是否隐藏
- **android:stretchColumns**, 对应的方法: **setStretchAllColumns(boolean)**, 作用: 设置表格的列是否拉伸

看下面的 **res/layout/main.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:shrinkColumns="0,1,2"><!-- have an eye on !
```



android:stretchColumns="0,1"> 缩减与延伸-->

```

<TableRow><!-- row1 -->
<Button android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello, I am a Button1"
    android:layout_column="0"
    />
<Button android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello, I am a Button2"
    android:layout_column="1"
    />
</TableRow>
<TableRow><!-- row2 -->
<Button android:id="@+id/button3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello, I am a Button3"
    android:layout_column="1"
    />
<Button android:id="@+id/button4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello, I am a Button4"
    android:layout_column="1"
    />
</TableRow>
<TableRow>
<Button android:id="@+id/button5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello, I am a Button5"
    android:layout_column="2"
    />
</TableRow>
</TableLayout>

```

运行之后可以得出下面的结果：





图 6、表格布局

3.5、AbsoluteLayout

在 Android UI 中，最基本的构建单位 (building block) 是 `android.view.View`。一个 `View` 占据屏幕上的一个矩形区域，并负责该区域的绘画和事件处理。`View` 有一些子类，比如 `ImageView`、`TextView` 等可分别用来显示图像、文字…… `View` 还有一个特殊的子类 `ViewGroup`，`ViewGroup` 在 UI layout 中充当“容器”的角色，用以“包含”其他 `View` 以及 `ViewGroup`：

viewgroup.png

由于 `ViewGroup` 是一个 `abstract class` 无法直接实例化，所以在 layout 中真正充当“容器”角色的应该是 `ViewGroup` 的子类：`AbsoluteLayout`、`FrameLayout`、`LinearLayout`、`RelativeLayout` 等。在实际的 UI 编程中，使用不同的 `Layout` 类作为容器，对该容器中的各个子 `View` 的排列方式有很大影响。比如，`LinearLayout` 中的各个子 `View` 按照横向或者纵向线性排列；而 `AbsoluteLayout` 中各个子 `View` 可以指定以像素为单位的“绝对”位置。`AbsoluteLayout` 的这种“绝对”定位的布局方式和我们非常熟悉的 Windows 编程中的 `SetWindowPos()` 或 `Form1.Left = 10` 的布局方式是一样的，比较简单：

现在我们新建一个 Android 工程中，在其主 `Activity` 类中添加如下三个成员：

```
private AbsoluteLayout al;
private TextView tv;
private View v;
```

改写这个类的 `onCreate` 方法如下：

```
public void onCreate(Bundle icle) {
    super.onCreate(icle);
```



构造一个 `AbsoluteLayout`，设置其背景色

```
al = new AbsoluteLayout(this);
al.setBackgroundColor(Color.YELLOW);
```

构造一个 `TextView` 并设置其 `text` 和 背景色

```
tv = new TextView(this);
tv.setText("Android is a software stack for mobile devices that includes an operating
system, middleware and key applications. ");
tv.setBackgroundColor(Color.BLUE);
```

用该 `View` 在父 `View` 中的 `width`，`height`，`x`，`y` 作为参数构造一个 `AbsoluteLayout.LayoutParams`

```
AbsoluteLayout.LayoutParams tvLP = new AbsoluteLayout.LayoutParams(70, 50,
10, 20);
```

把这个 `TextView` 加入到 `AbsoluteLayout` 中，并应用上一步创建的 `LayoutParams`。这样 `TextView` 就会显示在我们指定的位置上了。

```
al.addView(tv, tvLP);
```

```
v = new View(this);
v.setBackgroundColor(Color.RED);
```

```
AbsoluteLayout.LayoutParams vLP = new AbsoluteLayout.LayoutParams(70, 50,
90, 70);
```

也可以先为子 `View` 设置 `LayoutParams`，然后再调用一个参数的 `ViewGroup.addView(View)` 来添加。效果是一样的。

```
v.setLayoutParams(vLP);
al.addView(v);
```

设置 `al` 为本 `Activity` 的 `content`

这样，在该 `Activity` 被调用时，就会显示该 `AbsoluteLayout` 和其子 `View`

```
this setContentView(al);
```

```
}
```

4、Android UI 控件

4.1、ImageButton

4.1.1、图案填充问题

<RelativeLayout

xmlns:android="http://schemas.android.com/apk/res/android"

android:layout_width="fill_parent"

android:layout_height="fill_parent"




```

<ImageButton
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@color/background"
    android:layout_marginTop="78dip"
    android:paddingLeft="120dip"
    android:src="@drawable/volume_selector"
/>

```

其中 color 为<color name="background">#00ffffff</color>

也有说可以用 android:adjustViewBounds="true"做的，没有验证过。

4.2、TextView

4.2.1、动态滚动

据说可以实现具有滚动效果的 textView

在 xml 文件中 <TextView 标签中添加如下属性

```

android:singleLine="true"
android:ellipsize="marquee"
android:marqueeRepeatLimit="marquee_forever"

```

4.3、EditText

4.3.1、光标选择

提起 Android 的 EditText 的光标选择问题，可以通过 android.text.Selection 包提供的方法来实现，Android SDK 提供了有关光标选择的多种方法，比如说 getSelectionEnd、getSelectionStart、removeSelection、selectAll、setSelection，详细的参数声明如下：

```
final static int getSelectionEnd(CharSequence text)
```

Return the offset of the selection edge or cursor, or -1 if there is no selection or cursor.

```
final static int getSelectionStart(CharSequence text)
```

Return the offset of the selection anchor or cursor, or -1 if there is no selection or cursor.

```
final static void removeSelection(Spannable text)
```

Remove the selection or cursor, if any, from the text.



```
final static void selectAll(Spannable text)
```

Select the entire text.

```
final static void setSelection(Spannable text, int index)
```

Move the cursor to offset index.

```
static void setSelection(Spannable text, int start, int stop)
```

Set the selection anchor to start and the selection edge to stop.

Android123 提示大家，从上面的参数来看，可以发现 `Spannable` 类型，常规我们的 `EditText` 中的编辑中 `Editable` 直接实现 `Spannable` 接口，所以我们可以通过下面的方法来设置选择：

```
Editable ea= etEdit.getText(); //etEdit 为 EditText
```

```
Selection.setSelection(ea, ea.length()-1); // Android 开发网提示这里 ea 的长度必须大于 1。否则会有异常发生。
```

4.4、TitleBar

4.4.1、非全屏状态下不显示 title 标题栏

答：`requestWindowFeature(Window.FEATURE_NO_TITLE);` //设置是否显示 title

```
getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,WindowManager.LayoutParams.FLAG_FULLSCREEN); //设置是否全屏。
```

4.4.2、标题栏进度指示器

Android 自带的浏览器在载入网页时等待时间可能会在标题栏的右上角有一个小圆圈在不断旋转，由于其不包含具体进度，很多网友可能没有找到详细的操作方法在 SDK 中。作为标题栏进度指示器其实属于 `Activity` 类的方法。

在使用时我们首先需要在 `setContentView` 之前声明

```
requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);
```

在需要显示进度时调用

```
setProgressBarIndeterminateVisibility(true);
```

停止时调用

```
setProgressBarIndeterminateVisibility(false);
```



4.4.3、titleBar 高级实现方法(更美观)

很多细心的网友发现 Android 浏览器的标题栏 TitleBar 的功能比较多，细心的网友在查看 Browser 时会发现，从左到右依次为网站图标(favicon)、标题、最右边的动画进度条(圆圈)、背景进度条(和前面的不在一层)，今天我们就一起来看看 Android 标题栏高级实现方法。

在 Android Browser 程序中标题栏是自绘的，TitleBar 类继承于线性布局 LinearLayout 类，通过 LayoutInflater 调用 layout 中的 xml 布局文件实现相关方法

```
public class TitleBar extends LinearLayout {
    private TextView mTitle; //标题文字
    private Drawable mCloseDrawable;
    private ImageView mRtButton;
    private Drawable mCircularProgress; //圆圈进度指示
    private ProgressBar mHorizontalProgress; //水平进度条
    private ImageView mFavicon; //网站图标
    private ImageView mLockIcon;
    private Drawable mStopDrawable; //停止状态的图标
    private Drawable mBookmarkDrawable; //是一个书签的图标
    private boolean mInLoad;
    private BrowserActivity mBrowserActivity;
    private Drawable mGenericFavicon; //如果站点没有 favicon.ico 时显示的默认图标
    private int mIconDimension;
    private View mTitleBg; //文字的背景
    private MyHandler mHandler;
    private static int LONG_PRESS = 1;
    public TitleBar(BrowserActivity context) {
        super(context, null);
        mHandler = new MyHandler();
        LayoutInflater factory = LayoutInflater.from(context);
        factory.inflate(R.layout.title_bar, this); //从 xml 文件创建，android123 提示大家，该文件的详细内容在本
        段代码最下方。
        mBrowserActivity = context;
        mTitle = (TextView) findViewById(R.id.title);
        mTitle.setCompoundDrawablePadding(5);
        mTitleBg = findViewById(R.id.title_bg);
        mLockIcon = (ImageView) findViewById(R.id.lock);
        mFavicon = (ImageView) findViewById(R.id.favicon);
        mRtButton = (ImageView) findViewById(R.id.rt_btn);
        Resources resources = context.getResources();
        mCircularProgress = (Drawable) resources.getDrawable(com.android.internal.R.drawable.search_spinner);
        mIconDimension = (int) TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP,
        20f,resources.getDisplayMetrics());
        mCircularProgress.setBounds(0, 0, mIconDimension, mIconDimension);
```

```

mHorizontalProgress = (ProgressBar) findViewById(R.id.progress_horizontal);
mGenericFavicon = context.getResources().getDrawable(R.drawable.app_web_browser_sm);
}
private class MyHandler extends Handler {
public void handleMessage(Message msg) {
if (msg.what == LONG_PRESS) {
mTitleBg.setPressed(false);
mBrowserActivity.showTitleBarContextMenu();
}
}
};
@Override
protected void onCreateContextMenu(ContextMenu menu) { //创建上下文菜单，相关的 xml 代码在本文最
后
MenuInflater inflater = mBrowserActivity.getMenuInflater();
inflater.inflate(R.menu.title_context, menu);
}
@Override
public boolean onTouchEvent(MotionEvent event) {
switch (event.getAction()) {
case MotionEvent.ACTION_DOWN:
if ((int) event.getX() > mTitleBg.getRight()) {
mRtButton.setPressed(true);
} else {
mTitleBg.setPressed(true);
mHandler.sendMessageDelayed(mHandler.obtainMessage(
LONG_PRESS),
ViewConfiguration.getLongPressTimeout());
}
break;
case MotionEvent.ACTION_MOVE:
int slop = ViewConfiguration.get(mBrowserActivity)
.getScaledTouchSlop();
if ((int) event.getY() > getHeight() + slop) {
mTitleBg.setPressed(false);
mRtButton.setPressed(false);
mHandler.removeMessages(LONG_PRESS);
break;
}
int x = (int) event.getX();
int titleRight = mTitleBg.getRight();
if (mTitleBg.isPressed() && x > titleRight + slop) {
mTitleBg.setPressed(false);
mHandler.removeMessages(LONG_PRESS);
}
}
}

```



```
} else if (mRtButton.isPressed() && x < titleRight - slop) {
mRtButton.setPressed(false);
}
break;
case MotionEvent.ACTION_CANCEL:
mRtButton.setPressed(false);
mTitleBg.setPressed(false);
mHandler.removeMessages(LONG_PRESS);
break;
case MotionEvent.ACTION_UP:
if (mRtButton.isPressed()) {
if (mInLoad) {
mBrowserActivity.stopLoading();
} else {
mBrowserActivity.bookmarksOrHistoryPicker(false);
}
mRtButton.setPressed(false);
} else if (mTitleBg.isPressed()) {
mHandler.removeMessages(LONG_PRESS);
mBrowserActivity.onSearchRequested();
mTitleBg.setPressed(false);
}
break;
default:
break;
}
return true;
}
boolean isInLoad() {
return mInLoad;
}
void setFavicon(Bitmap icon) {
Drawable[] array = new Drawable[3];
array[0] = new PaintDrawable(Color.BLACK);
PaintDrawable p = new PaintDrawable(Color.WHITE);
array[1] = p;
if (icon == null) {
array[2] = mGenericFavicon;
} else {
array[2] = new BitmapDrawable(icon);
}
LayerDrawable d = new LayerDrawable(array);
d.setLayerInset(1, 1, 1, 1, 1);
d.setLayerInset(2, 2, 2, 2, 2);
```



```

mFavicon.setImageDrawable(d);
}
void setLock(Drawable d) {
if (null == d) {
mLockIcon.setVisibility(View.GONE);
} else {
mLockIcon.setImageDrawable(d);
mLockIcon.setVisibility(View.VISIBLE);
}
}
void setProgress(int newProgress) {
if (newProgress >= mHorizontalProgress.getMax()) {
mTitle.setCompoundDrawables(null, null, null, null);
((Animatable) mCircularProgress).stop();
mHorizontalProgress.setVisibility(View.INVISIBLE);
if (mBookmarkDrawable != null) {
mRtButton.setImageDrawable(mBookmarkDrawable);
}
mInLoad = false;
} else {
mHorizontalProgress.setProgress(newProgress);
if (!mInLoad && getWindowToken() != null) {
mTitle.setCompoundDrawables(null, null, mCircularProgress,
null);
((Animatable) mCircularProgress).start();
mHorizontalProgress.setVisibility(View.VISIBLE);
if (mBookmarkDrawable == null) {
mBookmarkDrawable = mRtButton.getDrawable();
}
if (mStopDrawable == null) {
mRtButton.setImageResource(R.drawable.ic_btn_stop_v2);
mStopDrawable = mRtButton.getDrawable();
} else {
mRtButton.setImageDrawable(mStopDrawable);
}
mInLoad = true;
}
}
}
void setTitleAndUrl(CharSequence title, CharSequence url) {
if (url == null) {
mTitle.setText(R.string.title_bar_loading);
} else {
mTitle.setText(url.toString());
}
}

```



```

    }
}
void setToTabPicker() {
    mTitle.setText(R.string.tab_picker_title);
    setFavicon(null);
    setLock(null);
    mHorizontalProgress.setVisibility(View.GONE);
}
}

```

本文的相关的 title_bar.xml 布局文件内容，其中大家可以理解下部分图标在创建时使用隐藏 GONE 的 visibility 显示属性，通过条件来设置其显示

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```

    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:paddingLeft="8dip"
    android:paddingRight="12dip"
    android:paddingTop="2dip"
    android:paddingBottom="1dip"
    android:background="@drawable/search_plate_browser" >

```

```
<ProgressBar android:id="@+id/progress_horizontal"
```

```

    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="fill_parent"
    android:layout_height="5dip"
    android:max="100"
/>

```

```
<LinearLayout
```

```

    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
>

```

```
<LinearLayout android:id="@+id/title_bg"
```

```

    android:background="@drawable/title_text"
    android:layout_width="0dip"
    android:layout_weight="1.0"
    android:layout_height="wrap_content"
    android:gravity="center_vertical"
    android:orientation="horizontal"
>

```



```

<ImageView android:id="@+id/favicon"
    android:layout_width="20dip"
    android:layout_height="20dip"
    android:layout_marginLeft="3dip"
/>

<ImageView android:id="@+id/lock"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="6dip"
    android:visibility="gone"
/>

<TextView
    android:id="@+id/title"
    android:layout_height="wrap_content"
    android:layout_width="0dip"
    android:layout_weight="1.0"
    android:paddingLeft="8dip"
    android:paddingRight="6dip"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:textColor="@color/black"
    android:gravity="center_vertical"
    android:singleLine="true"
    android:ellipsize="end"
/>

</LinearLayout>

<ImageView
    android:id="@+id/rt_btn"
    android:layout_width="wrap_content"
    android:layout_height="fill_parent"
    android:layout_marginLeft="6dip"
    android:scaleType="center"
    android:layout_marginBottom="4dip"
    android:background="@drawable/btn_bookmark"
    android:src="@drawable/ic_btn_bookmarks"
/>

</LinearLayout>

</LinearLayout>

```

以下为 title_context.xml，里面仅有两条一个为分享该页和复制本页的 URL

```

<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/title_bar_share_page_url"
        android:title="@string/share_page"/>
    <item android:id="@+id/title_bar_copy_page_url"
        android:title="@string/copy_page"/>

```



</menu>

4.4.4、获取标题栏和状态栏高度

1. 获取状态栏高度:

decorView 是 window 中的最顶层 view, 可以从 window 中获取到 decorView, 然后 decorView 有个 getWindowVisibleDisplayFrame 方法可以获取到程序显示的区域, 包括标题栏, 但不包括状态栏。

于是, 我们就可以算出状态栏的高度了。

```
Rect frame = new Rect();
getWindow().getDecorView().getWindowVisibleDisplayFrame(frame);
int statusBarHeight = frame.top;
```

2. 获取标题栏高度:

getWindow().findViewById(Window.ID_ANDROID_CONTENT) 这个方法获取到的 view 就是程序不包括标题栏的部分, 然后就可以知道标题栏的高度了。

```
int contentTop = getWindow().findViewById(Window.ID_ANDROID_CONTENT).getTop();
//statusBarHeight 是上面所求的状态栏的高度
```

```
int titleBarHeight = contentTop - statusBarHeight
```

4.4.5、标题栏显示简单的进度框

很多网友可能发现, 比如 Android 自带的浏览器在载入网页时等待时间可能会在标题栏的右上角有一个小圆圈在不断旋转, 由于其不包含具体进度, 很多网友可能没有找到详细的操作方法在 SDK 中。作为标题栏进度指示器其实属于 Activity 类的方法。

在 **setContentView** 之前声明 :

```
requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS); ,
```

在需要显示进度时调用 `setProgressBarIndeterminateVisibility(true);`

停止时调用 `setProgressBarIndeterminateVisibility(false);`



4.5、Menu

4.5.1、简单的代码

```
public static final int ITEM_1_ID = Menu.FIRST;
public static final int ITEM_2_ID = Menu.FIRST + 1;
public static final int ITEM_3_ID = Menu.FIRST + 2;

public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    //不带图标的 menu
    menu.add(0, ITEM_1_ID, 0, "item-1");
    //带图标的 menu
    menu.add(0, ITEM_2_ID, 1, "item-2").setIcon(R.drawable.editbills2);
    menu.add(0, ITEM_3_ID, 2, "item-3").setIcon(R.drawable.bills1);
    return true;
}

public boolean onOptionsItemSelected(MenuItem item){
    switch (item.getItemId()) {
        case 1:
            Toast.makeText(this, "menu1", Toast.LENGTH_SHORT).show();
            return true;
        case 2:

            return true;
        case 3:

            return true;
    }
    return false;
}
```

4.5.2、menu实现的两种方法

大部分的应用程序都包括两种人机互动方式，一种是直接通过GUI的 Views，其可以满足大部分的交互操作。另外一种是用Menu，当按下Menu按钮后，会弹出与当前活动状态下的应用程序相匹配的菜单。

这两种方式相比较都有各自的优势，而且可以很好的相辅相成，即便用户可以由主界面完成大部分操作，但是适当的拓展Menu功能可以更加完善应用程序，至少用户可以通过排列整齐的 按钮清晰的了解当前模式下可以使用的功能。

有两种方法可以为Android APPs添加菜单功能,下边将对设置过程给出详细的介绍:

第一种方法,通过Layout来添加静态菜单元素。

一般情况下,开发者在res/Layout路径下来定义应用程序的GUI。应用Eclipse创建一个新项目后,可以看到res/layout中存在一个预置的main.xml文件,其作为程序默认启动界面。同样,可以通过这种方式创建一个静态的Menu,创建方法参阅下边的源代码:

[View Code XML](#)

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/previous"
        android:title="@string/previous"
        android:enabled="false"
        android:icon="@android:drawable/ic_media_previous"/>
    <!--these may not be available in next api (level > 3), so be carefull-->
    <item
        android:id="@+id/play_pause"
        android:title="@string/play"
        android:icon="@android:drawable/ic_media_play"/>
    <item
        android:id="@+id/next"
        android:title="@string/next"
        android:icon="@android:drawable/ic_menu_next"/>
</menu>
```

在Activity类中调用刚刚创建的Menu, 首先将当前的Activity与指定的Menu XML相关联:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    getMenuInflater().inflate(R.layout.menu_mainactivity, menu);
    return true;
}
```

实现onOptionsItemSelected方法: (其目的是捕捉到菜单触发事件后, 对具体触发的选项作出响应, 实际调用的函数包含在各自的case中)

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.previous:
            previous(); //go to previous song in the playlist
            return true;
```



```

        case R.id.play_pause:
            isPlaying() ? pause() : play(); //toggle play/pause
            return true;
        case R.id.next:
            next(); //go to next song in the playlist
            return true;
    }
    return false; //should never happen
}

```

最后可以通过onPrepareOptionsMenu方法初始化Menu Items的属性:

```

@Override
public boolean onPrepareOptionsMenu(Menu menu) {
    //set play_pause menu item look
    if(isPlaying()) {
        menu
            .findItem(R.id.play_pause)
            .setTitle(R.string.pause)
            .setIcon(android.R.drawable.ic_media_pause);
    } else {
        menu
            .findItem(R.id.play_pause)
            .setTitle(R.string.play)
            .setIcon(android.R.drawable.ic_media_play);
    }
    return true;
}

```

大部分程序都通过这种方式添加Menu菜单功能, 而且通过以上的步骤来看, 其实实现方法非常简单。

第二种方法, 在Activity类中动态创建Menu。

首先需要定义Menu Item识别序号:

```

public static final MENU_PREVIOUS = 0; //no more R.ids
public static final MENU_PLAY_PAUSE = 1;
public static final MENU_NEXT = 2;

```

实现onCreateOptionsMenu()方法:(第一种方法中已经通过xml定义了现成的Menu结构, 所以不需要应用这个方法)

```

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    menu

```



```

        .add(0, MENU_PREVIOUS, 0, R.string.previous)
        .setIcon(android.R.drawable.ic_media_previous);
    menu
        .add(0, MENU_PLAY_PAUSE, 0, R.string.play)
        .setIcon (android.R.drawable.ic_media_play);
    menu
        .add(0, MENU_NEXT, 0, R.string.next)
        .setIcon(android.R.drawable.ic_media_next);
    return true;
}

```

引用与第一种方法相同的方式来捕捉菜单的行为:

@Override

```

public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case MENU_PREVIOUS:
            previous(); //go to previous song in the playlist
            return true;
        case MENU_PLAY_PAUSE:
            isPlaying() ? pause() : play(); //toggle play/pause
            return true;
        case MENU_NEXT:
            next(); //go to next song in the playlist
            return true;
    }
    return false; //should never happen
}

```

对以上两种方法的补充:

根据需要设置不同Menu Item的属性:

```
menu.findItem(R.id.next).setEnabled(false);
```

设置Menu Item从属关系(添加子父级别):

直接写在方法中:

```

menu
    .addSubMenu(R.id.repeat)
    .add(R.id.one)
    .add(R.id.all)
    .add(R.id.none);

```



直接定义在XML Layout中:

[View Code XML](#)

```
<item android:id="@+id/repeat" android:title="@string/repeat">
<menu>
    <item android:id="@+id/one" android:title="@string/repeat_one"></item>
    <item android:id="@+id/all" android:title="@string/repeat_all"></item>
    <item android:id="@+id/none" android:title="@string/repeat_none"></item>
</menu>
```

这两种不同的方法实现的目的是一样的，而且不存在本质上的却别，具体根据实际情况(根据项目的结构需要或者团队开发标准)选择合适的方法来创建 Menu。

4.5.3、自定义MENU背景

/* The Options Menu (the one that pops up on pressing the menu button on the emulator)

* can be customized to change the background of the menu */

```
package com.tut.menu;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.os.Handler;
import android.util.AttributeSet;
import android.util.Log;
import android.view.InflateException;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.View;
import android.view.LayoutInflater.Factory;

public class Options_Menu extends Activity {

    private static final String TAG = "DEBUG";

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```



```
setContentView(R.layout.main);

}

/* Invoked when the menu button is pressed */

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // TODO Auto-generated method stub
    super.onCreateOptionsMenu(menu);
    MenuInflater inflater = new MenuInflater(getApplicationContext());
    inflater.inflate(R.menu.options_menu, menu);
    setMenuBackground();
    return true;
}

/*IconMenuItemView is the class that creates and controls the options menu
 * which is derived from basic View class. So We can use a LayoutInflater
 * object to create a view and apply the background.
 */
protected void setMenuBackground(){

    Log.d(TAG, "Entering setMenuBackGround");
    getLayoutInflater().setFactory( new Factory() {

        @Override
        public View onCreateView ( String name, Context context, AttributeSet attrs ) {

            if ( name.equalsIgnoreCase( "com.android.internal.view.menu.IconMenuItemView" ) ) {

                try { // Ask our inflater to create the view
                    LayoutInflater f = getLayoutInflater();
                    final View view = f.createView( name, null, attrs );
                    /*
                     * The background gets refreshed each time a new item is added the options menu.
                     * So each time Android applies the default background we need to set our own
                     * background. This is done using a thread giving the background change as runnable
                     * object
                     */
                    new Handler().post( new Runnable() {
                        public void run () {
                            view.setBackgroundResource( R.drawable.background);
                        }
                    } );
                }
            }
        }
    });
}
```



```
return view;
}
catch ( InflateException e ) {}
catch ( ClassNotFoundException e ) {}
}
return null;
}
});
}
}
```

4.5.4、触发menu

只需根据实际情况在点击按钮映射中添加如下语句:

```
this.openOptionsMenu();
或
openOptionsMenu();
```

4.5.5、Context Menu和Options Menu菜单的区别

Context Menu – 显示一个 Activity 中特定 View 的信息。在 Android 中，通过按下并 Hold 一段时间来激活上下文菜单。

Options Menu – 显示当前 Activity 的信息。在 Android 中，通过按下 MENU 键来激活选项菜单。

Options Menu 需要重写两个方法——onCreateOptionsMenu()和 onOptionsItemSelected()。onCreateOptionsMenu()方法在 MENU 按钮被按下时调用。当一个菜单项被选中时，onOptionsItemSelected()方法会被调用。Context Menu 需要重写 onCreateContextMenu()和 onContextItemSelected()方法。在创建 ContextMenu 是调用 onCreateContextMenu(), 当选项被选中时调用 onContextItemSelected()。

4.5.6、Context menus for expandable lists

By steveoliverc

An expandable list supports context menus in pretty much the same way that a standard list does: add a listener for the context menu (when the user has long-pressed on a list item). Unlike a standard list view, however, you probably want to know whether the user has selected a group (expandable item) or a child (sub-item) list item.

Furthermore, you might not want to do anything if the user tries to bring up a context menu on a group item. There might be cases where you would want to do something to all of the children under a group, but in my

Librarium application, I wanted to ignore group items and present the context menu only for children.

First, you need to know when the context menu is going to be created so that you can identify whether the user pressed on a group or a child. If they pressed on a group, then cancel the context menu. This also gives us a chance to get the text of the child item, so that we can put it into the header of the context menu.

```
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo menuInfo) {

    super.onCreateContextMenu(menu, v, menuInfo);
    ExpandableListView.ExpandableListContextMenuInfo info =
        (ExpandableListView.ExpandableListContextMenuInfo) menuInfo;
    int type =
        ExpandableListView.getPackedPositionType(info.packedPosition);
    int group =
        ExpandableListView.getPackedPositionGroup(info.packedPosition);
    int child =
        ExpandableListView.getPackedPositionChild(info.packedPosition);
    //Only create a context menu for child items
    if (type == 1) {
        //Array created earlier when we built the expandable list
        String page = mListStringArray[group][child];
        menu.setHeaderTitle(page);
        menu.add(0, MENU_READ, 0, "Read page");
        menu.add(0, MENU_EDIT, 0, "Edit page");
        menu.add(0, MENU_FAVORITE, 0, "Add page to favorites");
        menu.add(0, MENU_EXPORT, 0, "Export page to file");
        menu.add(0, MENU_DELETE, 1, "Delete page");
    }
}
```

Second, create the context menu:

```
public boolean onContextItemSelected(MenuItem menuItem) {
    ExpandableListContextMenuInfo info =
        (ExpandableListContextMenuInfo) menuItem getMenuInfo();
    int groupPos = 0, childPos = 0;
    int type = ExpandableListView.getPackedPositionType(info.packedPosition);
    if (type == ExpandableListView.PACKED_POSITION_TYPE_CHILD) {
        groupPos = ExpandableListView.getPackedPositionGroup(info.packedPosition);
        childPos = ExpandableListView.getPackedPositionChild(info.packedPosition);
    }
    //Pull values from the array we built when we created the list
```

```
String author = mListStringArray[groupPos][0];
String page = mListStringArray[groupPos][childPos * 3 + 1];
rowId = Integer.parseInt(mListStringArray[groupPos][childPos * 3 + 3]);
switch (menuItem.getItemId()) {
case MENU_READ:
readNote(rowId);
return true;
case MENU_EDIT:
editNote(rowId);
return true;
//etc....
default:
return super.onContextItemSelected(menuItem);
}
}
```

That's it. Now users can long-press on an item in an expandable list, and get the context menu if it's a child item.

4.6、ListView

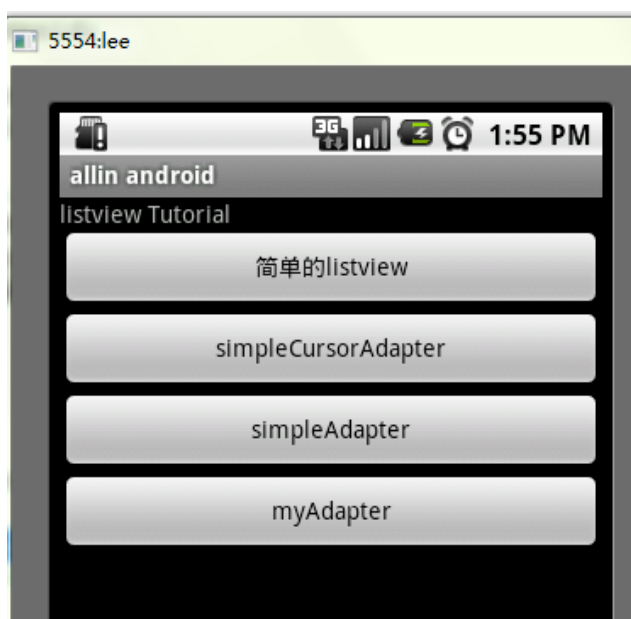
4.6.1、ListView自定义分割线

```
int[] colors = {0, 0xFFCCCC, 0}; // red for the example
listview.setDivider(new GradientDrawable(Orientation.RIGHT_LEFT, colors));
```

4.6.2、LIST例一

在 android 开发中 ListView 是比较常用的组件，它以列表的形式展示具体内容，并且能够根据数据的长度自适应显示。抽空把对 ListView 的使用做了整理，并写了个小例子，如下图。





列表的显示需要三个元素：

1. ListView 用来展示列表的 View。
2. 适配器 用来把数据映射到 ListView 上的中介。
3. 数据 具体的将被映射的字符串，图片，或者基本组件。

根据列表的适配器类型，列表分为三种，ArrayAdapter，SimpleAdapter 和 SimpleCursorAdapter

其中以 ArrayAdapter 最为简单，只能展示一行字。SimpleAdapter 有最好的扩充性，可以自定义出各种效果。SimpleCursorAdapter 可以认为是 SimpleAdapter 对数据库的简单结合，可以方便的把数据库的内容以列表的形式展示出来。

我们从最简单的 ListView 开始：

```
/**
 * @author allin
 *
 */
public class MyListView extends Activity {
    private ListView listView;
    //private List<String> data = new ArrayList<String>();
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        listView = new ListView(this);
        listView.setAdapter(new ArrayAdapter<String>(this,
        android.R.layout.simple_expandable_list_item_1,getData()));
        setContentView(listView);
    }
    private List<String> getData(){
        List<String> data = new ArrayList<String>();
        data.add("测试数据 1");
    }
}
```

```

data.add("测试数据 2");
data.add("测试数据 3");
data.add("测试数据 4");
return data;
}
}

```

复制代码

上面代码使用了 `ArrayAdapter(Context context, int textViewResourceId, List<T> objects)` 来装配数据，要装配这些数据就需要一个连接 `ListView` 视图对象和数组数据的适配器来两者的适配工作，`ArrayAdapter` 的构造需要三个参数，依次为 `this`, 布局文件（注意这里的布局文件描述的是列表的每一行的布局，`android.R.layout.simple_list_item_1` 是系统定义好的布局文件只显示一行文字，数据源（一个 `List` 集合）。同时用 `setAdapter()` 完成适配的最后工作。运行后的现实结构如下图：



SimpleCursorAdapter

sdk 的解释是这样的：An easy adapter to map columns from a cursor to TextViews or ImageViews defined in an XML file. You can specify which columns you want, which views you want to display the columns, and the XML file that defines the appearance of these views。简单的说就是方便把从游标得到的数据进行列表显示，并可以把指定的列映射到对应的 `TextView` 中。

下面的程序是从电话簿中把联系人显示到类表中。先在通讯录中添加一个联系人作为数据库的数据。然后获得一个指向数据库的 `Cursor` 并且定义一个布局文件（当然也可以使用系统自带的）。

```

/**
 * @author allin
 *
 */
public class MyListView2 extends Activity {
    private ListView listView;
    //private List<String> data = new ArrayList<String>();
}

```



```

@Override
public void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    listView = new ListView(this);
    Cursor cursor = getContentResolver().query(People.CONTENT_URI, null, null, null, null);
    startManagingCursor(cursor);
    ListAdapter listAdapter = new SimpleCursorAdapter(this,
    android.R.layout.simple_expandable_list_item_1,
    cursor,
    new String[]{People.NAME},
    new int[]{android.R.id.text1});
    listView.setAdapter(listAdapter);
    setContentView(listView);
}
}

```

复制代码

Cursor cursor = getContentResolver().query(People.CONTENT_URI, null, null, null, null);先获得一个指向系统通讯录数据库的 Cursor 对象获得数据来源。

startManagingCursor(cursor);我们将获得的 Cursor 对象交由 Activity 管理，这样 Cursor 的生命周期和 Activity 便能够自动同步，省去自己手动管理 Cursor。

SimpleCursorAdapter 构造函数前面 3 个参数和 ArrayAdapter 是一样的，最后两个参数：一个包含数据库的列的 String 型数组，一个包含布局文件中对应组件 id 的 int 型数组。其作用是自动的将 String 型数组所表示的每一列数据映射到布局文件对应 id 的组件上。上面的代码，将 NAME 列的数据一次映射到布局文件的 id 为 text1 的组件上。

注意：需要在 AndroidManifest.xml 中如权限：<uses-permission android:name="android.permission.READ_CONTACTS"></uses-permission>

运行后效果如下图：



SimpleAdapter

simpleAdapter 的扩展性最好，可以定义各种各样的布局出来，可以放上 ImageView（图片），还可以放上 Button（按钮），CheckBox（复选框）等等。下面的代码都直接继承了 ListActivity，ListActivity 和普通的 Activity 没有太大的差别，不同就是对显示 ListView 做了许多优化，方

面显示而已。

下面的程序是实现一个带有图片的类表。

首先需要定义好一个用来显示每一个列内容的 xml

vlist.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <ImageView android:id="@+id/img"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="5px"/>
    <LinearLayout android:orientation="vertical"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <TextView android:id="@+id/title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="#FFFFFF"
            android:textSize="22px" />
        <TextView android:id="@+id/info"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="#FFFFFF"
            android:textSize="13px" />
    </LinearLayout>
</LinearLayout>
```

复制代码

下面是实现代码：

```
/**
 * @author allin
 *
 */
public class MyListView3 extends ListActivity {
    // private List<String> data = new ArrayList<String>();
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        SimpleAdapter adapter = new SimpleAdapter(this,getData(),R.layout.vlist,
            new String[]{"title","info","img"},
            new int[]{R.id.title,R.id.info,R.id.img});
        setListAdapter(adapter);
    }
    private List<Map<String, Object>> getData() {
```



```

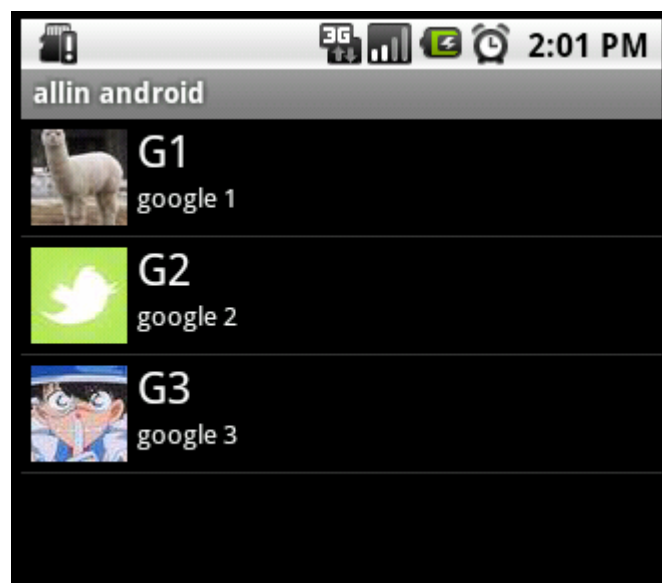
List<Map<String, Object>> list = new ArrayList<Map<String, Object>>();
Map<String, Object> map = new HashMap<String, Object>();
map.put("title", "G1");
map.put("info", "google 1");
map.put("img", R.drawable.i1);
list.add(map);
map = new HashMap<String, Object>();
map.put("title", "G2");
map.put("info", "google 2");
map.put("img", R.drawable.i2);
list.add(map);
map = new HashMap<String, Object>();
map.put("title", "G3");
map.put("info", "google 3");
map.put("img", R.drawable.i3);
list.add(map);
return list;
}
}

```

复制代码

使用 simpleAdapter 的数据用一般都是 HashMap 构成的 List，list 的每一节对应 ListView 的每一行。HashMap 的每个键值数据映射到布局文件中对应 id 的组件上。因为系统没有对应的布局文件可用，我们可以自己定义一个布局 vlist.xml。下面做适配，new 一个 SimpleAdapter 参数一次是：this，布局文件（vlist.xml），HashMap 的 title 和 info，img。布局文件的组件 id，title，info，img。布局文件的各组件分别映射到 HashMap 的各元素上，完成适配。

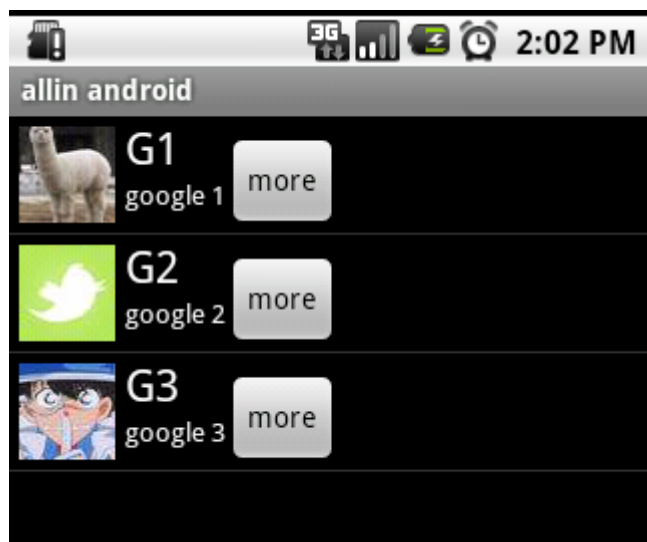
运行效果如下图：



有按钮的 ListView

但是有时候，列表不光会用来做显示用，我们同样可以在上面添加按钮。添加按钮

首先要写一个有按钮的 xml 文件，然后自然会想到用上面的方法定义一个适配器，然后将数据映射到布局文件上。但是事实并非这样，因为按钮是无法映射的，即使你成功的用布局文件显示出了按钮也无法添加按钮的响应，这时就要研究一下 `ListView` 是如何现实的了，而且必须要重写一个类继承 `BaseAdapter`。下面的示例将显示一个按钮和一个图片，两行字如果单击按钮将删除此按钮的所在行。并告诉你 `ListView` 究竟是如何工作的。效果如下：



vlist2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <ImageView android:id="@+id/img"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="5px"/>
    <LinearLayout android:orientation="vertical"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <TextView android:id="@+id/title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="#FFFFFF"
            android:textSize="22px" />
        <TextView android:id="@+id/info"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="#FFFFFF"
            android:textSize="13px" />
    </LinearLayout>
    <Button android:id="@+id/view_btn"
```




```

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/s_view_btn"
        android:layout_gravity="bottom|right" />
</LinearLayout>

```

复制代码

程序代码:

```

/**
 * @author allin
 *
 */
public class MyListView4 extends ListActivity {
    private List<Map<String, Object>> mData;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mData = getData();
        MyAdapter adapter = new MyAdapter(this);
        setListAdapter(adapter);
    }
    private List<Map<String, Object>> getData() {
        List<Map<String, Object>> list = new ArrayList<Map<String, Object>>();
        Map<String, Object> map = new HashMap<String, Object>();
        map.put("title", "G1");
        map.put("info", "google 1");
        map.put("img", R.drawable.i1);
        list.add(map);
        map = new HashMap<String, Object>();
        map.put("title", "G2");
        map.put("info", "google 2");
        map.put("img", R.drawable.i2);
        list.add(map);
        map = new HashMap<String, Object>();
        map.put("title", "G3");
        map.put("info", "google 3");
        map.put("img", R.drawable.i3);
        list.add(map);
        return list;
    }
    // ListView 中某项被选中后的逻辑
    @Override
    protected void onItemClick(ListView l, View v, int position, long id) {
        Log.v("MyListView4-click", (String)mData.get(position).get("title"));
    }
}

```



```
/**
 * listview 中点击按钮弹出对话框
 */
public void showInfo(){
    new AlertDialog.Builder(this)
        .setTitle("我的 listview")
        .setMessage("介绍...")
        .setPositiveButton("确定", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
            }
        })
        .show();
}
public final class ViewHolder{
    public ImageView img;
    public TextView title;
    public TextView info;
    public Button viewBtn;
}
public class MyAdapter extends BaseAdapter{
    private LayoutInflater mInflater;
    public MyAdapter(Context context){
        this.mInflater = LayoutInflater.from(context);
    }
    @Override
    public int getCount() {
        // TODO Auto-generated method stub
        return mData.size();
    }
    @Override
    public Object getItem(int arg0) {
        // TODO Auto-generated method stub
        return null;
    }
    @Override
    public long getItemId(int arg0) {
        // TODO Auto-generated method stub
        return 0;
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        ViewHolder holder = null;
        if (convertView == null) {
```



```

holder=new ViewHolder();
convertView = mInflater.inflate(R.layout.vlist2, null);
holder.img = (ImageView)convertView.findViewById(R.id.img);
holder.title = (TextView)convertView.findViewById(R.id.title);
holder.info = (TextView)convertView.findViewById(R.id.info);
holder.viewBtn = (Button)convertView.findViewById(R.id.view_btn);
convertView.setTag(holder);
}else {
    holder = (ViewHolder)convertView.getTag();
}
holder.img.setBackgroundResource((Integer)mData.get(position).get("img"));
holder.title.setText((String)mData.get(position).get("title"));
holder.info.setText((String)mData.get(position).get("info"));
holder.viewBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        showInfo();
    }
});
return convertView;
}
}
}

```

复制代码

下面将对上述代码,做详细的解释, listView 在开始绘制的时候,系统首先调用 getCount () 函数,根据他的返回值得到 listView 的长度(这也是为什么在开始的第一张图特别的标出列表长度),然后根据这个长度,调用 getView () 逐一绘制每一行。如果你的 getCount () 返回值是 0 的话,列表将不显示同样 return 1, 就只显示一行。

系统显示列表时,首先实例化一个适配器(这里将实例化自定义的适配器)。当手动完成适配时,必须手动映射数据,这需要重写 getView () 方法。系统在绘制列表的每一行的时候将调用此方法。getView()有三个参数, position 表示将显示的是第几行, convertView 是从布局文件中 inflate 来的布局。我们用 LayoutInflater 的方法将定义好的 vlist2.xml 文件提取成 View 实例用来显示。然后将 xml 文件中的各个组件实例化(简单的 findViewById()方法)。这样便可以将数据对应到各个组件上了。但是按钮为了响应点击事件,需要为它添加点击监听器,这样就能捕获点击事件。至此一个自定义的 listView 就完成了,现在让我们回过头从新审视这个过程。系统要绘制 ListView 了,他首先获得要绘制的这个列表的长度,然后开始绘制第一行,怎么绘制呢?调用 getView()函数。在这个函数里面首先获得一个 View (实际上是一个 ViewGroup),然后再实例并设置各个组件,显示之。好了,绘制完这一行了。那再绘制下一行,直到绘完为止。在实际的运行过程中会发现 listView 的每一行没有焦点了,这是因为 Button 抢夺了 listView 的焦点,只要布局文件中将 Button 设置为没有焦点就 OK 了。

运行效果如下图:





4.6.3、LIST例二

ListView 是 android 开发中最常用的组件之一，它通过一个 adapter 来构建显示通常有三种 adapter 可以使用 ArrayAdapter, SimpleAdapter, CursorAdapter。CursorAdapter 主要正对数据库使用，下面通过例子介绍 ArrayAdapter, SimpleAdapter 的简单使用：

1: ArrayAdapter 它接受一个数组或者 List 作为参数来构建。

一下通过简单例子说明：

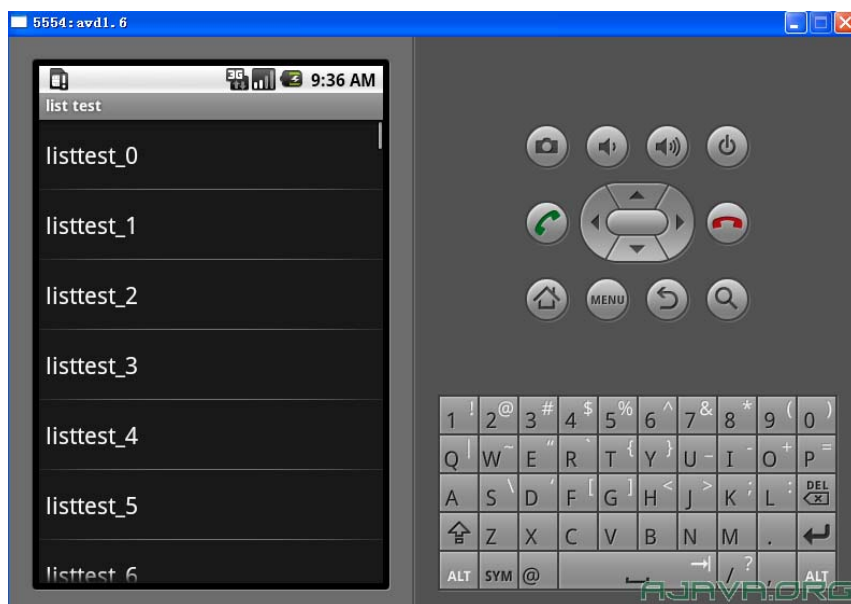
创建 Test 继承 ListActivity 这里我们传入一个 string 数组

```
public class ListTest extends ListActivity {
    /** **/ Called when the activity is first created. */

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        String[] sw = new String[100];
        for (int i = 0; i < 100; i++) {
            sw[i] = "listtest_" + i;
        }
        ArrayAdapter<String> adapter = new
        ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, sw); // 使用系统已经实现好的
        xml 文件 simple_list_item_1
        setListAdapter(adapter);
    }
}
```

运行如图：

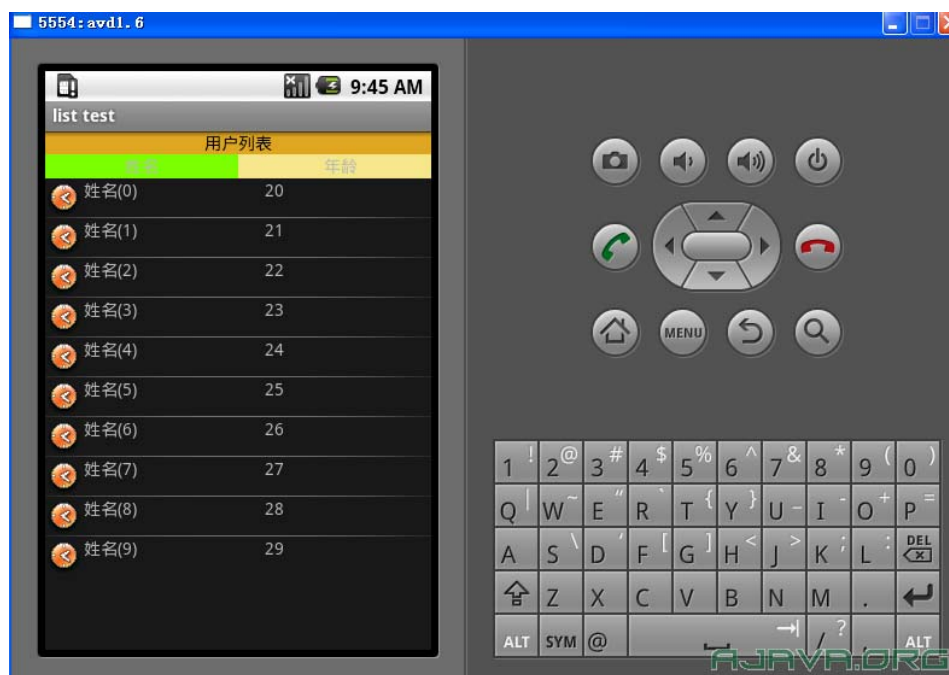




从以上代码可以看不需要加载我们自己的 layout 而是用系统已经实现的 layout 很快的实现了 listview

第二种 SimpleAdapter:

先看下我们例子的最终截图:



通过上图可以看出 listview 每行不仅仅是一个 string 包括了很多项, 图片, 多项文字 我们通过构建 list, 并设置每项为一个 map 来实现:

代码: 创建 TestList 类继承 Activity

```

super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    ArrayList<HashMap<String, Object>> users = new ArrayList<HashMap<String,
Object>>>();
    for (int i = 0; i < 10; i++) {
        HashMap<String, Object> user = new HashMap<String, Object>();
        user.put("img", R.drawable.user);
        user.put("username", "姓名(" + i + ")");
        user.put("age", (20 + i) + "");
        users.add(user);
    }
    SimpleAdapter saImageItems = new SimpleAdapter(this,
        users, // 数据来源
        R.layout.user, // 每一个 user xml 相当 ListView 的一个组件
        new String[] { "img", "username", "age" },
        // 分别对应 view 的 id
        new int[] { R.id.img, R.id.name, R.id.age });
    // 获取 listview
    ((ListView) findViewById(R.id.users)).setAdapter(saImageItems);

```

下面是 main.xml 的内容:

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView android:text="用户列表" android:gravity="center"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" android:background="#DAA520"
        android:textColor="#000000">
    </TextView>
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <TextView android:text="姓名"
            android:gravity="center" android:layout_width="160px"
            android:layout_height="wrap_content" android:textStyle="bold"
            android:background="#7CFC00">
        </TextView>
        <TextView android:text="年龄"
            android:layout_width="170px" android:gravity="center"
            android:layout_height="wrap_content" android:textStyle="bold"
            android:background="#F0E68C">
        </TextView>
    </LinearLayout>

```



```

</LinearLayout>
<ListView android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:id="@+id/users">
    </ListView>
</LinearLayout>

```

之中 listView 前面的可以说是标题行，listview 相当于用来显示数据的容器，里面每行是一个用户信息，而用户信息是样子呢？

看看 use.xml

```

<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    android:layout_width="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="wrap_content"
    >
    <TableRow >
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/img">
    </ImageView>
    <TextView
        android:layout_height="wrap_content"
        android:layout_width="150px"
        android:id="@+id/name">
    </TextView>
    <TextView
        android:layout_height="wrap_content"
        android:layout_width="170px"
        android:id="@+id/age">
    </TextView>
    </TableRow>
</TableLayout>

```

也就是说每行包含了一个 img 和 2 个文字信息
这个文件以参数的形式通过 adapter 在 listview 中显示。
也就是：

```

SimpleAdapter saImageItems = new SimpleAdapter(this,
    users,// 数据来源
    R.layout.user,//每一个 user xml 相当 ListView 的一个组件
    new String[] { "img", "username", "age" },

    // 分别对应 view 的 id

```

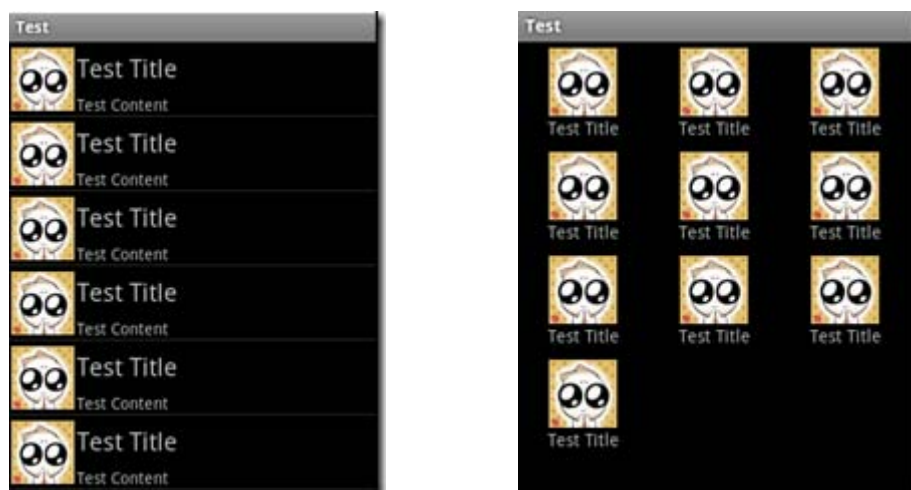


```
new int[] { R.id.img, R.id.name, R.id.age }));
```

4.6.4、LIST例三

简约而不简单——Android SimpleAdapter

列表(ListView)、表格(Gridview)，这在手机应用上面肯定是少不了的，怎样实现比较复杂一点的界面呢，先看一下我的效果图。



这样布局的情况是最基本的，也是最常用的，网上关于这样的布局有多种版本的实现方法，但是有很多需要自己实现 Adapter，那样子是比较复杂而且没有必要的，因为我们有简约而不简单的 SimpleAdapter。

1. ListView

SimpleAdapter 的核心代码：

```
for (int i = 0; i < 10; i++) {
    Map<String, Object> map = new HashMap<String, Object>();
    map.put("PIC", R.drawable.pic);
    map.put("TITLE", "Test Title");
    map.put("CONTENT", "Test Content");
    contents.add(map);
}
SimpleAdapter adapter = new SimpleAdapter(this,
    (List<Map<String, Object>>) contents, R.layout.listitem,
    new String[] { "PIC", "TITLE", "CONTENT" }, new int[] {
        R.id.listitem_pic, R.id.listitem_title,
        R.id.listitem_content });

listView.setAdapter(adapter);
```



listitem 的 Layout:

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

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="?android:attr/listPreferredItemHeight">
    <ImageView android:id="@+id/listitem_pic"
        android:layout_width="wrap_content" android:layout_height="fill_parent"
        android:layout_alignParentTop="true" android:layout_alignParentBottom="true"
        android:src="@drawable/pic" android:adjustViewBounds="true"
        android:padding="2dip" />
    <TextView android:id="@+id/listitem_title"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_toRightOf="@+id/listitem_pic"
        android:layout_alignParentRight="true" android:layout_alignParentTop="true"
        android:layout_above="@+id/listitem_content"
        android:layout_alignWithParentIfMissing="true" android:gravity="center_vertical"
        android:text="@+id/listitem_title" android:textSize="22px" />
    <TextView android:id="@+id/listitem_content"
        android:layout_width="fill_parent" android:layout_height="wrap_content"
        android:layout_toRightOf="@+id/listitem_pic"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true" android:singleLine="true"
        android:ellipsize="marquee" android:text="@+id/item_content"
        android:textSize="14px" />
</RelativeLayout>
```

2. GridView

SimpleAdapter 的核心代码:

```
for (int i = 0; i < 10; i++) {
    Map<String, Object> map = new HashMap<String, Object>();
    map.put("PIC", R.drawable.pic);
    map.put("TITLE", "Test Title");
    contents.add(map);
}
SimpleAdapter adapter = new SimpleAdapter(this,
    (List<Map<String, Object>>) contents, R.layout.griditem,
    new String[] { "PIC", "TITLE" }, new int[] { R.id.griditem_pic,
        R.id.griditem_title, });
```



```
gridView.setAdapter(adapter);
```

griditem 的 Layout:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="fill_parent" android:layout_width="fill_parent"
    android:orientation="vertical">
    <ImageView android:id="@+id/griditem_pic"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal">
    </ImageView>
    <TextView android:id="@+id/griditem_title"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal" android:text="test">
    </TextView>
</LinearLayout>
```

4.6.5、ListView 被选中item的背景颜色

这用到了 Android 的 Selector（根据组件的状态显示该状态对应的图片做为显示背景）。

把下面的 XML 文件保存成你自己命名的.xml 文件（比如 list_bg.xml），注意，这个文件相当于一个背景图片选择器，在系统使用时根据 ListView 中的列表项的状态来使用相应的背景图片，什么情况使用什么图片我在下面都进行了说明。还有，你可以把它看成是一个图片来使用，放于 drawable 目录下，配置背景属性 android:background="@drawable/list_bg" 就能达到你需要的目的了。

```
<?xml version="1.0" encoding="utf-8" ?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_window_focused="false"
        android:drawable="@drawable/没有焦点时的图片背景" />
    <item android:state_focused="true" android:state_pressed="true"
        android:drawable=
            "@drawable/非触摸模式下获得焦点并单击时的背景图片" />
    <item android:state_focused="false" android:state_pressed="true"
        android:drawable="@drawable/触摸模式下单击时的背景图片" />
    <item android:state_selected="true"
        android:drawable="@drawable/选中时的图片背景" />
    <item android:state_focused="true"
        android:drawable="@drawable/获得焦点时的图片背景" />
</selector>
```

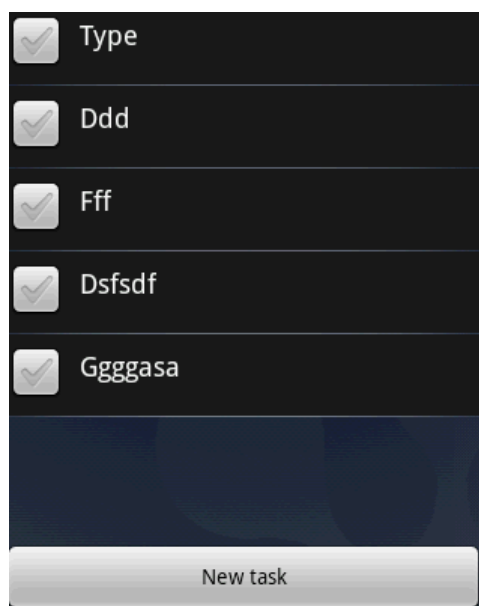


你可以看下源代码 ListView 列表项背景的默认实现

SDK 目录\tools\lib\res\default\drawable\list_selector_background.xml,我也是根据他来改自己需要的效果。希望对你有用

4.6.6、ListView 自定义背景颜色

在 Android 中, ListView 是最常用的一个控件, 在做 UI 设计的时候, 很多人希望能够改变一下它的背景, 使他能够符合整体的 UI 设计, 改变背景很简单只需要准备一张图片然后指定属性 `android:background="@drawable/bg"`, 不过不要高兴地太早, 当你这么做以后, 发现背景是变了, 但是当你拖动, 或者点击 list 空白位置的时候发现 ListItem 都变成黑色的了, 破坏了整体效果, 如下图所示:



这是为什么呢?

这个要从 Listview 的效果说起, 默认的 ListItem 背景是透明的, 而 ListView 的背景是固定不变的, 所以在滚动条滚动的过程中如果实时地去将当前每个 Item 的显示内容跟背景进行混合运算, 所以 android 系统为了优化这个过程用, 就使用了一个叫做 `android:cacheColorHint` 的属性, 在黑色主题下默认的颜色值是 #191919, 所以就出现了刚才的画面, 有一半是黑色的

那怎么办呢?

如果你只是换背景的颜色, 可以直接指定 `android:cacheColorHint` 为你所要的颜色, 如果你是用图片做背景的话, 那也只要将 `android:cacheColorHint` 指定为透明 (#00000000) 就可以了, 当然为了美化是要牺牲一些效率的。最后美化的效果如图:





4.6.7、List长按与短按消息映射

ListActivity 和 ListView 是很常用的组件，用来制作列表形式的用户界面。本文介绍如何正确处理 ListView 中的条目短按和长按事件，他们的处理方式是不同的。

对于短按事件，处理起来比较简单，我们只需要覆盖 ListActivity 的 onItemClick() 方法，如下所示：

```
@Override
protected void onItemClick(ListView arg0, View arg1, int arg2, long arg3) {
    CharSequence s = ((TextView)arg1).getText();
    Log.e("CallLogActivity", s + " is clicked");
    super.onItemClick(arg0, arg1, arg2, arg3);
}
```

对于长按事件，我们需要给 listview 注册一个 OnItemLongClickListener，并实现 Listener 中定义的方法，如下所示：

```
getListView().setOnItemLongClickListener(this); //注册

public boolean onItemLongClick(AdapterView parent, View view, int position,
    long id) {
    Log.e("CallLogActivity", view.toString() + "position=" + position);
    CharSequence number = ((TextView) view).getText();
    Toast t = Toast.makeText(this, number + " is long clicked",
        Toast.LENGTH_LONG);
    t.show();
}
```



```

    return true;
}

```

我们使用前面介绍的 CallLog 的演示例子来演示一下如何实现这两种事件，完整的代码如下所示：

```

import android.app.ListActivity;
import android.database.Cursor;
import android.os.Bundle;
import android.provider.CallLog;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.CursorAdapter;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;
import android.widget.TextView;
import android.widget.Toast;

public class CallLogActivity extends ListActivity implements
    OnItemClickListener {

    @Override
    protected void onCreate(Bundle arg0) {
        super.onCreate(arg0);
        setContentView(R.layout.main);
        Cursor cursor = getContentResolver().query(CallLog.Calls.CONTENT_URI,
            null, null, null, CallLog.Calls.DEFAULT_SORT_ORDER);
        startManagingCursor(cursor);
        SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
            android.R.layout.simple_list_item_1, cursor,
            new String[] { CallLog.Calls.NUMBER },
            new int[] { android.R.id.text1 });
        getListView().setOnItemClickListener(this);
        setListAdapter(adapter);
    }

    public boolean onItemClick(AdapterView parent, View view, int position,
        long id) {
        Log.e("CallLogActivity", view.toString() + "position=" + position);
        CharSequence number = ((TextView) view).getText();
        Toast t = Toast.makeText(this, number + " is long clicked",
            Toast.LENGTH_LONG);
        t.show();
    }
}

```



```

    return true;
}

@Override
protected void onListItemClick(ListView arg0, View arg1, int arg2, long arg3) {
    CharSequence s = ((TextView)arg1).getText();
    Log.e("CallLogActivity", s + " is clicked");
    super.onListItemClick(arg0, arg1, arg2, arg3);
}

```

21. ListView 点击与长按消息

```

//添加并且显示 存在缺点，长按时也会执行点击时的代码？？
list.setAdapter(listItemAdapter);

//添加点击
list.setOnItemClickListener(new OnItemClickListener() {

    @Override
    public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
        long arg3) {
        setTitle("点击第"+arg2+"个项目");
    }
});

//添加长按点击
list.setOnCreateContextMenuListener(new OnCreateContextMenuListener() {

    @Override
    public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo menuInfo) {
        menu.setHeaderTitle("长按菜单-ContextMenu");
        menu.add(0, 0, 0, "弹出长按菜单 0");
        menu.add(0, 1, 0, "弹出长按菜单 1");
    }
});

//长按菜单响应函数
@Override
public boolean onContextItemSelected(Menu.Item item) {
    setTitle("点击了长按菜单里面的第"+item.getItemId()+"个项目");
    return super.onContextItemSelected(item);
}

```

22. WebView

1、添加权限：AndroidManifest.xml 中必须使用许可"android.permission.INTERNET",否则会出 Web page not available 错误。



2、在要 Activity 中生成一个 WebView 组件：WebView webView = new WebView(this);

3、设置 WebView 基本信息：

如果访问的页面中有 Javascript，则 webview 必须设置支持 Javascript。

```
webView.getSettings().setJavaScriptEnabled(true);
```

触摸焦点起作用

```
requestFocus();
```

取消滚动条

```
this.setScrollBarStyle(SCROLLBARS_OUTSIDE_OVERLAY);
```

4、设置 WevView 要显示的网页：

互联网用：webView.loadUrl("http://www.google.com");

本地文件用：webView.loadUrl("file:///android_asset/XX.html"); 本地文件存放在：assets 文件中

5、如果希望点击链接由自己处理，而不是新开 Android 的系统 browser 中响应该链接。

给 WebView 添加一个事件监听对象（WebViewClient）

并重写其中的一些方法

shouldOverrideUrlLoading：对网页中超链接按钮的响应。

当按下某个连接时 WebViewClient 会调用这个方法，并传递

参数：按下的 url

onLoadResource

onPageStart

onPageFinish

onReceiveError

onReceivedHttpAuthRequest

6、如果用 webview 点链接看了很多页以后，如果不做任何处理，点击系统“Back”键，整个浏览器会调用 finish()而结束自身，如果希望浏览的网页回退而不是退出浏览器，需要在当前 Activity 中处理并消费掉该 Back 事件。

覆盖 Activity 类的 onKeyDown(int keyCode,KeyEvent event)方法。

4.6.8、点击ListView改变背景色

改变整个 ListView 很简单，直接取得 ListView 组件进行设置背景就行了

如果你想改变每个 Item 的点击改变背景，建议你研究一下 Theme，默认的 Theme 在 sdk/tools/lib/res/default/value/styles.xml 中，代码如下：

```
<style name="Widget.ListView" parent="Widget.AbsListView">
```

```
    <item
```

```
name="android:listSelector">@android:drawable/list_selector_background</item>
```

<!--选择每一项的时候，出现的背景颜色，默认的是橘黄色的图片，在

sdk/tools/lib/res/default/drawable 可以找到这张图片-->



```

        <item name="android:cacheColorHint">?android:attr/colorBackground</item>
        <item
name="android:divider">@android:drawable/divider_horizontal_dark</item>
    </style>

```

应用这个样式的代码在 Themes.xml 中

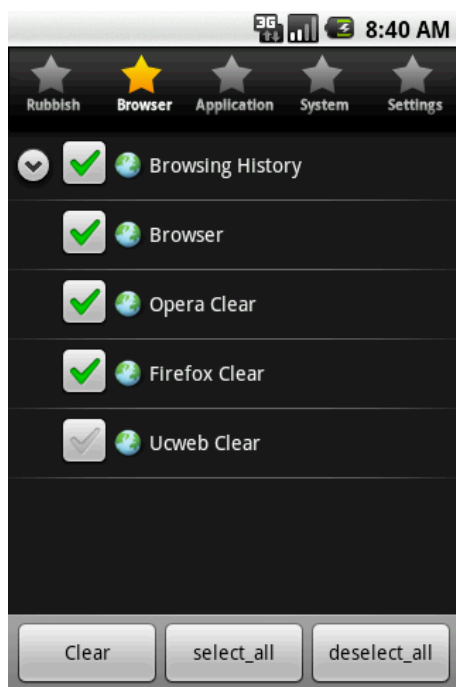
4.6.9、自动滚动ListView

```

<ListView android:id="@android:id/list"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:stackFromBottom="true"
android:transcriptMode="alwaysScroll"
/>

```

4.6.10、BaseExpandableListAdapter例



Java 代码:

```

import java.util.List;

import Android.content.Context;
import android.graphics.drawable.Drawable;
import android.view.LayoutInflater;

```



```

import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseExpandableListAdapter;
import android.widget.CheckBox;
import android.widget.ImageView;
import android.widget.TextView;
import com.iwidsets.clear.manager.R;
import com.iwidsets.clear.manager.adapter.BrowserInfo;

public class ClearExpandableListAdapter extends BaseExpandableListAdapter {

    class ExpandableListHolder {
        ImageView appIcon;
        TextView appInfo;
        CheckBox appCheckBox;
    }

    private Context context;
    private LayoutInflater mChildInflater;
    private LayoutInflater mGroupInflater;
    private List<GroupInfo> group;

    public ClearExpandableListAdapter(Context c, List<GroupInfo> group) {
        this.context = c;
        mChildInflater = (LayoutInflater) context
            .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        mGroupInflater = (LayoutInflater) context
            .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        this.group = group;
    }

    public Object getChild(int childPosition, int itemPosition) {
        return group.get(childPosition).getChild(itemPosition);
    }

    public long getChildId(int childPosition, int itemPosition) {

        return itemPosition;
    }

    @Override
    public int getChildrenCount(int index) {
        return group.get(index).getChildSize();
    }

```



```
public Object getGroup(int index) {
    return group.get(index);
}

public int getGroupCount() {
    return group.size();
}

public long getGroupId(int index) {
    return index;
}

public View getGroupView(int position, boolean flag, View view,
    ViewGroup parent) {
    ExpandableListHolder holder = null;
    if (view == null) {
        view = mGroupInflater.inflate(
            R.layout.browser_expandable_list_item, null);
        holder = new ExpandableListHolder();
        holder.appIcon = (ImageView) view.findViewById(R.id.app_icon);
        view.setTag(holder);
        holder.appInfo = (TextView) view.findViewById(R.id.app_info);
        holder.appCheckBox = (CheckBox) view
            .findViewById(R.id.app_checkbox);
    } else {
        holder = (ExpandableListHolder) view.getTag();
    }
    GroupInfo info = this.group.get(position);
    if (info != null) {
        holder.appInfo.setText(info.getBrowserInfo().getAppInfoId());
        Drawable draw = this.context.getResources().getDrawable(
            info.getBrowserInfo().getImageId());
        holder.appIcon.setImageDrawable(draw);
        holder.appCheckBox.setChecked(info.getBrowserInfo().isChecked());
    }
    return view;
}

public boolean hasStableIds() {
    return false;
}

public boolean isChildSelectable(int arg0, int arg1) {
```



```

        return false;
    }

    @Override
    public View getChildView(int groupPosition, int childPosition,
        boolean isLastChild, View convertView, ViewGroup parent) {
        ExpandableListHolder holder = null;
        if (convertView == null) {
            convertView = mChildInflater.inflate(
                R.layout.browser_expandable_list_item, null);
            holder = new ExpandableListHolder();
            holder.appIcon = (ImageView) convertView
                .findViewById(R.id.app_icon);
            convertView.setTag(holder);
            holder.appInfo = (TextView) convertView.findViewById(R.id.app_info);
            holder.appCheckBox = (CheckBox) convertView
                .findViewById(R.id.app_checkbox);
        } else {
            holder = (ExpandableListHolder) convertView.getTag();
        }
        BrowserInfo info = this.group.get(groupPosition)
            .getChild(childPosition);
        if (info != null) {
            holder.appInfo.setText(info.getAppInfoId());
            Drawable draw = this.context.getResources().getDrawable(
                info.getImageId());
            holder.appIcon.setImageDrawable(draw);
            holder.appCheckBox.setChecked(info.isChecked());
        }
        return convertView;
    }
}

```

要想让 **child** 获得焦点，只在改

```

public boolean isChildSelectable(int arg0, int arg1) {
    return true;
}

```

browser_expandable_list_item.xml 用于显示每一个 ITEM 的 XML:

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:layout_height="wrap_content"

```



```

        android:orientation="horizontal" android:minHeight="40px"
        android:layout_gravity="center_vertical">
        <CheckBox android:id="@+id/app_checkbox" android:focusable="false"
            android:layout_width="wrap_content" android:layout_height="wrap_content"
            android:layout_marginLeft="35px" android:checked="true"/>
        <ImageView android:id="@+id/app_icon" android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:layout_gravity="center_vertical" />

        <TextView android:id="@+id/app_info" android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="?android:attr/textColorPrimary"
            android:paddingLeft="3px" android:layout_gravity="center_vertical" />
    </LinearLayout>

```

browserlayout.xml 用于显示整个界面

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:layout_height="fill_parent"
    android:orientation="vertical">
    <ExpandableListView android:id="@+id/appList"
        android:layout_width="fill_parent" android:layout_height="0dip"
        android:layout_weight="1" />
    <LinearLayout android:layout_height="wrap_content"
        android:layout_width="fill_parent" android:paddingLeft="4dip"
        android:paddingRight="4dip" android:paddingBottom="1dip"
        android:paddingTop="5dip" android:background="@android:drawable/bottom_bar"
        android:id="@+id/app_footer">
        <Button android:layout_weight="1" android:id="@+id/btn_export"
            android:layout_width="0dip" android:layout_height="fill_parent"
            android:text="@string/clear"></Button>
        <Button android:layout_weight="1" android:id="@+id/btn_sel_all"
            android:layout_width="0dip" android:layout_height="fill_parent"
            android:text="select_all"></Button>
        <Button android:layout_weight="1" android:id="@+id/btn_desel_all"
            android:layout_width="0dip" android:layout_height="fill_parent"
            android:text="deselect_all"></Button>
    </LinearLayout>
</LinearLayout>

```

BrowserInfo 用于提供一个项的信息

```
public class BrowserInfo {
```



```
private int appInfoId;
private int imageId;
private boolean checked;

public BrowserInfo(int appInfoId, int imageId, boolean checked) {
    this.appInfoId = appInfoId;
    this.imageId = imageId;
    this.checked = checked;
}

public boolean isChecked() {
    return checked;
}

public void setChecked(boolean checked) {
    this.checked = checked;
}

public int getAppInfoId() {
    return appInfoId;
}

public void setAppInfoId(int appInfoId) {
    this.appInfoId = appInfoId;
}

public int getImageId() {
    return imageId;
}

public void setImageId(int imageId) {
    this.imageId = imageId;
}
}
```

GroupInfo 用于显示一个组的信息

```
import java.util.List;

public class GroupInfo {

    private BrowserInfo group;
    private List<BrowserInfo> child;
```



```
public GroupInfo(BrowserInfo group, List<BrowserInfo> child) {

    this.group = group;
    this.child = child;
}

public void add(BrowserInfo info){
    child.add(info);
}

public void remove(BrowserInfo info){
    child.remove(info);
}

public void remove(int index){
    child.remove(index);
}

public int getChildSize(){
    return child.size();
}

public BrowserInfo getChild(int index){
    return child.get(index);
}

public BrowserInfo getBrowserInfo() {
    return group;
}

public void setBrowserInfo(BrowserInfo group) {
    this.group = group;
}

public List<BrowserInfo> getChild() {
    return child;
}

public void setChild(List<BrowserInfo> child) {
    this.child = child;
}
}
```



ClearBrowserActivity 最后就是把要显示的内容显示出来的。

```
public class ClearBrowserActivity extends Activity implements
ExpandableListView.OnGroupClickListener,ExpandableListView.OnChildClickListener{

    private List<GroupInfo> group;

    private ClearExpandableListAdapter listAdapter = null;
    private ExpandableListView appList;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.browserlayout);
        appList = (ExpandableListView) findViewById(R.id.appList);
        init();
        BrowserInfo browsingHistoryParents = new BrowserInfo(
            R.string.browsinghistory, R.drawable.browser_image,true);
        List<BrowserInfo> browsingHistoryChild = new ArrayList<BrowserInfo>();
        BrowserInfo browser = new BrowserInfo(R.string.browser,
            R.drawable.browser_image, true);
        browsingHistoryChild.add(browser);

        BrowserInfo operaClear = new BrowserInfo(R.string.opera_clear,
            R.drawable.browser_image,true);
        browsingHistoryChild.add(operaClear);

        BrowserInfo firefoxClear = new BrowserInfo(R.string.firefox_clear,
            R.drawable.browser_image,true);
        browsingHistoryChild.add(firefoxClear);

        BrowserInfo ucwebClear = new BrowserInfo(R.string.ucweb_clear,
            R.drawable.browser_image,false);
        browsingHistoryChild.add(ucwebClear);

        GroupInfo browserGroup = new GroupInfo(browsingHistoryParents,
            browsingHistoryChild);
        addGroup(browserGroup);

        listAdapter = new ClearExpandableListAdapter(this, group);
        appList.setOnChildClickListener(this);
        appList.setOnGroupClickListener(this);
        appList.setAdapter(listAdapter);
    }
}
```



```

private void init() {
    group = new ArrayList<GroupInfo>();
}

private void addGroup(GroupInfo info) {
    group.add(info);
}

private static BrowserInfo newBrowserInfo(int infoId, int imageId, boolean checked) {
    return new BrowserInfo(infoId, imageId, checked);
}

@Override
public boolean onGroupClick(ExpandableListView parent, View v,
    int groupPosition, long id) {
    return false;
}

@Override
public boolean onChildClick(ExpandableListView parent, View v,
    int groupPosition, int childPosition, long id) {
    return false;
}
}

```

4.6.11、列表视图（List View）

列表布局：是一个 **ViewGroup** 以列表显示它的子视图（**view**）元素，列表是可滚动的列表。列表元素通过 **ListAdapter** 自动插入到列表。

ListAdapter：扩展自 **Adapter**，它是 **ListView** 和数据列表之间的桥梁。**ListView** 可以显示任何包装在 **ListAdapter** 中的数据。该类提供两个公有类型的抽象方法：

1. **public abstract boolean areAllItemsEnabled ()** : 表示 **ListAdapter** 中的所有元素是否可激活的？如果返回真，即所有的元素是可选择的即可点击的。
2. **public abstract boolean isEnabled (int position)** : 判断指定位置的元素是否可激活的？

下面通过一个例子来，创建一个可滚动的列表，并从一个字符串数组读取列表元素。当一个元素被选择时，显示该元素在列表中的位置的消息。

1)、首先，将 **res/layout/main.xml** 的内容置为如下：




```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10dp"
    android:textSize="16sp" >
</TextView>
```

这样就定义了元素在列表中的布局。

2)、[src/skynet.com.cnblogs.www/HelloWorld.Java](http://src.skynet.com.cnblogs/www/HelloWorld.Java)文件的代码如下:

```
package skynet.com.cnblogs.www;import android.app.ListActivity;import
android.os.Bundle;import android.view.View;import android.wi
dget.AdapterView;import android.widget.ArrayAdapter;import androi
d.widget.ListView;import android.widget.TextView;import android.w
idget.Toast;import android.widget.AdapterView.OnItemClickListener;
r;
public class HelloWorld extends ListActivity {
    //注意这里 Helloworld 类不是扩展自 Acitvity, 而是扩展自 ListActi
vity /** Called when the activity is first created. */
    @Override public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setListAdapter(new ArrayAdapter<String>(this, R.layout.
main, COUNTRIES));
        ListView lv = getListView();
        lv.setTextFilterEnabled(true);
        lv.setOnItemClickListener(new OnItemClickListener() {
            public void onItemClick(AdapterView<?> parent, View view,
                int position, long id) {
                // When clicked, show a toast with the TextView text
                Toast.makeText(getApplicationContext(), ((TextView) v
iew).getText(), Toast.LENGTH_SHORT).show();
            }
        });
        static final String[] COUNTRIES = new String[] {
            "1", "2", "3", "4", "5", "6", "7", "8", "9", "10",
            "11", "12", "13", "14", "15", "16", "17", "18",
            "19", "20", "21", "22", "23", "24"};
```

Note: `onCreate()`函数中并不像往常一样通过`setContentView()`为活动（Activity）加载布局文件，替代的是通过`setListAdapter(ListAdapter)`自动添加一个`ListView`填充整

个屏幕的ListActivity。在此文件中这个方法以一个ArrayAdapter为参数：`setListAdapter(new ArrayAdapter<String>(this, R.layout.main, COUNTRES))`，这个ArrayAdapter管理填入ListView中的列表元素。ArrayAdapter的构造函数的参数为：`this`（表示应用程序的上下文context）、表示ListViewde布局文件（这里是R.layout.main）、插入ListView的List对象对数组（这里是COUNTRES）。

`setOnItemClickListener(OnItemClickListener)`定义了每个元素的点击（on-click）的监听器，当ListView中的元素被点击时，`onItemClick()`方法被调用，在这里是即一个 **Toast**消息——每个元素的位置将显示。

3)、运行应用程序得如下结果（点击1之后，在下面显示了1）：

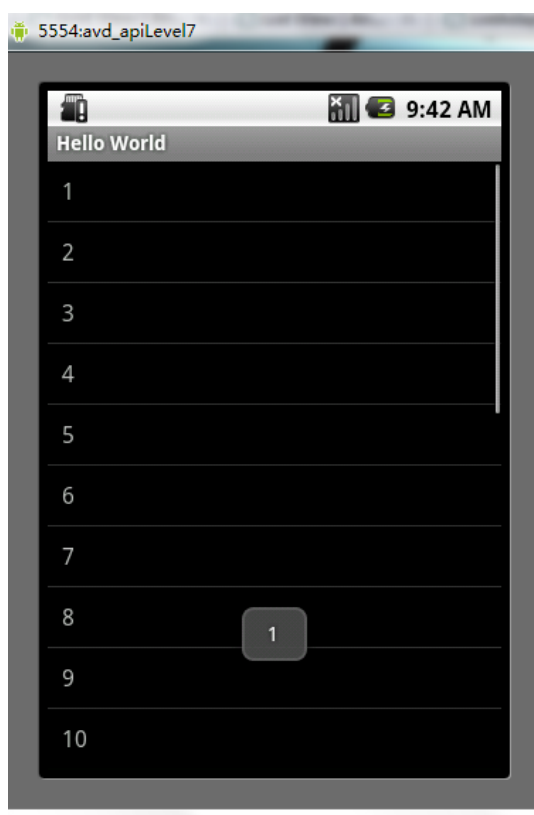


图 7、列表布局

NOTE:如果你改了 HelloWorld `extends ListActivity` 而不是 `Activity` 之后，运行程序是提示：“Conversion to Dalvik format failed with error 1”。可以这么解决：解决办法是 `Project > Clean... > Clean project selected below > Ok`

一个小的改进

上面我们是把要填充到 `ListView` 中的元素硬编码到 `HelloWorld.java` 文件中，这样就缺乏灵活性！也不符合推荐的应用程序的界面与控制它行为的代码更好地分离的准则！



其实我们可以把要填充到 **ListView** 的元素写到 **res/values/strings.xml** 文件中的 `<string-array>` 元素中，然后再源码中动态地读取。这样 **strings.xml** 的内容类似下面：

```
<?xml version="1.0" encoding="utf-8"?><resources>    <string-array
name="countries_array">        <item>1</item>        <item>2</
item>        <item>3</item>        <item>4</item>        <item>5
</item>        <item>6</item>        <item>7</item>    </string-
array></resources>
```

然而 **HelloWorld.java** 文件中的 **onCreate()** 函数，则这样动态访问这个数组及填充到 **ListVies**:

```
String[] countries = getResources().getStringArray(R.array.countries_array);
setListAdapter(new ArrayAdapter<String>(this, R.layout.list_item, countries));
```

补充说明

首先总结一下列表布局的关键部分：

- 布局文件中定义 **ListView**
- **Adapter** 用来将数据填充到 **ListView**
- 要填充到 **ListView** 的数据，这些数据可以是字符串、图片、控件等等

其中 **Adapter** 是 **ListView** 和数据源之间的桥梁，根据数据源的不同 **Adapter** 可以分为三类：

- **String[]: ArrayAdapter**
- **List<Map<String,?>>: SimpleAdapter**
- 数据库 **Cursor: SimpleCursorAdapter**

使用 **ArrayAdapter**（数组适配器）顾名思义，需要把数据放入一个数组以便显示，上面的例子就是这样的；**SimpleAdapter**能定义各种各样的布局出来，可以放上 **ImageView**（图片），还可以放上 **Button**（按钮），**CheckBox**（复选框）等等；**SimpleCursorAdapter**是和数据库有关的东西。篇幅有限后面两种就不举例实践了。你可以参考 [android ListView详解orArrayAdapter ,SimpleAdapter ,SimpleCursorAdapter 区别](#)。

4.6.12、NoteList

开卷语

俗话说，“熟读唐诗三百首，不会作诗也会吟”。最近收集了很多 **Android** 的示例代码，从这些代码的阅读和实验中学学习到很多知识，从而产生写这个系列的打算，目标就是一步步跟着实例进行动手实作，真正从“做”中体会和学习 **Android** 开发。

本文目标是 **Android** 自带的一个范例程序：记事本，

预备知识

搭建开发环境，尝试编写”Hello World”，了解 **Android** 的基本概念，熟悉 **Android** 的 API(官方文档中都有，不赘述)。

程序截图

先来简单了解下程序运行的效果



程序入口点

类似于 win32 程序里的 WinMain 函数，Android 自然也有它的程序入口点。它通过在

AndroidManifest.xml 文件中配置来指明, 可以看到名为 NotesList 的 activity 节点下有这样一个 intent-filter, 其 action 为 android.intent.action.MAIN, Category 指定为 android.intent.category.LAUNCHER, 这就指明了这个 activity 是作为入口 activity, 系统查找找到它后, 就会创建这个 activity 实例来运行, 若未发现就不启动(你可以把 MAIN 改名字试试)。

```
<intent-filter>
<action android:name="android.intent.action.MAIN"
/>
<category android:name="android.intent.category.LAUNCHER"
/>
</intent-filter>
```

NotesList 详解

就从入口点所在的 activity(见图 1)开始, 可以看到这个 activity 最重要的功能就是显示日志列表。这个程序的日志都存放在 SQLite 数据库中, 因此需要读取出所有的日志记录并显示。

先来看两个重要的私有数据, 第一个 PROJECTION 字段指明了“日志列表”所关注的数据库中的字段(即只需要 ID 和 Title 就可以了)。

```
private
static
final String[] PROJECTION =
new String[] {
    Notes._ID, // 0
    Notes.TITLE, // 1
};
```

第二个字段 COLUMN_INDEX_TITLE 指明 title 字段在数据表中的索引。

```
private
static
final
int COLUMN_INDEX_TITLE = 1;
然后就进入第一个调用的函数 onCreate。
    Intent intent = getIntent();
    if (intent.getData() ==
null)
    {
        intent.setData(Notes.CONTENT_URI);
    }
```

因为 NotesList 这个 activity 是系统调用的, 此时的 intent 是不带数据和操作类型的, 系统只是在其中指明了目标组件是 Notelist, 所以这里把 "content://"

com.google.provider.NotePad/notes” 保存到 intent 里面，这个 URI 地址指明了数据库中的数据表名（参见以后的 NotePadProvider 类），也就是保存日志的数据表 notes。

```
Cursor cursor = managedQuery(getIntent().getData(), PROJECTION, null, null,
Notes.DEFAULT_SORT_ORDER);
```

然后调用 managedQuery 函数查询出所有的日志信息，这里第一个参数就是上面设置的” content:// com.google.provider.NotePad/notes” 这个 URI，即 notes 数据表。PROJECTION 字段指明了结果中所需要的字段，Notes.DEFAULT_SORT_ORDER 指明了结果的排序规则。实际上 managedQuery 并没有直接去查询数据库，而是通过 Content Provider 来完成实际的数据库操作，这样就实现了逻辑层和数据库层的分离。

```
SimpleCursorAdapter adapter =
```

```
new SimpleCursorAdapter(this, R.layout.noteslist_item, cursor,
                        new String[] { Notes.TITLE }, new
int[] { android.R.id.text1 });
setListAdapter(adapter);
```

查询出日志列表后，构造一个 CursorAdapter，并将其作为 List View 的数据源，从而在界面上显示出日志列表。可以看到，第二个参数是 R.layout.noteslist_item，打开对应的 noteslist_item.xml 文件，

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/text1"
    android:layout_width="fill_parent"
    android:layout_height="?android:attr/listPreferredItemHeight"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:gravity="center_vertical"
    android:paddingLeft="5dip"
    android:singleLine="true"
/>
```

就是用来显示一条日志记录的 TextView，最后两个字段指明了实际的字段映射关系，通过这个 TextView 来显示一条日志记录的 title 字段。

处理“选择日志”事件

既然有了“日志列表”，就自然要考虑如何处理某一条日志的单击事件，这通过重载 onItemClick 方法来完成，

```
@Override
protected
void onItemClick(ListView l, View v, int position, long id) {
    Uri uri = ContentUris.withAppendedId(getIntent().getData(), id);

    String action = getIntent().getAction();
    if (Intent.ACTION_PICK.equals(action)
Intent.ACTION_GET_CONTENT.equals(action)) {
        // The caller is waiting for us to return a note selected by
```



```

        // the user. The have clicked on one, so return it now.
        setResult(RESULT_OK, new Intent().setData(uri));
    } else {
        // Launch activity to view/edit the currently selected item
        startActivity(new Intent(Intent.ACTION_EDIT, uri));
    }
}

```

首先通过”content://com.google.provider.NotePad/notes”和日志的id号拼接得到选中日志的真正URI,然后创建一个新的Intent,其操作类型为Intent.ACTION_EDIT,数据域指出待编辑的日志URI(这里只分析else块)。

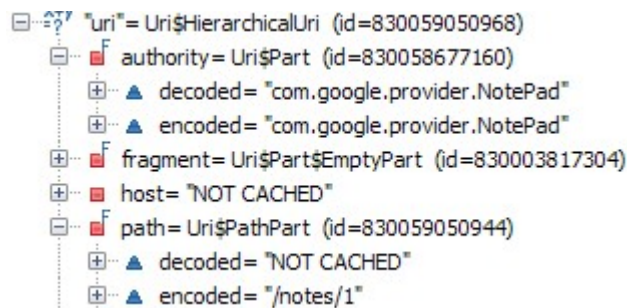
Intent 深度剖析

那么,上面这句startActivity(new Intent(Intent.ACTION_EDIT, uri))执行后会发生什么事情呢?这时候Android系统就跳出来接管了,它会根据intent中的信息找到对应的activity,在这里找到的是NoteEditor这个activity,然后创建这个activity的实例并运行。

那么,Android又是如何找到NoteEditor这个对应的activity的呢?这就是intent发挥作用的时刻了。

```
new Intent(Intent.ACTION_EDIT, uri)
```

这里的Intent.ACTION_EDIT=”android.intent.action.EDIT”,另外通过设置断点,我们看下这里的uri值:



可以看到选中的日志条目的URI是:content://com.google.provider.NotePad/notes/1

然后我们再来看下AndroidManifest.xml,其中有这个provider

```

<provider android:name="NotePadProvider"
    android:authorities="com.google.provider.NotePad"
/>

```

发现没有?它也有com.google.provider.NotePad,这个是content://com.google.provider.NotePad/notes/1的一部分,同时

```

<activity android:name="NoteEditor"
    android:theme="@android:style/Theme.Light"

```




```

        android:label="@string/title_note"
        android:screenOrientation="sensor"
        android:configChanges="keyboardHidden|orientation"
    >
    <!-- This filter says that we can view or edit the data of
           a single note -->
    <intent-filter android:label="@string/resolve_edit">
    <action android:name="android.intent.action.VIEW"
    />
    <action android:name="android.intent.action.EDIT"
    />
    <action android:name="com.android.notepad.action.EDIT_NOTE"
    />
    <category android:name="android.intent.category.DEFAULT"
    />
    <data android:mimeType="vnd.android.cursor.item/vnd.google.note"
    />
    </intent-filter>
    <!-- This filter says that we can create a new note inside
           of a directory of notes. -->
    <intent-filter>
    <action android:name="android.intent.action.INSERT"
    />
    <category android:name="android.intent.category.DEFAULT"
    />
    <data android:mimeType="vnd.android.cursor.dir/vnd.google.note"
    />
    </intent-filter>
</activity>

```

上面第一个 `intent-filter` 中有一个 `action` 名为 `android.intent.action.EDIT`，而前面我们创建的 `Intent` 也正好是

`Intent.ACTION_EDIT=" android.intent.action.EDIT"`，想必大家已经明白是怎么回事了吧。

下面就进入 `activity` 选择机制了：

系统从 `intent` 中获取 `uri`，得到了 `content://com.google.provider.NotePad/notes/1`，去掉开始的 `content:` 标识，得到 `com.google.provider.NotePad/notes/1`，然后获取前面的 `com.google.provider.NotePad`，然后就到 `AndroidManifest.xml` 中找到 `authorities` 为 `com.google.provider.NotePad` 的 `provider`，这个就是后面要讲的 `contentprovider`，然后就加载这个 `content provider`。

```

<provider android:name="NotePadProvider"
        android:authorities="com.google.provider.NotePad"

```




```
</>
```

在这里是 NotePadProvider,然后调用 NotePadProvider 的 gettype 函数,并把上述 URI 传给这个函数,函数返回 URI 所对应的类型(这里返回 Notes.CONTENT_ITEM_TYPE,代表一条日志记录,而 CONTENT_ITEM_TYPE = "vnd.android.cursor.item/vnd.google.note")。

```
@Override
public String getType(Uri uri) {
    switch (sUriMatcher.match(uri)) {
        case NOTES:
            return Notes.CONTENT_ITEM_TYPE;
        case NOTE_ID:
            return Notes.CONTENT_ITEM_TYPE;
        default:
            throw
new IllegalArgumentException("Unknown URI "
+ uri);
    }
}
```

上面的 sUriMatcher.match 是用来检测 uri 是否能够被处理,而 sUriMatcher.match(uri) 返回值其实是由

```
sUriMatcher =
new UriMatcher(UriMatcher.NO_MATCH);
sUriMatcher.addURI(NotePad.AUTHORITY, "notes", NOTES);
sUriMatcher.addURI(NotePad.AUTHORITY, "notes/#", NOTE_ID);
```

决定的。

然后系统使用获得的"vnd.android.cursor.item/vnd.google.note"和"android.intent.action.EDIT"到 androidmanifest.xml 中去找匹配的 activity.

```
<intent-filter android:label="@string/resolve_edit">
<action android:name="android.intent.action.VIEW"
/>
<action android:name="android.intent.action.EDIT"
/>
<action android:name="com.android.notepad.action.EDIT_NOTE"
/>
<category android:name="android.intent.category.DEFAULT"
/>
<data android:mimeType="vnd.android.cursor.item/vnd.google.note"
/>
```



</intent-filter>

正好 NoteEditor 这个 activity 的 intent-filter 满足上述条件,这样就找到了 NoteEditor。于是系统加载这个类并实例化,运行,然后就到了 NoteEditor 的 onCreate 函数中(见后续文章)。

小技巧

1, 在命令行中使用 "adb shell" 命令进入系统中,然后 "cd app" 进入应用程序所在目录,"rm XXX" 就可以删除你指定的 apk,从而去掉其在系统顶层界面占据的图标,若两次 "cd data" 则可以进入应用程序使用的数据目录,你的数据可以保存在这里,例如 Notepad 就是把其数据库放在它的 databases 目录下,名为 note_pad.db.

2, 第一次启动模拟器会比较慢,但以后就别关闭模拟器了,修改代码,调试都不需要再次启动的,直接修改后 run 或 debug 就是。

4.7、Tab与TabHost

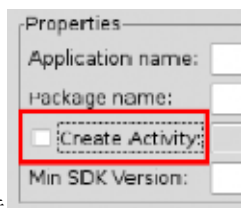


这就是 Tab, 而盛放 Tab 的容器就是 TabHost

如何实现??

每一个 Tab 还对应了一个布局,这个就有点好玩了。一个 Activity, 对应了多个功能布局。

①新建一个 Tab 项目, 注意, 不要生成 main Activity



这里不要选

②在包里面新建一个类 MyTab, 继承于 TabActivity

其实, TabActivity 是 Activity 的子类

```
package zyf.tab.test;
import android.app.TabActivity;
public class MyTab extends TabActivity {
}
```

③从父类继承 onCreate() 入口方法

```

package zyf.tab.test;
import android.app.TabActivity;
import android.os.Bundle;
public class MyTab extends TabActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
    }
}

```

④在 Manifest.xml 文件中注册一下 MyTab 类(Activity)

⑤这时候，需要设计一下标签页对应的布局，一般采用 FrameLayout 作为根布局，每个标签页面对应一个子节点的 Layout

```

        android:layout_width="fill_parent" android:layout_height="fill_parent">
        android:layout_width="fill_parent" android:layout_height="fill_parent"
        android:orientation="vertical">
    android:layout_height="wrap_content" android:text="EditText"
        android:textSize="18sp">
        android:layout_height="wrap_content" android:text="Button">
        android:layout_width="fill_parent" android:layout_height="fill_parent"
        android:orientation="vertical">
        android:layout_width="wrap_content" android:layout_height="wrap_content">
        android:layout_width="fill_parent" android:layout_height="fill_parent"
        android:orientation="vertical">
        android:layout_width="166px" android:layout_height="98px"
        android:orientation="vertical">
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:text="RadioButton">
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:text="RadioButton">

```

⑥首先，应该声明 TabHost，然后用 LayoutInflater 过滤出布局来，给 TabHost 加上含有 Tab 页面的 FrameLayout

```

private TabHost myTabhost;

myTabhost=this.getTabHost();//从 TabActivity 上面获取放置 Tab 的 TabHost
LayoutInflater.from(this).inflate(R.layout.main, myTabhost.getTabContentView(), true);
//from(this)从这个 TabActivity 获取 LayoutInflater
//R.layout.main 存放 Tab 布局
//通过 TabHost 获得存放 Tab 标签页内容的 FrameLayout
//是否将 inflate 拴系到根布局元素上
myTabhost.setBackgroundColor(Color.argb(150, 22, 70, 150));

```



//设置一下 TabHost 的颜色

⑦接着，在 TabHost 创建一个标签，然后设置一下标题/图标/标签页布局

```
myTabhost
.addTab(myTabhost.newTabSpec("TT")// 制造一个新的标签 TT
.setIndicator("KK",
.getResources().getDrawable(R.drawable.ajjc))
// 设置一下显示的标题为 KK，设置一下标签图标为 ajjc
.setContent(R.id.widget_layout_red));
//设置一下该标签页的布局内容为 R.id.widget_layout_red，这是 FrameLayout 中的一个子 Layout
```

⑧标签切换事件处理，setOnTabChangeListener

```
myTabhost.setOnTabChangeListener(new OnTabChangeListener(){
@Override
public void onTabChanged(String tabId) {
// TODO Auto-generated method stub
}
});
```

⑨各个标签页的动态 MENU

先把 XML 中设计好的 MENU 放到一个 int 数组里

```
private static final int myMenuResources[] = { R.menu.phonebook_menu,
R.menu.addphone_menu, R.menu.chatting_menu, R.menu.userapp_menu };
```

在 setOnTabChangeListener()方法中根据标签的切换情况来设置 myMenuSettingTag

```
@Override
public void onTabChanged(String tagString) {
// TODO Auto-generated method stub
if (tagString.equals("One")) {
myMenuSettingTag = 1;
}
if (tagString.equals("Two")) {
myMenuSettingTag = 2;
}
if (tagString.equals("Three")) {
myMenuSettingTag = 3;
}
if (tagString.equals("Four")) {
myMenuSettingTag = 4;
}
}
```



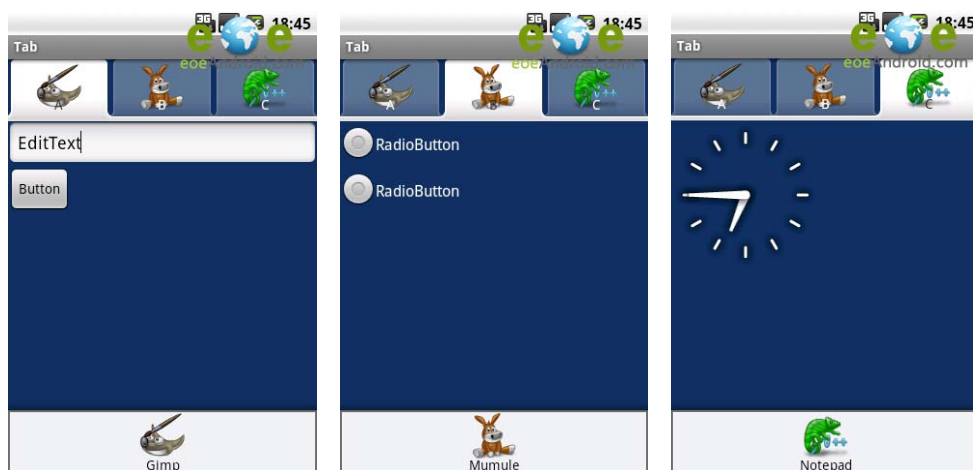
```
if (myMenu != null) {  
    onCreateOptionsMenu(myMenu);  
}  
}
```

然后 onCreateOptionsMenu(Menu menu) 方法中通过 MenuInflater 过滤器动态加入 MENU

```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    // TODO Auto-generated method stub  
    // Hold on to this  
    myMenu = menu;  
    myMenu.clear();//清空 MENU 菜单  
    // Inflate the currently selected menu XML resource.  
    MenuInflater inflater = getMenuInflater();  
    //从 TabActivity 这里获取一个 MENU 过滤器  
    switch (myMenuSettingTag) {  
        case 1:  
            inflater.inflate(myMenuResources[0], menu);  
            //动态加入数组中对应的 XML MENU 菜单  
            break;  
        case 2:  
            inflater.inflate(myMenuResources[1], menu);  
            break;  
        case 3:  
            inflater.inflate(myMenuResources[2], menu);  
            break;  
        case 4:  
            inflater.inflate(myMenuResources[3], menu);  
            break;  
        default:  
            break;  
    }  
    return super.onCreateOptionsMenu(menu);  
}
```

⑩运行效果





4.8、RatingBar

4.8.1、例一



和 SeekBar 类似，RatingBar 也是扩展，只不过它是 SeekBar 和 ProgressBar 的扩展，它是用星星来显示等级。当使用默认尺寸的 RatingBar 时，用户能够触摸/拖动或使用箭头键来设置评级。较小的 RatingBar 类型 (ratingBarStyleSmall) 以及较大的单一指示器类型 (ratingBarStyleIndicator) 都不支持用户交互，只适用于指示器。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
    />
    <RatingBar
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        style="?android:attr/ratingBarStyleIndicator"
```

```

        android:id="@+id/ratingbar_Indicator"
    />
    <RatingBar
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        style="?android:attr/ratingBarStyleSmall"
        android:id="@+id/ratingbar_Small"
        android:numStars="20"
    />
    <RatingBar
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        style="?android:attr/ratingBarStyle"
        android:id="@+id/ratingbar_default"
    />
</LinearLayout>

```

Java 代码: AndroidRatingBar.java

```

package com.AndroidRatingBar;

import android.app.Activity;
import android.os.Bundle;
import android.widget.RatingBar;
import android.widget.Toast;

public class AndroidRatingBar extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final RatingBar ratingBar_Small =
(RatingBar)findViewById(R.id.ratingbar_Small);
        final RatingBar ratingBar_Indicator =
(RatingBar)findViewById(R.id.ratingbar_Indicator);
        final RatingBar ratingBar_default =
(RatingBar)findViewById(R.id.ratingbar_default);
        ratingBar_default.setOnRatingBarChangeListener(new
RatingBar.OnRatingBarChangeListener() {
            @Override
            public void onRatingChanged(RatingBar ratingBar, float rating,
                boolean fromUser) {

```



```

        // TODO Auto-generated method stub
        ratingBar_Small.setRating(rating);
        ratingBar_Indicator.setRating(rating);
        Toast.makeText(AndroidRatingBar.this,
            "rating:"+String.valueOf(rating),
                Toast.LENGTH_LONG).show();
    }
    });
}
}

```

4.8.2、例二



一、概述

RatingBar 是 SeekBar 和 ProgressBar 的扩展，用星星来评级。使用的默认大小 RatingBar 时，用户可以触摸/拖动或使用键来设置评分，它有两种样式（大、小），其中大的只适合指示，不适合于用户交互。

二、实例

1. 布局文件

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:paddingLeft="10dip"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

```




```
<RatingBar android:id="@+id/ratingbar1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:numStars="3"
    android:rating="2.5" />

<RatingBar android:id="@+id/ratingbar2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:numStars="5"
    android:rating="2.25" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dip">

    <TextView android:id="@+id/rating"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <RatingBar android:id="@+id/small_ratingbar"
        style="?android:attr/ratingBarStyleSmall"
        android:layout_marginLeft="5dip"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical" />

</LinearLayout>

<RatingBar android:id="@+id/indicator_ratingbar"
    style="?android:attr/ratingBarStyleIndicator"
    android:layout_marginLeft="5dip"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_vertical" />
```

```
</LinearLayout>
```

2. Java 代码

```
package wjq.WidgetDemo;

import android.app.Activity;
import android.os.Bundle;
```



```

import android.widget.RatingBar;
import android.widget.TextView;
import android.widget.RatingBar.OnRatingBarChangeListener;

public class RatingBarDemo extends Activity implements
    OnRatingBarChangeListener {
    private RatingBar mSmallRatingBar;
    private RatingBar mIndicatorRatingBar;
    private TextView mRatingText;

    /*
     * (non-Javadoc)
     *
     * @see android.app.Activity#onCreate(android.os.Bundle)
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.ratingbarpage);

        mRatingText = (TextView) findViewById(R.id.rating);

        // We copy the most recently changed rating on to these indicator-only
        // rating bars
        mIndicatorRatingBar = (RatingBar) findViewById(R.id.indicator_ratingbar);
        mSmallRatingBar = (RatingBar) findViewById(R.id.small_ratingbar);

        // The different rating bars in the layout. Assign the listener to us.
        ((RatingBar)findViewById(R.id.ratingbar1)).setOnRatingBarChangeListener(this);
        ((RatingBar)findViewById(R.id.ratingbar2)).setOnRatingBarChangeListener(this);
    }

    @Override
    public void onRatingChanged(RatingBar ratingBar, float rating,
        boolean fromUser) {
        final int numStars = ratingBar.getNumStars();
        mRatingText.setText(
            " 受欢迎度" + rating + "/" + numStars);

        // Since this rating bar is updated to reflect any of the other rating
        // bars, we should update it to the current values.
        if (mIndicatorRatingBar.getNumStars() != numStars) {

```

```

        mIndicatorRatingBar.setNumStars(numStars);
        mSmallRatingBar.setNumStars(numStars);
    }
    if (mIndicatorRatingBar.getRating() != rating) {
        mIndicatorRatingBar.setRating(rating);
        mSmallRatingBar.setRating(rating);
    }
    final float ratingBarStepSize = ratingBar.getStepSize();
    if (mIndicatorRatingBar.getStepSize() != ratingBarStepSize) {
        mIndicatorRatingBar.setStepSize(ratingBarStepSize);
        mSmallRatingBar.setStepSize(ratingBarStepSize);
    }
}
}

```

4.9、Date/Time Set

4.9.1、DatePicker/TimePicker



实现步骤：

第一步：建立Android 工程：DatePickerDemo。

第二步：编写Activity 的子类别：DatePickerDemo，其程序代码如下：

```

package com.a3gs.datepicker;
import java.util.Calendar;
import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.widget.DatePicker;

```



```
import android.widget.TextView;
import android.widget.TimePicker;
public class DatePickerDemo extends Activity {
    private DatePicker dPicker;
    private TimePicker tPicker;
    private TextView tTextView;
    // 声明时间变量
    private int mYear;
    private int mMonth;
    private int mDay;
    private int mHour;
    private int mMinute;
    private Calendar calendar;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        /*取得当前日期与时间*/
        calendar = Calendar.getInstance();
        mYear = calendar.get(Calendar.YEAR);
        mMonth = calendar.get(Calendar.MONTH);
        mDay = calendar.get(Calendar.DAY_OF_MONTH);
        mHour = calendar.get(Calendar.HOUR_OF_DAY);
        mMinute = calendar.get(Calendar.MINUTE);
        tTextView = (TextView) findViewById(R.id.tTextView);
        tTextView.setTextColor(Color.RED);
        updateTime();
        dPicker = (DatePicker) findViewById(R.id.dPicker);
        tPicker = (TimePicker) findViewById(R.id.tPicker);
        tPicker.setIs24HourView(true);    // 时间设为24小时制
        /* 当日期改变时, tTextView的日期也改变*/
        dPicker.init(mYear, mMonth, mDay, new DatePicker.OnDateChangedListener() {
            @Override
            public void onDateChanged(DatePicker view, int year,
                int monthOfYear, int dayOfMonth) {
                // TODO Auto-generated method stub
                mYear = year;
                mMonth = monthOfYear;
                mDay = dayOfMonth;
                updateTime();
            }
        });
        /* 当时间改变时, tTextView的时间也改变*/
    }
}
```



```

    tPicker.setOnTimeChangedListener(new TimePicker.OnTimeChangedListener() {
        @Override
        public void onTimeChanged(TimePicker view, int hourOfDay, int minute) {
            // TODO Auto-generated method stub
            mHour = hourOfDay;
            mMinute = minute;
            updateTime();
        }
    });
}

/* 更新时间 */
private void updateTime() {
    tTextView.setText(
        new StringBuilder().append(mYear).append("-")
            .append(formatTime(mMonth+1)).append("-")
            .append(formatTime(mDay)).append(" ")
            .append(formatTime(mHour)).append(":")
            .append(formatTime(mMinute))
    );
}

/* 时间格式 */
private String formatTime(int time) {
    String timeStr = "";
    if (time < 10)
        timeStr = "0" + String.valueOf(time);
    else
        timeStr = String.valueOf(time);
    return timeStr;
}
}

```

第三步：修改res/layout/main.xml，其代码如下：

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <DatePicker
        android:id="@+id/dPicker"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    />

```



```
<TimePicker
    android:id="@+id/tPicker"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
/>

<TextView
    android:id="@+id/tTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:textSize="20sp"
/>

</LinearLayout>
```

扩展学习

Android API 另外提供了其它的对象来实现动态修改日期时间的功能: DatePickerDialog 与 TimePickerDialog。这两种类型的对象最大的差别在于 DatePicker 与 TimePicker 是直接显示在屏幕画面上, 而 DatePickerDialog 与 TimePickerDialog 对象则是以跳出 Dialog 的方式来显示, 有兴趣的朋友可以自己研究一下。

```
new DatePickerDialog(DatePickerDemo.this, new DatePickerDialog.OnDateSetListener() {
    @Override
    public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth) {
        // TODO Auto-generated method stub

    }
}, mYear, mMonth, mDay).show();

new TimePickerDialog(DatePickerDemo.this, new TimePickerDialog.OnTimeSetListener() {
    @Override
    public void onTimeSet(TimePicker view, int hourOfDay, int minute) {

        // TODO Auto-generated method stub
    }
}, mHour, mMinute, false).show();
```



4.9.2、DatePickerDialog/TimePickerDialog



方法例:

1. 声明:

```
int hour = 0;
int minute = 0;
int year = 2010;
int month = 0;
int day = 1;
static final int TIME_DIALOG_ID = 0;
static final int DATE_DIALOG_ID = 1;
```

2. 添加 Java 代码:

```
@Override
protected Dialog onCreateDialog(int id) {
    // TODO Auto-generated method stub
    //return super.onCreateDialog(id);
    switch (id) {
        case TIME_DIALOG_ID:
            return new TimePickerDialog( this, mTimeSetListener, hour, minute,
false);

        case DATE_DIALOG_ID:
            return new DatePickerDialog( this, mDateSetListener, year, month,
day);
    }
    return null;
}
```

```
private TimePickerDialog.OnTimeSetListener mTimeSetListener = new
```

作者:craining (曲阜师范大学) 个人主页:<http://craining.blog.163.com/> 邮箱:craining@163.com 119



```

TimePickerDialog.OnTimeSetListener() {
    @Override
    public void onTimeSet(TimePicker view, int hourOfDay, int minute1) {
        // TODO Auto-generated method stub
        hour = hourOfDay;
        minute = minute1;
        Toast.makeText(getApplicationContext(),
            "You have selected : " + hour + ":" + minute,
            Toast.LENGTH_SHORT).show();
    }
};

private DatePickerDialog.OnDateSetListener mDateSetListener = new
DatePickerDialog.OnDateSetListener() {
    @Override
    public void onDateSet(DatePicker view, int year1, int month1, int day1){
        // TODO Auto-generated method stub
        year = year1;
        month = month1 - 1;
        day = day1;
        Toast.makeText(getApplicationContext(),
            "You have selected : " + year + "-" + month + "-" + day,
            Toast.LENGTH_SHORT).show();
    }
};

```

3. 调用对话框:

```

showDialog(TIME_DIALOG_ID);
showDialog(DATE_DIALOG_ID);

```

4.10、WebView

4.10.1、WebView的使用

1、添加权限: AndroidManifest.xml 中必须使用许可"android.permission. INTERNET", 否则会出 Web page not available 错误。

2、在要 Activity 中生成一个 WebView 组件: `WebView webView = new WebView(this);`

3、设置 WebView 基本信息:

如果访问的页面中有 Javascript, 则 webview 必须设置支持 Javascript。
`webView.getSettings().setJavaScriptEnabled(true);`
 触摸焦点起作用




```
requestFocus();
```

取消滚动条

```
this.setScrollBarStyle(SCROLLBARS_OUTSIDE_OVERLAY);
```

4、设置 WebView 要显示的网页：

互联网用：webView.loadUrl("http://www.google.com");

本地文件用：

```
webView.loadUrl("file:///android_asset/XX.html");
```

本地文件存放在：assets 文件中

5、如果希望点击链接由自己处理，而不是新开 Android 的系统 browser 中响应该链接。

给 WebView 添加一个事件监听对象 (WebViewClient)

并重写其中的一些方法

shouldOverrideUrlLoading：对网页中超链接按钮的响应。

当按下某

个连接时 WebViewClient 会调用这个方法，并传递参数：按下的 url

```
onLoadResource
```

```
onPageStart
```

```
onPageFinish
```

```
onReceiveError
```

```
onReceivedHttpAuthRequest
```

6、如果用 webview 点链接看了很多页以后，如果不做任何处理，点击系统 “Back” 键，整个浏览器会调用 finish() 而结束自身，如果希望浏览的网页回退而不是退出浏览器，需要在当前 Activity 中处理并消费掉该 Back 事件。覆盖 Activity 类的 onKeyDown(int keyCode, KeyEvent event) 方法。

```
public boolean onKeyDown(int keyCode, KeyEvent event){
    if(webView.canGoBack() && keyCode == KeyEvent.KEYCODE_BACK){
        webview.goBack(); //goBack()表示返回 webView 的上一页面
        return true;
    }
    return false;
}
```

4.11、ScrollView

4.11.1、ScrollView的使用

XML 源码:

作者: craining (曲阜师范大学) 个人主页: <http://craining.blog.163.com/> 邮箱: craining@163.com 121



```
<?xml version="1.0" encoding="UTF-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_width="fill_parent"
    android:scrollbars="vertical"
    android:background="#FF000000">
    <ScrollView
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <LinearLayout
            xmlns:android="http://schemas.android.com/apk/res/android"
            android:orientation="vertical"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:background="#FFc0c0c0"
            android:padding="10dip">
            <TextView
                android:id="@+id/showText"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="内容:"
                android:textColor="#FF000000"
                android:autoLink="all"
                android:textSize="20sp"/>
            <RelativeLayout
                xmlns:android="http://schemas.android.com/apk/res/android"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:gravity="left"
            >
            <Button
                android:id="@+id/btndownload"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="按钮 1"
            >
            </Button>
        </RelativeLayout>
    </RelativeLayout>
```



```

        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="left"
    >
    <Button
        android:id="@+id/btnreturn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="按钮 2"
    >
    </Button>
</RelativeLayout>

    <RelativeLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="right"
    >
    <Button
        android:id="@+id/btnndel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="按钮 3"
    >
    </Button>
</RelativeLayout>
</LinearLayout>
</ScrollView>
</LinearLayout>

```

Java 代码:

```

public class ActivityShow extends Activity {

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        super.setTitle("ScrollView 实例: ");
        TextView tv = (TextView) findViewById(R.id.showText);
        tv.setText("这是要显示的内容");
        button3 = (Button) findViewById(R.id.btnndel);
        button2 = (Button) findViewById(R.id.btnreturn);
        button1 = (Button) findViewById(R.id.btndownload);
        /* 监听 button 的事件信息 */
    }
}

```



```
button1.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {

        }
    }
});
button2.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {

        }
    }
});

button3.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {

        }
    }
});
}
```

4.12、GridView

4.12.1、GridView的使用

网格布局：是一个ViewGroup以网格显示它的子视图（**view**）元素，即二维的、滚动的网格。网格元素通过 **ListAdapter** 自动插入到网格。**ListAdapter** 跟上面的列表布局是一样的，这里就不重复累述了。

下面也通过一个例子来，创建一个显示图片缩略图的网格。当一个元素被选择时，显示该元素在列表中的位置的消息。

1)、首先，将上面实践截取的图片放入 **res/drawable/**

2)、**res/layout/main.xml** 的内容置为如下：这个 **GridView** 填满整个屏幕，而且它的属性都很好理解，按英文单词的意思就对了。

```
<?xml version="1.0" encoding="utf-8"?>
<GridView xmlns:android="http://schemas.android.com/apk/res/andro
id"
    android:id="@+id/gridview"
```



```

    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:columnWidth="90dp"
    android:numColumns="auto_fit"
    android:verticalSpacing="10dp"
    android:horizontalSpacing="10dp"
    android:stretchMode="columnWidth"
    android:gravity="center"/>

```

3)、然后，HelloWorld.java 文件中 **onCreate()** 函数如下：

```

    public void onCreate(Bundle savedInstanceState) {          super
    onCreate(savedInstanceState);
        setContentView(R.layout.main);
        GridView gridView = (GridView) findViewById(R.id.gridvie
w);
        gridView.setAdapter(new ImageAdapter(this));
        gridView.setOnItemClickListener(new OnItemClickListener()
{
            public void onItemClick(AdapterView<?> parent, View
v, int position, long id) {          Toast.makeText(Hello
World.this, " " + position, Toast.LENGTH_SHORT).show();
            }          });
    }

```

onCreate() 函数跟通常一样，首先调用超类的 **onCreate()** 函数函数，然后通过 **setContentView()** 为活动（**Activity**）加载布局文件。紧接着是，通过 **GridView** 的 **id** 获取布局文件中的 **gridview**，然后调用它的 **setListAdapter(ListAdapter)** 函数填充它，它的参数是一个我们自定义的 **ImageAdapter**。后面的工作跟列表布局中一样，为监听网格中的元素被点击的事件而做的工作。

4)、实现我们自定义 **ImageAdapter**，新添加一个类文件，它的代码如下：

```

package skynet.com.cnblogs.www;import android.content.Context;imp
ort android.view.View;import android.view.ViewGroup;import androi
d.widget.BaseAdapter;import android.widget.GridView;import androi
d.widget.ImageView;public class ImageAdapter extends BaseAdapter
{    private Context mContext;    public ImageAdapter(Context c)
    {        mContext = c;    }    public int getCount() {
return mThumbIds.length;    }    public Object getItem(int posit
ion) {        return null;    }    public long getItemId(int pos
ition) {        return 0;    }    // create a new ImageView for
each item referenced by the Adapter    public View getView(int p
osition, View convertView, ViewGroup parent) {        ImageView
imageView;        if (convertView == null) { // if it's not rec

```

```

ycled, initialize some attributes        imageView = new Image
imageView(mContext);                    imageView.setLayoutParams(new GridV
iew.LayoutParams(85, 85));                imageView.setScaleType(Image
imageView.ScaleType.CENTER_CROP);          imageView.setPadding(8,
8, 8, 8);    } else {                    imageView = (ImageView) c
onvertView;    }    imageView.setImageResource(mThumbIds
[position]);    return imageView;    }    // references to o
ur images    private Integer[] mThumbIds = {                R.drawab
le.linearlayout1, R.drawable.linearlayout2,                R.drawabl
e.linearlayout3, R.drawable.listview,                R.drawable.rela
tivelayout, R.drawable.tablelayout    };}

```

ImageAdapter类扩展自 **BaseAdapter**，所以首先得实现它所要求必须实现的方法。构造函数和 `getcount()` 函数很好理解，而 `getItem(int)` 应该返回实际对象在适配器中的特定位置，但是这里我们不需要。类似地，`getItemId(int)` 应该返回元素的行号，但是这里也不需要。

这里重点要介绍的是 `getView()` 方法，它为每个要添加到 **ImageAdapter** 的图片都创建了一个新的 **View**。当调用这个方法时，一个 **View** 是循环再用的，因此要确认对象是否为空。如果是空的话，一个 **ImageView** 就被实例化且配置想要的显示属性：

- `setLayoutParams(ViewGroup.LayoutParams)`：设置 **View** 的高度和宽度，这确保不管 **drawable** 中图片的大小，每个图片都被重新设置大小且剪裁以适应这些尺寸。
- `setScaleType(ImageView.ScaleType)`：声明图片应该向中心剪裁（如果需要的话）。
- `setPadding(int, int, int, int)`：定义补距，如果图片有不同的纵横比，小的补距将导致更多的剪裁以适合设置的 **ImageView** 的高度和宽度。

如果 **View** 传到 `getView()` 不是空的，则本地的 **ImageView** 初始化时将循环再用 **View** 对象。在 `getView()` 方法末尾，`position` 整数传入 `setImageResource()` 方法以从 `mThumbIds` 数组中选择图片。

运行程序会得到如下结果（点击第一张图片之后）：





图 8、网格布局

4.13、GameView

4.13.1、显示到一个布局中

```
public class GameActivity extends Activity {
    private GameView gameView = null;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        gameView = new GameView(this);
        LinearLayout mainLayout = (LinearLayout)findViewById(R.id.mainlayout);
        Button button = (Button) findViewById(R.id.Button01);
        mainLayout.addView(gameView);

        button.setOnClickListener(new Button.OnClickListener() {
            @Override
            public void onClick(View v) {
                gameView.Angle += 10;
            }
        });
    }
}
```



```

    });

//    addContentView(gameView, new ViewGroup.LayoutParams(200,150));
}
}

```

4.14、Toaste

以下两个方法的使用方法：

```
DisplayToast(this, "");
```

4.14.1、短时间显示

```

public static void DisplayToast(Context context, String str) {
    Toast.makeText(context, str, Toast.LENGTH_SHORT).show();
}

```

4.14.2、长时间显示

```

public static void DisplayToast(Context context, String str) {
    Toast.makeText(context, str, Toast.LENGTH_LONG).show();
}

```

4.15、对话框

4.15.1、简单的对话框：

```

private void openOptionsDialog() {
    // TODO Auto-generated method stub
    new AlertDialog.Builder(this).setTitle("关于 Android ")
        .setMessage("Android is good !")
        .show();
}

```

4.15.2、包含两个按钮的对话框

0. 声明对话框值

```
private static final int NOTEDLG = 1;
```



1. 定义对话框

```
private Dialog errorDlg(Context context) {
    AlertDialog.Builder builder = new AlertDialog.Builder(context);

    builder.setIcon(R.drawable.img);
    builder.setTitle("提示: ");
    builder.setMessage("您确定要````?");

    builder.setPositiveButton("确定",
        new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int
whichButton){
                //确定按钮的响应
            }
        });
    builder.setNegativeButton("取消",
        new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int whichButton)
{
                //取消按钮的响应
            }
        });
    return builder.create();
}
```

2. 定义显示对话框

```
protected Dialog onCreateDialog(int id) {
    switch (id) {
        case NOTEDLG:
            return errorDlg(~~~~~.this); //~~~是activity名
    }
    return null;
}
```

3. 显示对话框命令

```
showDialog(NOTEDLG);
```

4.15.3、三个按钮的提示框

0. 声明对话框值

```
private static final int NOTEDLG2 = 2;
```



1. 定义对话框

```
private Dialog buildDialog2(Context context) {
    AlertDialog.Builder builder = new AlertDialog.Builder(context);
    // builder.setIcon(R.drawable.alert_dialog_icon);
    builder.setTitle("提示: ");
    builder.setMessage("Hello! ");
    builder.setPositiveButton("确定",
        new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int whichButton)
            {

                setTitle("点击了对话框上的确定按钮");
            }
        });
    builder.setNeutralButton("进入详细",
        new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int whichButton)
            {

                setTitle("点击了对话框上的进入详细按钮");
            }
        });
    builder.setNegativeButton("取消",
        new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int whichButton)
            {

                setTitle("点击了对话框上的取消按钮");
            }
        });
    return builder.create();
}
```

2. 定义显示对话框

```
protected Dialog onCreateDialog(int id) {
    switch (id) {
        case NOTEDLG2:
            return buildDialog2 (~~~~~.this); //~~~是activity名
    }
    return null;
}
```

3. 显示对话框命令

```
showDialog(NOTEDLG2);
```



4.15.4、包含输入的dlg

1. 对话框布局文件:xml==== R.layout.dlg_testself.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <TextView
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_marginLeft="20dip"
        android:layout_marginRight="20dip"
        android:text="请输入...: "
        android:gravity="left"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/testdaysnum"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:layout_marginLeft="20dip"
        android:layout_marginRight="20dip"
        android:scrollHorizontally="true"
        android:autoText="false"
        android:capitalize="none"
        android:gravity="fill_horizontal"
        android:textAppearance="?android:attr/textAppearanceMedium" />
</LinearLayout>
```

2. Java 代码中生成对话框代码

2.1 声明:

```
private ProgressDialog m_Dialog;
```

2.2 实现对话框代码:

```
LayoutInflater factory = LayoutInflater.from(ChooseThings.this);
//得到自定义对话框
final View testDialogView = factory.inflate(R.layout.dlg_testself, null);
final EditText edit_GetDaysNum = (EditText)
testDialogView.findViewById( R.id.testdaysnum );
//创建对话框
AlertDialog.Builder testDialog = new
AlertDialog.Builder(ChooseThings.this);
```



```

testDialog.setTitle("提示: ");
testDialog.setView( testDialogView );//设置自定义对话框的样式
testDialog.setPositiveButton("确定", //设置"确定"按钮

new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int whichButton) {
        //输入完成后, 点击"确定"开始登陆

        String nullOrNot = edit_GetDaysNum.getText().toString();

        if( !TextUtils.isEmpty( nullOrNot ) ) {
            int getDaysNumFromDlg =
Integer.parseInt( edit_GetDaysNum.getText().toString() ); //将editText中的内容
转为int型

            m_Dialog = ProgressDialog.show (ChooseThings.this, "请等待...",
"正在为您计算...", true);
            new Thread() {
                public void run() {
                    try {
                        sleep(3000);
                    }
                    catch (Exception e) {
                        e.printStackTrace();
                    }
                    finally {
                        //登录结束, 取消m_Dialog对话框
                        m_Dialog.dismiss();
                        Intent intent = new Intent();
                        intent.setClass(ChooseThings.this, TestResult.class);
                        startActivity( intent );
                        ChooseThings.this.finish();
                    }
                }
            }.start();
        }
    }
});

testDialog.setNeutralButton("查看详细",
new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int whichButton) {
        }
    }
});

```



```

    });

    testDialog.setNegativeButton("取消", //设置"取消"按钮
    new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int whichButton) {
            //点击"取消"按钮之后退出程序
        }
    })

    .create();//创建
    testDialog.show();//显示
showDialog(NOTEDLG3);

```

4.15.5、圆形进度框

0. 声明对话框值

```
private static final int NOTEDLG4 = 4;
```

1. 定义对话框

```

private Dialog buildDialog4(Context context) {
    ProgressDialog dialog = new ProgressDialog(context);
    dialog.setTitle("正在下载歌曲");
    dialog.setMessage("请稍候.....");
    return dialog;
}

```

2. 定义显示对话框

```

protected Dialog onCreateDialog(int id) {
    switch (id) {
        case NOTEDLG4:
            return buildDialog4 (~~~~~.this); //~~~是activity名
    }
    return null;
}

```

3. 显示对话框命令

```
showDialog(NOTEDLG4);
```

4.15.6、AlertDialog.Builder

```
new AlertDialog.Builder(this)
```



```

.setTitle("Android 提示")
.setMessage("这是一个提示,请确定")
.show();
带一个确定的对话框
new AlertDialog.Builder(this)
    .setMessage("这是第二个提示")
    .setPositiveButton("确定",
        new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialoginterface, int i) {
                //按钮事件
            }
        })
    .show();
AlertDialog.Builder 还有很多复杂的用法,有确定和取消的对话框
new AlertDialog.Builder(this)
    .setTitle("提示")
    .setMessage("确定退出?")
    .setIcon(R.drawable.quit)
    .setPositiveButton("确定", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int whichButton) {
            setResult(RESULT_OK);//确定按钮事件
            finish();
        }
    })
    .setNegativeButton("取消", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int whichButton) {
            //取消按钮事件
        }
    })
    .show();

```

4.15.7、模式对话框

方法一:

```
alert("javascript 1 行代码, 模式对话框的效果.");
```

方法二:

```

Dialog d = new Dialog(NavActivity.this);
Window window = d.getWindow();
window.setFlags(WindowManager.LayoutParams.FLAG_BLUR_BEHIND,
    WindowManager.LayoutParams.FLAG_BLUR_BEHIND);
d.show();

```



4.16、拖动Button获得位置



布局文件

```
<?xml
version="1.0"
encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
<Button
android:id="@+id/btn_hello"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/hello"
/>
</LinearLayout>
```

代码:

```
public class DraftTest extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        DisplayMetrics dm=getResources().getDisplayMetrics();
        final int screenWidth=dm.widthPixels;
        final int screenHeight=dm.heightPixels-50;

        final Button b=(Button)findViewById(R.id.btn);
```

作者: craining (曲阜师范大学) 个人主页: <http://craining.blog.163.com/> 邮箱: craining@163.com 135



```

        b.setOnTouchListener(new OnTouchListener(){

int lastX,lastY;

        @Override
public boolean onTouch(View v, MotionEvent event) {
    // TODO Auto-generated method stub
    int ea=event.getAction();
    Log.i("TAG", "Touch:"+ea);

    switch(ea){
    case MotionEvent.ACTION_DOWN:

        lastX=(int)event.getRawX();
        lastY=(int)event.getRawY();
        break;
        /**
         * layout(l,t,r,b)
         * l Left position, relative to parent
         *   t Top position, relative to parent
         *   r Right position, relative to parent
         *   b Bottom position, relative to parent
         * */
        case MotionEvent.ACTION_MOVE:
            int dx=(int)event.getRawX()-lastX;
            int dy=(int)event.getRawY()-lastY;

            int l=v.getLeft()+dx;
            int b=v.getBottom()+dy;
            int r=v.getRight()+dx;
            int t=v.getTop()+dy;
            if(l<0){
                l=0;
                r=l+v.getWidth();
            }

            if(t<0){
                t=0;
                b=t+v.getHeight();
            }

            if(r>screenWidth){

```




```

        r=screenWidth;
        l=r-v.getWidth();
    }

    if(b>screenHeight){
        b=screenHeight;
        t=b-v.getHeight();
    }
    v.layout(l, t, r, b);

    lastX=(int)event.getRawX();
    lastY=(int)event.getRawY();
    Toast.makeText(DraftTest.this,
        "当前位置: "+l+", "+t+", "+r+", "+b,
        Toast.LENGTH_SHORT).show();
    v.postInvalidate();
    break;
    case MotionEvent.ACTION_UP:
        break;
    }
    return false;
});
}
}

```

5、Android UI 美化

5.1、简单美化Button、ImageButton、TextView等控件

1. Java 代码实现

对于 Android 自带的 Button 按钮控件很多网友感觉不是很美观，如果界面上按钮不多，我们可以通过一种简单的方法实现 Button 脱胎换骨的外观，考虑到效率 Android 的 layout 方式的 xml 文件先不用了，毕竟控件不多模拟一个个性化 Button 还是很简单，我们直接通过图片实现颜色的，代码如下：

```
private Button mBtn; //定义我们的按钮
在 onCreate 中加入
```

```
mBtn = (Button) findViewById(R.id.btn); //btn 为 layout 中的 Button ID
mBtn.setOnClickListener(new OnClickListener() {
```



```

public boolean onTouch(View arg0, MotionEvent arg1) {
    if(arg1.getAction() == MotionEvent.ACTION_DOWN) {
        arg0.setBackgroundResource(R.drawable.pressed); //按下的图片对应 pressed
    } else if(arg1.getAction() == MotionEvent.ACTION_UP) {
        arg0.setBackgroundResource(R.drawable.normal); //常态下的图片对应 normal
    }
    else if() // 这里还可以继续实现 MotionEvent.ACTION_MOVE 和
    MotionEvent.ACTION_CANCEL 等实现更多的特效
        return false;
    }
};

```

ImageButton 也相同

```

ImageButton.setOnTouchListener(new OnTouchListener(){
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        if(event.getAction() == MotionEvent.ACTION_DOWN){
            //更改为按下时的背景图片
            v.setBackgroundResource(R.drawable.pressed);
        }else if(event.getAction() == MotionEvent.ACTION_UP){
            //改为抬起时的图片
            v.setBackgroundResource(R.drawable.released);
        }
        return false;
    }
});

```

当然自己定义 xml 也很简单，处理下 selector 和 android:state_focused、android:state_pressed 即可，对于按键多了确实有必要定义一个 xml 文件，当然我们都是使用图片来实现的，考虑到拉伸需要考虑 9Patch 方法实现简单的无损拉伸方法

2. Xml 文件实现

一. ImageButton

定义：

.在 drawable 目录下添加新的 xml 文件 button_add_x.xml，也就是定义一些 selector 标签，并指定 Button 在各种状态下背景图片,如下代码：

```

<?xml version="1.0" encoding="UTF-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="false" android:drawable="@drawable/button_add" />
    <item android:state_pressed="true" android:drawable="@drawable/button_add_pressed" />
    <item android:state_focused="true" android:drawable="@drawable/button_add_pressed" />
    <item android:drawable="@drawable/button_add" />
</selector>

```



使用:

<ImageButton

```

        android:id="@+id/ImageButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#00000000"
        android:src="@drawable/button_add_x" >

```

</ImageButton>

二.Text View

改变字体颜色:和 Button 的区别是改变的是 textColor 属性, 而且 selector 文件定义在 color 目录下:

1. 在 layout 文件中指定 TextView 的 textColor 属性, 如
 android:textColor="@color/textview_color";

2. 在 color 目录下添加新的 xml 文件 textview_color.xml 并指定 TextView 在各种状态下的色值, 如下代码:

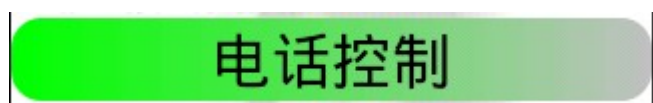
```

<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_selected="true" android:color="#FFF" />
    <item android:state_focused="true" android:color="#FFF" />
    <item android:state_pressed="true" android:color="#FFF" />
    <item android:color="#000" />
</selector>

```

5.2、Button美化案例☆

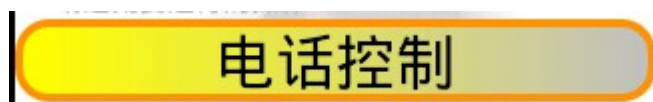
按钮常态:



选中按钮:



按下按钮:



背景配置文件:

button_bg.xml

```
<?xml version="1.0" encoding="UTF-8"?>

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

    <item
        android:state_pressed="true"
        android:drawable="@drawable/button_focusedandpressed" />

    <item
        android:state_focused="true"
        android:state_pressed="true"
        android:drawable="@drawable/button_focusedandpressed" />

    <item
        android:state_focused="false"
        android:state_pressed="true"
        android:drawable="@drawable/button_defocused" />

    <item
        android:state_focused="true"
        android:state_pressed="false"
        android:drawable="@drawable/button_focused" />

    <item
        android:state_focused="false"
        android:drawable="@drawable/button_defocused" />
</selector>
```

button_defocused.xml //实现渐变色的效果（焦点不在 button 上，且未按下）

```
<?xml version="1.0" encoding="UTF-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
<!-- 颜色渐变效果 -->
    <gradient
        android:startColor="#00FF00"
        android:endColor="#C0C0C0"
        android:paddingTop="5dp"
        android:angle="0" />
<!-- 按钮四个角的平滑度 -->
    <corners android:radius="15dp" />
<!-- 字距离按钮的四个边距 -->
    <padding
        android:left="5dp"
        android:top="5dp"
        android:right="5dp"
        android:bottom="5dp" />
<!-- 按钮的光环的宽度与颜色 -->
```



```

        <stroke
            android:width="2dp"
            android:color="#fad3cf" />
    </shape>

```

button_focused.xml (焦点在 button 上, 但没有按下)

```

<?xml version="1.0" encoding="UTF-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- 颜色渐变效果 -->
    <gradient
        android:startColor="#505050"
        android:endColor="#C0C0C0"
        android:paddingTop="5dp"
        android:angle="0" />

    <!-- 按钮四个角的平滑度 -->
    <corners android:radius="15dp" />

    <!-- 焦点在按钮上时光环的宽度与颜色 -->
    <stroke android:width="4dp" android:color="#FFF9200" />
</shape>

```

button_focusedandpressed.xml (按钮按下时, 或被点击时)

```

<?xml version="1.0" encoding="UTF-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- 颜色渐变效果 -->
    <gradient
        android:startColor="#FFFF00"
        android:endColor="#C0C0C0"
        android:paddingTop="5dp"
        android:angle="0" />

    <!-- 按钮四个角的平滑度 -->
    <corners android:radius="20dp" />

    <!-- 焦点在按钮上时光环的宽度与颜色 -->
    <stroke android:width="4dp" android:color="#FFF9200" />

</shape>

```

字体颜色配置文件: button_font.xml (尚不会用)

```

<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">

```



```

<item android:state_selected="true" android:color="#FFF" />
<item android:state_focused="true" android:color="#FFF" />
<item android:state_pressed="true" android:color="#FFF" />
<item android:color="#000" />
</selector>

```

用法;

<Button

```

    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/btn_ctrllist"
    android:textSize="30dip"
    android:id="@+id/button_cal"
    android:text=" 电话控制" />

```

5.3、ImageButton 按下时的动画效果

1. java 代码实现

```

imageButton.setOnTouchListener(new OnTouchListener(){
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        if(event.getAction() == MotionEvent.ACTION_DOWN){
            //更改为按下时的背景图片
            v.setBackgroundResource(R.drawable.pressed);
        }else if(event.getAction() == MotionEvent.ACTION_UP){
            //改为抬起时的图片
            v.setBackgroundResource(R.drawable.released);
        }
        return false;
    }
});
imageButton.setOnTouchListener(new OnTouchListener(){
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        if(event.getAction() == MotionEvent.ACTION_DOWN){
            //更改为按下时的背景图片
            v.setBackgroundResource(R.drawable.pressed);
        }else if(event.getAction() == MotionEvent.ACTION_UP){
            //改为抬起时的图片
            v.setBackgroundResource(R.drawable.released);
        }
        return false;
    }
});

```



```
    }
});
```

2. xml 里实现

```
<?xml version="1.0" encoding="UTF-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="false" android:drawable="@drawable/button_add" />
    <item android:state_pressed="true" android:drawable="@drawable/button_add_pressed" />
    <item android:state_focused="true" android:drawable="@drawable/button_add_pressed" />
</item android:drawable="@drawable/button_add" />
</selector>

<?xml version="1.0" encoding="UTF-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
<item android:state_pressed="false" android:drawable="@drawable/button_add" />

    <item android:state_pressed="true" android:drawable="@drawable/button_add_pressed" />
    <item android:state_focused="true" android:drawable="@drawable/button_add_pressed" />
    <item android:drawable="@drawable/button_add" />
</selector>
```

这个文件放在 drawable 目录下面。命名为 button_add_x.xml 使用的时候:

```
<ImageButton
    android:id="@+id/ImageButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#00000000"
    android:src="@drawable/button_add_x" >

</ImageButton>
<ImageButton
    android:id="@+id/ImageButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#00000000"
    android:src="@drawable/button_add_x" >

</ImageButton>
```

5.4、滚动条显示与隐藏

不活动时隐藏滚动条，滑动时显示滚动条

你可以在加载的时候让滚动条不显示:setVerticalScrollBarEnabled(false),活动事件里面让



setVerticalScrollBarEnabled(true).不知道是不是你要的效果

刚发现隐藏滚动条的实现方法，原文如下：

哈哈，让我发现了（不过 2.0 后才支持），fading 就是淡去的意思

```
void android.view.View.setScrollbarFadingEnabled(boolean fadeScrollbars)
```

```
public void setScrollbarFadingEnabled (boolean fadeScrollbars)
```

Since: API Level 5

Define whether scrollbars will fade when the view is not scrolling.

Parameters

fadeScrollbars wheter to enable fading

所以 setScrollbarFadingEnabled(true);就可以啦!!!!

当然，很明显不仅仅是 ListView 可以这样设置，继承 View 类的都可以啊！

5.5、ListView 与 ScrollView 解决办法

方法一：（重写ListView）

重构方法：

文件 1: AdapterForLinearLayout.java

```
package com.craining.book.Growth_Need.ListView_Self;
```

```
import java.util.List;
import java.util.Map;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.TextView;
```

```
/**
```

```
* About ListView ;
```

```
*/
```

```
public class AdapterForLinearLayout extends BaseAdapter {
    private LayoutInflater mInflater;
    private int resource;
    private List<? extends Map<String, ?>> data;
```




```
private String[] from;
private int[] to;

public AdapterForLinearLayout(Context context,
    List<? extends Map<String, ?>> data, int resource, String[] from, int[] to) {
    this.data = data;
    this.resource = resource;
    this.data = data;
    this.from = from;
    this.to = to;
    this.mInflater = (LayoutInflater) context
        .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
}

@Override
public int getCount() {

    return data.size();
}

@Override
public Object getItem(int position) {

    return data.get(position);
}

@Override
public String get(int position, Object key) {
    Map<String, ?> map = (Map<String, ?>) getItem(position);
    return map.get(key).toString();
}

@Override
public long getItemId(int position) {

    return position;
}

@Override
public View getView(int position, View convertView, ViewGroup parent) {
    convertView = mInflater.inflate(resource, null);
    Map<String, ?> item = data.get(position);
    int count = to.length;
    for (int i = 0; i < count; i++) {
```



```

        View v = convertView.findViewById(to[i]);
        bindView(v, item, from[i]);
    }
    convertView.setTag(position);
    return convertView;
}

/**
 * 绑定视图
 */
private void bindView(View view, Map<String, ?> item, String from) {
    Object data = item.get(from);
    if (view instanceof TextView) {
        ((TextView) view).setText(data == null ? "" : data.toString());
    }
}
}

```

文件 2: LinearLayoutForListView.java

```
package com.craining.book.Growth_Need.ListView_Self;
```

```
import android.content.Context;
import android.util.AttributeSet;
import android.view.View;
import android.widget.LinearLayout;
```

```

/**
 * About the ListView;
 */

public class LinearLayoutForListView extends LinearLayout{
    private AdapterForLinearLayout adapter;
    private OnClickListener onClickListener = null;

    /**
     * 绑定布局
     */
    public void bindLinearLayout() {
        int count = adapter.getCount();
        for (int i = 0; i < count; i++) {
            View v = adapter.getView(i, null, null);

            v.setOnClickListener(this.onClickListener);
            if (i == count - 1) {

```



```
        LinearLayout ly = (LinearLayout) v;
        ly.removeViewAt(2);
    }
    addView(v, i);
}
// Log.v("countTAG", "" + count);
}

public LinearLayoutForListView(Context context){
    super(context);
}

public LinearLayoutForListView(Context context, AttributeSet attrs) {
    super(context, attrs);
    // TODO Auto-generated constructor stub
}

/**
 * 获取 Adapter
 *
 * @return adapter
 */
public AdapterForLinearLayout getAdpater() {
    return adapter;
}

/**
 * 设置数据
 *
 * @param adpater
 */
public void setAdapter(AdapterForLinearLayout adpater) {
    this.adapter = adpater;
    bindLinearLayout();
}

/**
 * 获取点击事件
 *
 * @return
 */
public OnClickListener getOncklickListner() {
    return onClickListener;
}
```



```

/**
 * 设置点击事件
 *
 * @param onClickListener
 */
public void setOnClickListener(OnClickListener onClickListener) {
    this.setOnClickListener = onClickListener;
}
}

```

文件 3: self_listshow.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView android:id="@+id/TextView01"
        android:layout_marginLeft="10px"
        android:textColor="#FFFFFF"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </TextView>
    <TextView
        android:id="@+id/TextView02"
        android:layout_width="wrap_content"
        android:textColor="#FFFFFF"
        android:textAppearance="?android:attr/textAppearanceSmall"
        android:layout_marginLeft="10px"
        android:layout_height="wrap_content">
    </TextView>
    <View
        android:layout_height="1px" android:background="#FFFFFF"
        android:layout_width="fill_parent"></View>
</LinearLayout>

```

使用方法:

1. Xml 文件中: 添加布局:

```

<?xml version="1.0" encoding="UTF-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"

```



```

        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="@drawable/bg">
    <ScrollView
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:orientation="vertical"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content">
            <View
                android:layout_height="1px"
                android:background="#FFFFFF"
                android:layout_width="fill_parent">
            </View>

            <com.craining.book.Growth_Need.ListView_Self.LinearLayoutForListView
                android:orientation="vertical"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:id="@+id/selflistofdiarylist">
            </com.craining.book.Growth_Need.ListView_Self.LinearLayoutForListView>
            <View
                android:layout_height="1px"
                android:background="#FFFFFF"
                android:layout_marginBottom="100dip"
                android:layout_width="fill_parent">
            </View>
        </LinearLayout>
    </ScrollView>
</RelativeLayout>

```

.2。Java 代码:

声明:

```

private LinearLayoutForListView lv = null;
ArrayList<HashMap<String, Object>> list = new ArrayList<HashMap<String, Object>>();

```

Oncreate 中实现:

```

lv = (LinearLayoutForListView) findViewById(R.id.selflistofdiarylist);

```

UpdataDiaryAdapter();//获得列表内容

```

final AdapterForLinearLayout Layoutadpater = new AdapterForLinearLayout(
    this, list, R.layout.self_listshow,

```



```

        new String[] { "list_diary_date", "list_diary_title" },
        new int[] { R.id.TextView01, R.id.TextView02 } );
lv.setOnItemClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        getSelectedDate = Layoutadpater.get(Integer.parseInt(v.getTag().toString()),
"list_diary_date");
        setProgressBarIndeterminateVisibility( true );
        getSelectedDiaryInf( getSelectedDate );
    }
});
lv.setAdapter(Layoutadpater);

```

获得列表内容的方法： vector_DiaryDate 和 vector_DiaryTitle 要显示的条目，类型是 Vector

```

public void UpdataDiaryAdapter() {
    if( getAllDiaryInfo() ) {
        diaryCount = vector_DiaryDate.size();
        for (int i = diaryCount-1; i >= 0; i--) {
            String diaryDate = vector_DiaryDate.get(i);
            String diaryTitle = vector_DiaryTitle.get( i );
            HashMap<String, Object> map = new HashMap<String, Object>();
            map.put("list_diary_date", diaryDate);
            map.put("list_diary_title", diaryTitle);
            list.add( map );
        }
    }
}

```

方法二：

这几天一直被 listview 怎么合理的放进 scorllview 中的问题困扰，尝试过把 listview 放入 scorllview 中的朋友都知道，被放入的 listview 显示是有问题的，无论怎么设置 layout 都只显示大概 2 行的高度，看起来很郁闷，更别说美观了，后来上网查询了一下，解决方法有的是用 linearlayout 替换 listview，还有修改 onmeasure 的，我比较懒个人感觉很麻烦不喜欢，终于想出了一个还算和谐的解决方法：xml 中的 textlist 设置如下：

```

<?xml version="1.0" encoding="UTF-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:background="#44444444">
<ScrollView
    android:layout_width="fill_parent"

```



```

        android:layout_height="wrap_content">
<LinearLayout
    android:id="@+id/ll1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:scrollbars="vertical"
    android:orientation="vertical"
    android:paddingLeft="15dp"
    android:paddingRight="15dp"
    android:paddingTop="30dp"
    android:paddingBottom="30dp"
    android:background="#ff888888">
    <TextView
        android:text="あ"
        android:textColor="#ffeeeeee"
        android:textSize="18sp"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"></TextView>
    <ListView
        android:scrollbars="none"
        android:stackFromBottom="true"
        android:id="@+id/lv0"
        android:layout_width="fill_parent"
        android:layout_height="20dp"></ListView>
    </LinearLayout>
</ScrollView>
</LinearLayout>

```

其中的 textview 是我做的东西要用到的，和方法无关可以不看，然后就是在 java 中重新设置 listview 的高度了，目的是把 listview “撑”开：

```

LinearLayout.LayoutParams lp5 = new
LinearLayout.LayoutParams(LayoutParams.FILL_PARENT, listItem.size()*51-1);

```

其中第一个属性不必说了，第二个是为了计算 listview 要设置的总高度用的，51 是我事先设置好的一行的高度（50）+每行之间的间隔（1）而得来的，listItem.size()是我要显示的行数，用.setLayoutParams(lp5);来重新设置高度，其他别的设置跟以前一样，想要源码我整理完之后贴出来

5.6、3D魔方

```

import android.view.SurfaceView;
import android.view.SurfaceHolder;
import android.content.Context;
import android.util.AttributeSet;
import java.util.ArrayList;

```



```
import java.util.concurrent.Semaphore;

import javax.microedition.khronos.egl.EGL10;
import javax.microedition.khronos.egl.EGL11;
import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.egl.EGLContext;
import javax.microedition.khronos.egl.EGLDisplay;
import javax.microedition.khronos.egl.EGLSurface;
import javax.microedition.khronos.opengles.GL;
import javax.microedition.khronos.opengles.GL10;

public class View3D extends SurfaceView implements SurfaceHolder.Callback {

    private static final Semaphore sEglSemaphore = new Semaphore(1);
    private boolean mSizeChanged = true;

    private SurfaceHolder mHolder;
    private GLThread mGLThread;
    private GLWrapper mGLWrapper;

    public View3D(Context context) {
        super(context);
        init();
    }

    public View3D(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    private void init() {
        mHolder = getHolder();
        mHolder.addCallback(this);
        mHolder.setType(SurfaceHolder.SURFACE_TYPE_GPU);
    }

    public SurfaceHolder getSurfaceHolder() {
        return mHolder;
    }

    public void setGLWrapper(GLWrapper glWrapper) {
        mGLWrapper = glWrapper;
    }
}
```




```
public void setRenderer(Renderer renderer) {
    mGLThread = new GLThread(renderer);
    mGLThread.start();
}

public void surfaceCreated(SurfaceHolder holder) {
    mGLThread.surfaceCreated();
}

public void surfaceDestroyed(SurfaceHolder holder) {
    mGLThread.surfaceDestroyed();
}

public void surfaceChanged(SurfaceHolder holder,
                           int format, int w, int h) {
    mGLThread.onWindowResize(w, h);
}

public void onPause() {
    mGLThread.onPause();
}

public void onResume() {
    mGLThread.onResume();
}

@Override
public void onWindowFocusChanged(boolean hasFocus) {
    super.onWindowFocusChanged(hasFocus);
    mGLThread.onWindowFocusChanged(hasFocus);
}

public void queueEvent(Runnable r) {
    mGLThread.queueEvent(r);
}

@Override
protected void onDetachedFromWindow() {
    super.onDetachedFromWindow();
    mGLThread.requestExitAndWait();
}
```



```
public interface GLWrapper {
    GL wrap(GL gl);
}

public interface Renderer {

    int[] getConfigSpec();

    void surfaceCreated(GL10 gl);
    void sizeChanged(GL10 gl, int width, int height);
    void drawFrame(GL10 gl);
}

private class EglHelper {

    EGL10 mEgl;
    EGLDisplay mEglDisplay;
    EGLSurface mEglSurface;
    EGLConfig mEglConfig;
    EGLContext mEglContext;

    public EglHelper() {

    }

    public void start(int[] configSpec){

        mEgl = (EGL10) EGLContext.getEGL();
        mEglDisplay = mEgl.eglGetDisplay(EGL10.EGL_DEFAULT_DISPLAY);
        int[] version = new int[2];
        mEgl.eglInitialize(mEglDisplay, version);

        EGLConfig[] configs = new EGLConfig[1];
        int[] num_config = new int[1];
        mEgl.eglChooseConfig(mEglDisplay, configSpec, configs, 1,
            num_config);
        mEglConfig = configs[0];

        mEglContext = mEgl.eglCreateContext(mEglDisplay, mEglConfig,
            EGL10.EGL_NO_CONTEXT, null);

        mEglSurface = null;
    }
}
```



```
public GL createSurface(SurfaceHolder holder) {

    if (mEglSurface != null) {

        mEgl.eglMakeCurrent(mEglDisplay, EGL10.EGL_NO_SURFACE,
            EGL10.EGL_NO_SURFACE, EGL10.EGL_NO_CONTEXT);
        mEgl.eglDestroySurface(mEglDisplay, mEglSurface);
    }

    mEglSurface = mEgl.eglCreateWindowSurface(mEglDisplay,
        mEglConfig, holder, null);

    mEgl.eglMakeCurrent(mEglDisplay, mEglSurface, mEglSurface,
        mEglContext);

    GL gl = mEglContext.getGL();
    if (mGLWrapper != null) {
        gl = mGLWrapper.wrap(gl);
    }
    return gl;
}

public boolean swap() {
    mEgl.eglSwapBuffers(mEglDisplay, mEglSurface);
    return mEgl.eglGetError() != EGL11.EGL_CONTEXT_LOST;
}

public void finish() {
    if (mEglSurface != null) {
        mEgl.eglMakeCurrent(mEglDisplay, EGL10.EGL_NO_SURFACE,
            EGL10.EGL_NO_SURFACE,
            EGL10.EGL_NO_CONTEXT);
        mEgl.eglDestroySurface(mEglDisplay, mEglSurface);
        mEglSurface = null;
    }
    if (mEglContext != null) {
        mEgl.eglDestroyContext(mEglDisplay, mEglContext);
        mEglContext = null;
    }
    if (mEglDisplay != null) {
        mEgl.eglTerminate(mEglDisplay);
        mEglDisplay = null;
    }
}
```



```
    }  
}  
  
}  
  
class GLThread extends Thread {  
  
    private boolean mDone;  
    private boolean mPaused;  
    private boolean mHasFocus;  
    private boolean mHasSurface;  
    private boolean mContextLost;  
    private int mWidth;  
    private int mHeight;  
    private Renderer mRenderer;  
    private ArrayList<Runnable>  
        mEventQueue = new ArrayList<Runnable>();  
    private EglHelper mEglHelper;  
  
    GLThread(Renderer renderer) {  
        super();  
        mDone = false;  
        mWidth = 0;  
        mHeight = 0;  
        mRenderer = renderer;  
        setName("GLThread");  
    }  
  
    @Override  
    public void run() {  
  
        try {  
            try {  
                sEglSemaphore.acquire();  
            } catch (InterruptedException e) {  
                return;  
            }  
            guardedRun();  
        } catch (InterruptedException e) {  
  
        } finally {  
            sEglSemaphore.release();  
        }  
    }  
}
```



```
}

private void guardedRun() throws InterruptedException {
    mEglHelper = new EglHelper();
    int[] configSpec = mRenderer.getConfigSpec();
    mEglHelper.start(configSpec);

    GL10 gl = null;
    boolean tellRendererSurfaceCreated = true;
    boolean tellRendererSurfaceChanged = true;

    while (!mDone) {

        int w, h;
        boolean changed;
        boolean needStart = false;
        synchronized (this) {
            Runnable r;
            while ((r = getEvent()) != null) {
                r.run();
            }
            if (mPaused) {
                mEglHelper.finish();
                needStart = true;
            }
            if (needToWait()) {
                while (needToWait()) {
                    wait();
                }
            }
            if (mDone) {
                break;
            }
            changed = mSizeChanged;
            w = mWidth;
            h = mHeight;
            mSizeChanged = false;
        }
        if (needStart) {
            mEglHelper.start(configSpec);
            tellRendererSurfaceCreated = true;
            changed = true;
        }
        if (changed) {
```



```
        gl = (GL10) mEglHelper.createSurface(mHolder);
        tellRenderSurfaceChanged = true;
    }
    if (tellRenderSurfaceCreated) {
        mRenderer.surfaceCreated(gl);
        tellRenderSurfaceCreated = false;
    }
    if (tellRenderSurfaceChanged) {
        mRenderer.sizeChanged(gl, w, h);
        tellRenderSurfaceChanged = false;
    }
    if ((w > 0) && (h > 0)) {

        mRenderer.drawFrame(gl);
        mEglHelper.swap();
    }
}
mEglHelper.finish();
}

private boolean needToWait() {
    return (mPaused || (! mHasFocus) || (! mHasSurface) || mContextLost)
        && (! mDone);
}

public void surfaceCreated() {
    synchronized(this) {
        mHasSurface = true;
        mContextLost = false;
        notify();
    }
}

public void surfaceDestroyed() {
    synchronized(this) {
        mHasSurface = false;
        notify();
    }
}

public void onPause() {
    synchronized (this) {
        mPaused = true;
    }
}
```



```
}

public void onResume() {
    synchronized (this) {
        mPaused = false;
        notify();
    }
}

public void onWindowFocusChanged(boolean hasFocus) {
    synchronized (this) {
        mHasFocus = hasFocus;
        if (mHasFocus == true) {
            notify();
        }
    }
}

public void onWindowResize(int w, int h) {
    synchronized (this) {
        mWidth = w;
        mHeight = h;
        mSizeChanged = true;
    }
}

public void requestExitAndWait() {
    synchronized(this) {
        mDone = true;
        notify();
    }
    try {
        join();
    } catch (InterruptedException ex) {
        Thread.currentThread().interrupt();
    }
}

public void queueEvent(Runnable r) {
    synchronized(this) {
        mEventQueue.add(r);
    }
}
```



```
private Runnable getEvent() {
    synchronized(this) {
        if (mEventQueue.size() > 0) {
            return mEventQueue.remove(0);
        }
    }
    return null;
}
}
```

6、Android UI 动画

6.1、四种 2D动画

先列出四种动画实现的 xml 代码:

6.1.1、透明度控制动画效果 alpha

文件名: `my_alpha_action.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android" >
<alpha
    android:fromAlpha="0.1"
    android:toAlpha="1.0"
    android:duration="3000"
/>
```

<!-- 透明度控制动画效果 alpha

浮点型值:

fromAlpha 属性为动画起始时透明度

toAlpha 属性为动画结束时透明度

说明:

0.0 表示完全透明

1.0 表示完全不透明

以上值取 0.0-1.0 之间的 float 数据类型的数字

长整型值:



duration 属性为动画持续时间

说明:

时间以毫秒为单位

-->

</set>

6.1.2、旋转动画效果 rotate

文件名: my_rotate_action.xml

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

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
```

```
<rotate
```

```
    android:interpolator="@android:anim/accelerate_decelerate_interpolator"
```

```
    android:fromDegrees="0"
```

```
    android:toDegrees="+350"
```

```
    android:pivotX="50%"
```

```
    android:pivotY="50%"
```

```
    android:duration="3000" />
```

```
<!-- rotate 旋转动画效果
```

属性: interpolator 指定一个动画的插入器

在我试验过程中, 使用 android.res.anim 中的资源时候发现有三种动画插入器

accelerate_decelerate_interpolator 加速-减速 动画插入器

accelerate_interpolator 加速-动画插入器

decelerate_interpolator 减速- 动画插入器

其他的属于特定的动画效果

浮点数型值:

fromDegrees 属性为动画起始时物件的角度

toDegrees 属性为动画结束时物件旋转的角度 可以大于 360 度

说明:

当角度为负数——表示逆时针旋转

当角度为正数——表示顺时针旋转

(负数 from——to 正数:顺时针旋转)

(负数 from——to 负数:逆时针旋转)

(正数 from——to 正数:顺时针旋转)

(正数 from——to 负数:逆时针旋转)

pivotX 属性为动画相对于物件的 X 坐标的开始位置

pivotY 属性为动画相对于物件的 Y 坐标的开始位置



说明:

以上两个属性值 从 0%-100%中取值
50%为物件的 X 或 Y 方向坐标上的中点位置

长整型值:

duration 属性为动画持续时间

说明:

时间以毫秒为单位

-->

</set>

6.1.3、尺寸伸缩动画效果 scale

文件名: my_scale_action.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <scale
        android:interpolator="@android:anim/accelerate_decelerate_interpolator"

        android:fromXScale="0.0"
        android:toXScale="1.4"

        android:fromYScale="0.0"
        android:toYScale="1.4"

        android:pivotX="50%"
        android:pivotY="50%"

        android:fillAfter="false"
        android:duration="700" />
</set>
```

<!-- 尺寸伸缩动画效果 scale

属性: interpolator 指定一个动画的插入器

在我试验过程中, 使用 android.res.anim 中的资源时候发现有三种动画插入器

accelerate_decelerate_interpolator 加速-减速 动画插入器

accelerate_interpolator 加速-动画插入器

decelerate_interpolator 减速- 动画插入器

其他的属于特定的动画效果



浮点型值:

fromXScale 属性为动画起始时 X 坐标上的伸缩尺寸

toXScale 属性为动画结束时 X 坐标上的伸缩尺寸

fromYScale 属性为动画起始时 Y 坐标上的伸缩尺寸

toYScale 属性为动画结束时 Y 坐标上的伸缩尺寸

说明:

以上四种属性值

0.0 表示收缩到没有

1.0 表示正常无伸缩

值小于 1.0 表示收缩

值大于 1.0 表示放大

pivotX 属性为动画相对于物件的 X 坐标的开始位置

pivotY 属性为动画相对于物件的 Y 坐标的开始位置

说明:

以上两个属性值 从 0%-100%中取值

50%为物件的 X 或 Y 方向坐标上的中点位置

长整型值:

duration 属性为动画持续时间

说明:

时间以毫秒为单位

布尔型值:

fillAfter 属性 当设置为 true , 该动画转化在动画结束后被应用

-->

6.1.4、位置转移动画效果 translate

文件名: my_translate_action.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
<translate
android:fromXDelta="30"
android:toXDelta="-80"
android:fromYDelta="30"
android:toYDelta="300"
android:duration="800"
```



```
/>
```

```
<!-- translate 位置转移动画效果
```

整型值:

fromXDelta 属性为动画起始时 X 坐标上的位置

toXDelta 属性为动画结束时 X 坐标上的位置

fromYDelta 属性为动画起始时 Y 坐标上的位置

toYDelta 属性为动画结束时 Y 坐标上的位置

注意:

没有指定 fromXType toXType fromYType toYType 时候, 默认是以自己为相对参照

物

长整型值:

duration 属性为动画持续时间

说明:

时间以毫秒为单位

```
-->
```

```
</set>
```

6.1.5、四种动画效果的调用

在这里, 我将每种动画分别应用于四个按钮为例:

(1) main.xml 代码如下:(声明四个按钮控件)

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/widget32"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
    <TextView
        android:id="@+id/widget29"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        android:layout_x="0px"
        android:layout_y="0px"
    >
    </TextView>
    <Button
        android:id="@+id/button_Alpha"
        android:layout_width="150px"
```



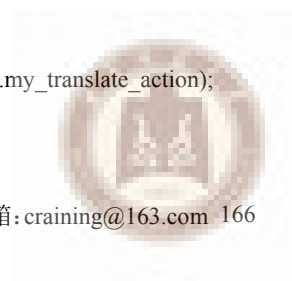
```
android:layout_height="150px"
android:text="Alpha 动画"
android:textSize="50px"
android:layout_x="0px"
android:layout_y="30px">
</Button>
<Button
android:id="@+id/button_Scale"
android:layout_width="150px"
android:layout_height="150px"
android:text="Scale 动画"
android:textSize="50px"
android:layout_x="0px"
android:layout_y="180px">
</Button>
<Button
android:layout_width="150px"
android:layout_height="150px"
android:text="Translate 动画"
android:layout_x="161px"
android:layout_y="30px"
android:textSize="30px"
android:id="@+id/button_Translate">
</Button>
<Button
android:id="@+id/button_Rotate"
android:layout_width="150px"
android:layout_height="150px"
android:text="Rotate 动画"
android:layout_y="180px"
android:layout_x="161px"
android:textSize="44px">
</Button>
</AbsoluteLayout>
```

(2) java 代码

```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
```



```
public class myActionAnimation extends Activity implements OnClickListener {  
    /** Called when the activity is first created. */  
    private Button button_alpha;  
    private Button button_scale;  
    private Button button_translate;  
    private Button button_rotate;  
    private Animation myAnimation_Alpha;  
    private Animation myAnimation_Scale;  
    private Animation myAnimation_Translate;  
    private Animation myAnimation_Rotate;  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
  
        button_alpha = (Button) findViewById(R.id.button_Alpha);  
        button_alpha.setOnClickListener(this);  
  
        button_scale = (Button) findViewById(R.id.button_Scale);  
        button_scale.setOnClickListener(this);  
  
        button_translate = (Button) findViewById(R.id.button_Translate);  
        button_translate.setOnClickListener(this);  
  
        button_rotate = (Button) findViewById(R.id.button_Rotate);  
        button_rotate.setOnClickListener(this);  
    }  
    public void onClick(View button) {  
        // TODO Auto-generated method stub  
        switch (button.getId()) {  
            case R.id.button_Alpha: {  
                myAnimation_Alpha = AnimationUtils.loadAnimation(this,R.layout.my_alpha_action);  
                button_alpha.startAnimation(myAnimation_Alpha);  
            }  
            break;  
            case R.id.button_Scale: {  
                myAnimation_Scale= AnimationUtils.loadAnimation(this,R.layout.my_scale_action);  
                button_scale.startAnimation(myAnimation_Scale);  
            }  
            break;  
            case R.id.button_Translate: {  
                myAnimation_Translate= AnimationUtils.loadAnimation(this,R.layout.my_translate_action);  
                button_translate.startAnimation(myAnimation_Translate);  
            }  
        }  
    }  
}
```



```

    }
    break;
case R.id.button_Rotate: {
    myAnimation_Rotate= AnimationUtils.loadAnimation(this,R.layout.my_rotate_action);
    button_rotate.startAnimation(myAnimation_Rotate);
}
break;

default:
    break;
}
}
}
}

```

效果图:



7、异步调用

android 有两种方式处理线程:

1. 比较耗时的操作放在后台服务, 通过通知机制通知用户使用的活动 (activity);
2. 在后台线程中处理耗时的操作

开辟一个线程:

```

//开启线程等待
Thread background = new Thread(new Runnable() {
    @Override
    public void run() {

```



```

        try {
            Thread.sleep(5000);
            stopService(new Intent(UsefullVerbs.PACKAGE_NAME +
".BackService")));
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}));

background.start();

```

Thread:

```

import java.util.Calendar;
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.widget.AnalogClock;
import android.widget.TextView;
/*
 * 通过产生 Thread 对象，在线程内同步调用 System.currentTimeMillis()
 * 获得系统时间，并且通过 Message 对象来通知 Handler 对象，Handler 则扮演
 * 联系 Activity 与 Thread 的桥梁，在接收 Message 对象后，将时间变量
 * 的值，显示在 TextView 当中，产生数字时钟的外观和功能
 */
public class analot extends Activity {
    /** Called when the activity is first created. */
    protected static final int GUINOTIFUER=0x1234;
    private TextView mTextView;
    public AnalogClock mAnalogClock;
    public Calendar mCalendar;
    public int mMinutes;
    public int mHour;
    public Handler mHandler;
    private Thread mClockThread;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mTextView = (TextView)findViewById(R.id.txt);
        mAnalogClock = (AnalogClock)findViewById(R.id.myAnalogClock);
    }
}

```




```

mHandler = new Handler(){
                                //处理消息队列中的消息

    @Override
    public void handleMessage(Message msg) {
        // TODO Auto-generated method stub
        switch(msg.what){
            case analot.GUINOTIFUER:
                                //进行显示
                mTextView.setText(mHour + " : " + mMinutes );
                break;
        }
        super.handleMessage(msg);
    }

};

mClockThread = new loopThread();
//启动 Thread （第一次）
mClockThread.start();
}

//Thread 中是不可以直接和控件进行通信的但是 Handler 可以
class loopThread extends Thread{

    @Override
    public void run() {
        // TODO Auto-generated method stub
        super.run();
        try{
            do{
                long time = System.currentTimeMillis();
                final Calendar mCalendar = Calendar.getInstance();
                mCalendar.setTimeInMillis(time);
                mHour = mCalendar.get(Calendar.HOUR);
                mMinutes = mCalendar.get(Calendar.MINUTE);
                Thread.sleep(1000);
                Message m = new Message();
                m.what = analot.GUINOTIFUER;
                analot.this.mHandler.sendMessage(m);
            } while (analot.loopThread.interrupted() == false);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
}

```



```
}
```

Handler

1 声明:

```
Handler hd1 = new Handler();
```

2 开启:

```
hd1.postDelayed(mTasks, delay);
```

3 添加方法

```
/** 速度控制参数(单位毫秒) */  
private int delay = 6000;  
/**  
 * 控制速度  
 */  
private Runnable mTasks = new Runnable() {  
    public void run() {  
        log();  
        hd1.postDelayed(mTasks, delay);  
    }  
};
```

其它:

```
private static final int EVENT_TIME_TO_CHANGE_IMAGE = 100;
```

```
android.os.Message message = mHandler.obtainMessage();  
message.what = EVENT_TIME_TO_CHANGE_IMAGE;  
mHandler.sendMessage(message);
```

```
private Handler mHandler = new Handler() {  
  
    public void handleMessage(android.os.Message msg) {  
        switch (msg.what) {  
            case EVENT_TIME_TO_CHANGE_IMAGE:  
  
                break;  
        }  
    }  
};
```



```
    }
};
```

使用 Handler

创建后台线程最友好的办法是创建一个 Handler 子类的实例。只需一个 Handler 对应一个 Activity。自定义的后台线程可与 Handler 通信，Handler 将与 UI 线程一起工作。

和 Handler 通信，需要两个选项，message 和 runnable 对象。

Message

发送 Message 到 Handler，第一步调用 `obtainMessage()`，从池中得到 Message 对象。

然后，可通过消息队列将 Message 发送给 Handler，通过 `sendMessage...`() 方法族：

- `sendMessage()` 立即发送 Message 到消息队列
- `sendMessageAtFrontOfQueue()` 立即发送 Message 到队列，而且是放在队列的最前面
- `sendMessageAtTime()` 设置时间，发送 Message 到队列
- `sendMessageDelayed()` 在延时若干毫秒后，发送 Message 到队列

为了处理 Message，Handler 需要实现 `handleMessage()`，当 Message 出现在队列中时，会调用 `handleMessage()` 方法。另外，Handler 可在需要时更新 UI。

以下示例演示一个进度条，每隔 1 秒钟增加 1/20 个单位。



布局文件：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
```

作者: craining (曲阜师范大学) 个人主页: <http://craining.blog.163.com/> 邮箱: craining@163.com 171



```
<ProgressBar android:id="@+id/progress"
    style="?android:attr/progressBarStyleHorizontal" android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
</LinearLayout>
```

java 代码:

```
package com.easymorse.thread;

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.widget.ProgressBar;

public class ShowThread extends Activity {

    ProgressBar bar;

    Handler handler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
            bar.incrementProgressBy(5);
        }
    };

    boolean isRunning = false;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        bar = (ProgressBar) findViewById(R.id.progress);
    }

    @Override
    protected void onStart() {
        super.onStart();
        bar.setProgress(0);

        Thread background = new Thread(new Runnable() {
            @Override
```



```
        public void run() {
            for (int i = 0; i < 20 && isRunning; i++) {
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                }
                handler.sendMessage(handler.obtainMessage());
            }
        }
    });

    isRunning = true;
    background.start();
}

@Override
protected void onStop() {
    super.onStop();
    isRunning = false;
}
}
```

Timer

```
isRunning = true;

timer = new Timer();
timer.schedule(new TimerTask() {

    @Override
    public void run() {
        // TODO Auto-generated method stub
        while(isRunning){
            ?toast.show();
        }
    }

}, 10);
```



Android 界面刷新

Android 提供了 `Invalidate` 方法实现界面刷新，但是 `Invalidate` 不能直接在线程中调用，因为他是违背了单线程模型：Android UI 操作并不是线程安全的，并且这些操作必须在 UI 线程中调用。

Android 程序中可以使用的界面刷新方法有两种，分别是利用 `Handler` 和利用 `postInvalidate()` 来实现在线程中刷新界面。

利用 `Handler` 刷新界面

实例化一个 `Handler` 对象，并重写 `handleMessage` 方法调用 `invalidate()` 实现界面刷新；而在线程中通过 `sendMessage` 发送界面更新消息。

```
// 在 onCreate() 中开启线程
new Thread(new GameThread()).start();
// 实例化一个 handler
Handler
myHandler
= new Handler()
{
// 接收到消息后处理
public void handleMessage(Message msg)
{
switch (msg.what)
{
case Activity01.REFRESH:
mGameView.invalidate();
// 刷新界面
break;
}
super.handleMessage(msg);
}
};
class GameThread implements Runnable
{
public void run()
{
while (!Thread.currentThread().isInterrupted())
{
Message message = new Message();
message.what = Activity01.REFRESH;
// 发送消息
Activity01.this.myHandler.sendMessage(message);
try
```



```
{
Thread.sleep(100);
}
catch (InterruptedException e)
{
Thread.currentThread().interrupt();
}
}
}
```

使用 `postInvalidate()` 刷新界面

使用 `postInvalidate` 则比较简单，不需要 `handler`，直接在线程中调用 `postInvalidate` 即可。

```
class GameThread implements Runnable
{
public void run()
{
while (!Thread.currentThread().isInterrupted())
{
try
{
Thread.sleep(100);
}
catch (InterruptedException e)
{
Thread.currentThread().interrupt();
}
//使用 postInvalidate 可以直接在线程中更新界面
mGameView.postInvalidate();
}
}
}
```

Message Handler

用法:

首先声明消息和监听句柄:

```
private static final int EMAIL_SEND = 1000;
private Handler oneHandler = new Handler();
```

然后定义消息监听函数:



```

private class oneHandler extends Handler {
    @Override
    public void handleMessage(Message msg) {
        switch (msg.what) {
            case EMAIL_SEND: {
                //这里添加出发消息时的操作:

                break;
            }
            default:
                break;
        }
    }
}

```

最后, 触发消息:

```
oneHandler.sendMessageDelayed(EMAIL_SEND, 0);
```

当然可以监听多个消息, 不过要注意, 在声明消息时, 后面的值不要重复!

或:

最常见的例子就是我们在更新 UI 时, 由于 Android UI 操作并不是线程安全的并且这些操作必须在 UI 线程中执行。所以我们需要使用利用 Handler 来实现 UI 线程的更新的。(当然 Handler 的用处也不仅限于此)。下面是代码片段

//处理消息

```

Handler myHandler = new Handler() {
    public void handleMessage(Message msg) {
        switch (msg.what) {
            case 100:
                //更新线程
                break;
        }
        super.handleMessage(msg);
    }
};

```

//发送消息

```

Message message = new Message();
message.what = 100;

```




```
myHandler.sendMessage(message);
```

线程与子线程调用MessageHandler

在一个 Android 程序开始运行的时候，会单独启动一个 Process。默认的情况下，所有这个程序中的 Activity 或者 Service（Service 和 Activity 只是 Android 提供的 Components 中的两种，除此之外还有 Content Provider 和 Broadcast Receiver）都会跑在这个 Process。

一个 Android 程序默认情况下也只有一个 Process，但一个 Process 下却可以有多个 Thread。

在这么多 Thread 当中，有一个 Thread，我们称之为 UI Thread。UI Thread 在 Android 程序运行的时候就被创建，是一个 Process 当中的主线程 Main Thread，主要是负责控制 UI 界面的显示、更新和控件交互。在 Android 程序创建之初，一个 Process 呈现的是单线程模型，所有的任务都在一个线程中运行。因此，我们认为，UI Thread 所执行的每一个函数，所花费的时间都应该是越短越好。而其他比较费时的任务（访问网络，下载数据，查询数据库等），都应该交由子线程去执行，以免阻塞主线程。

那么，UI Thread 如何和其他 Thread 一起工作呢？常用方法是：

诞生一个主线程的 Handler 物件，当做 Listener 去让子线程能将讯息 Push 到主线程的 Message Queue 里，以便触发主线程的 handleMessage（）函数，让主线程知道子线程的状态，并在主线程更新 UI。

Messagehandler实例

例如，在子线程的状态发生变化时，我们需要更新 UI。如果在子线程中直接更新 UI，通常会抛出下面的异常：

```
11-07 13:33:04.393: ERROR/JavaBinder(1029):android.view.ViewRoot$CalledFromWrongThreadException:Only the original thread that created a view hierarchy can touch its views.
```

意思是，无法在子线程中更新 UI。为此，我们需要通过 Handler 物件，通知主线程 Ui Thread 来更新界面。

如下，首先创建一个 Handler，来监听 Message 的事件：（以两个事件为例）

```
private final int UPDATE_UIONE= 1;
private final int UPDATE_UITWO = 2;
```



```
private Handler mHandler = new MainHandler();
```

```
private class MainHandler extends Handler {
```

```
    @Override
```

```
        public void handleMessage(Message msg) {
```

```
            switch (msg.what) {
```

```
                case UPDATE_UIONE: {
```

//这里可以放很多东西, 比如创建某个进度框线程, 执行完成需要取消进度框的显示, 可以发送一个消息, 然后这里就监听到了, 在这里将进度框取消掉, 还可以dispalyToast, 弹出对话框等等.....而这些在子线程中是无法使用的

```
                Log.i("TTSDeamon", "UPDATE_UIONE");
```

```
                showTextView.setText(editText.getText().toString());
```

```
                ShowAnimation();
```

```
                break;
```

```
            }
```

```
            case UPDATE_UITWO: {
```

```
                //执行某些命令
```

```
                break;
```

```
            }
```

```
            default:
```

```
                break;
```

```
        }
```

```
    }
```

```
};
```

或者

```
private Handler mHandler = new Handler(){
```

```
    @Override
```

```
        public void handleMessage(Message msg) {
```

```
            switch (msg.what) {
```

```
                case UPDATE_UIONE: {
```

```
                Log.i("TTSDeamon", "UPDATE_UIONE");
```

```
                showTextView.setText(editText.getText().toString());
```

```
                ShowAnimation();
```

```
                break;
```

```
            }
```

```
            case UPDATE_UITWO: {
```

```
                //执行某些命令
```

```
                break;
```

```
            }
```



```

        default:
            break;
    }
}
};

```

当子线程的状态发生变化，则在子线程中发出 **Message**，通知更新 UI。不同消息响应不同操作。

```
mHandler.sendMessageDelayed(UPDATE_UIONE, 0);
```

```
mHandler.sendMessageDelayed(UPDATE_UITWO, 0);
```

在我们的程序中，很多 **Callback** 方法有时候并不是运行在多线程当中的，所以如果在 **Callback** 方法中更新 UI 失败，也可以采用上面的方法。

8、数据存储与读取

在 Android 开发中我们会接触到四种数据存储方式，每种存储方式都各有不同；以下我分别列举了 Android 开发中的不同存储方式的特点

1. Preferences

Preferences 是一个较轻量级的存储数据的方法，具体使用方法：

在 A 中保存值：

```

SharedPreferences.Editor sharedata = getSharedPreferences("data", 0).edit();
    sharedata.putString("name", "shenrenkui");
    sharedata.commit();

```

在 B 中取值：

```

SharedPreferences sharedata = getSharedPreferences("data", 0);
String data = sharedata.getString("name", null);
Log.i(TAG, "data="+data);

```

注意，Context.getSharedPreferences(String name, int type) 的参数更我们在创建数据的时候的数据权限属性是一样的，存储和取值的过程这有点像 HashMap 但是比 HashMap 更具人性化，getXXX(Object key, Object defaultReturnValue)，第二个参数是当你所要的 key 对应没有时候返回的值。这就省去了很多逻辑判断。。。。



2. Files

在 Android 上面没有的 File 就是 J2se 中的纯种 File 了，可见功能之强大，这里就算是走马观花地严重路过了。

//创建文件

```
file = new File(FILE_PATH , FILE_NAME);
file.createNewFile();

//打开文件 file 的 OutputStream
out = new FileOutputStream(file);
String infoToWrite = "纸上得来终觉浅，绝知此事要躬行";
//将字符串转换成 byte 数组写入文件
out.write(infoToWrite.getBytes());
//关闭文件 file 的 OutputStream
out.close();

//打开文件 file 的 InputStream
in = new FileInputStream(file);
//将文件内容全部读入到 byte 数组
int length = (int)file.length();
byte[] temp = new byte[length];
in.read(temp, 0, length);
//将 byte 数组用 UTF-8 编码并存入 display 字符串中
display = EncodingUtils.getString(temp, TEXT_ENCODING);
//关闭文件 file 的 InputStream
in.close();
} catch (IOException e) {
//将出错信息打印到 Logcat
Log.e(TAG, e.toString());
this.finish();
}
```

//从资源读取

InputStream is=getResources().getRawResource(R.raw.文件名)

3. Databases

Android 内嵌了功能比其他手机操作系统强大的关系型数据库 sqlite3, 我们在大学时候学的 SQL 语句基本都可以使用，我们自己创建的数据可以用 adb shell 来操作。具体路径是 /data/data/package_name/databases。如，这里演示一下进入 com.android.providers.media 包下面的操作。



- 1, adb shell
- 2, cd /data/data/com.android.providers.media/databases
- 3, ls(查看 com.android.providers.media 下面的数据库)
- 4, sqlite3 internal.db
- 5, .help---看看如何操作
- 6, .table 列出 internal 数据中的表
- 7, select * from albums;

```

DatabaseHelper mOpenHelper;
private static final String DATABASE_NAME = "dbForTest.db";
private static final int DATABASE_VERSION = 1;
private static final String TABLE_NAME = "diary";
private static final String TITLE = "title";
private static final String BODY = "body";
private static class DatabaseHelper extends SQLiteOpenHelper {
    DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        String sql = "CREATE TABLE " + TABLE_NAME + " (" + TITLE
            + " text not null, " + BODY + " text not null " + ")";
        Log.i("haiyang:createDB=", sql);
        db.execSQL(sql);
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    }
}
/*
 * 重新建立数据表
 */
private void CreateTable() {
    SQLiteDatabase db = mOpenHelper.getWritableDatabase();
    String sql = "CREATE TABLE " + TABLE_NAME + " (" + TITLE
        + " text not null, " + BODY + " text not null " + ")";
    Log.i("haiyang:createDB=", sql);
    try {
        db.execSQL("DROP TABLE IF EXISTS diary");
        db.execSQL(sql);
        setTitle("数据表成功重建");
    } catch (SQLException e) {
        setTitle("数据表重建错误");
    }
}

```



```
    }
}
/*
 * 删除数据表
 */
private void dropTable() {
    SQLiteDatabase db = mOpenHelper.getWritableDatabase();
    String sql = "drop table " + TABLE_NAME;
    try {
        db.execSQL(sql);
        setTitle("数据表成功删除: " + sql);
    } catch (SQLException e) {
        setTitle("数据表删除错误");
    }
}
/*
 * 插入两条数据
 */
private void insertItem() {
    SQLiteDatabase db = mOpenHelper.getWritableDatabase();
    String sql1 = "insert into " + TABLE_NAME + " (" + TITLE + ", " + BODY
        + ") values(' haiyang', ' android 的发展真是迅速啊');";
    String sql2 = "insert into " + TABLE_NAME + " (" + TITLE + ", " + BODY
        + ") values(' icesky', ' android 的发展真是迅速啊');";
    try {
        Log.i("haiyang:sql1=", sql1);
        Log.i("haiyang:sql2=", sql2);
        db.execSQL(sql1);
        db.execSQL(sql2);
        setTitle("插入两条数据成功");
    } catch (SQLException e) {
        setTitle("插入两条数据失败");
    }
}
/*
 * 删除其中的一条数据
 */
private void deleteItem() {
    try {
        SQLiteDatabase db = mOpenHelper.getWritableDatabase();
        db.delete(TABLE_NAME, " title = ' haiyang'", null);
        setTitle("删除 title 为 haiyang 的一条记录");
    } catch (SQLException e) {
    }
}
```



```

}
/*
 * 在屏幕的 title 区域显示当前数据表当中的数据的条数。
 */
private void showItems() {
    SQLiteDatabase db = mOpenHelper.getReadableDatabase();
    String col[] = { TITLE, BODY };
    Cursor cur = db.query(TABLE_NAME, col, null, null, null, null, null);
    Integer num = cur.getCount();
    setTitle(Integer.toString(num) + " 条记录");
}

```

4. Network

这是借助 Internet 来存储我们要的数据，这是 CS 结构的存储方式，也是点一下名了。

如何使用 Content Provider

下边是用户经常接触到的几个典型 Content Provider 应用：

- * Content Provider Name : Intended Data
- * Browser : Browser bookmarks, Browser history, etc.
- * CallLog : Missed calls, Call details, etc.
- * Contacts : Contact details
- * MediaStore : Media files such as audio, Video and Images
- * Settings : Device Settings and Preferences

调用 Content Provider 资源的标准 URI 结构：

<standard_prefix>://<authority>/<data_path>/<id>

例如：

- 1) 取得浏览器所有“书签”信息： content://browser/bookmarks
- 2) 取得系统通讯录中的信息： content://contacts/people（如果取得某一个特定通讯记录，在路径 URI 的末端指定一个 ID 号： content://contacts/people/5

简单的实例片段：

```

Uri allCalls = Uri.parse("content://call_log/calls");
Cursor c = managedQuery(allCalls, null, null, null, null);

```

5. ContentProvider

一、ContentProvider 简介

当应用继承 ContentProvider 类，并重写该类用于提供数据和存储数据的方法，就可以向其他应用共享其数据。虽然使用其他方法也可以对外共享数据，但数据访问方式会因数据存储的方式而不同，如：采用文件方式对外共享数据，需要进行文件操作读写数据；采用 sharedpreferences 共享数据，需要使用 sharedpreferences API 读写数据。而使用 ContentProvider 共享数据的好处是统一了数据访问方式。

二、Uri 类简介

Uri 代表了要操作的数据，Uri 主要包含了两部分信息：1. 需要操作的 ContentProvider，2. 对 ContentProvider 中的什么数据进行操作，一个 Uri 由以下几部分组成：

1. scheme: ContentProvider (内容提供者) 的 scheme 已经由 Android 所规定为: content://。

2. 主机名 (或 Authority): 用于唯一标识这个 ContentProvider, 外部调用者可以根据这个标识来找到它。

3. 路径 (path): 可以用来表示我们要操作的数据, 路径的构建应根据业务而定, 如下:

- 要操作 contact 表中 id 为 10 的记录, 可以构建这样的路径: /contact/10
- 要操作 contact 表中 id 为 10 的记录的 name 字段, contact/10/name
- 要操作 contact 表中的所有记录, 可以构建这样的路径: /contact

要操作的数据不一定来自数据库, 也可以是文件等他存储方式, 如下:

要操作 xml 文件中 contact 节点下的 name 节点, 可以构建这样的路径: /contact/name

如果要把一个字符串转换成 Uri, 可以使用 Uri 类中的 parse() 方法, 如下:

```
Uri uri = Uri.parse("content://com.changcheng.provider.contactprovider/contact")
```

三、UriMatcher、ContentUrist 和 ContentResolver 简介

因为 Uri 代表了要操作的数据, 所以我们很经常需要解析 Uri, 并从 Uri 中获取数据。Android 系统提供了两个用于操作 Uri 的工具类, 分别为 UriMatcher 和 ContentUrist。掌握它们的使用, 会便于我们的开发工作。

UriMatcher: 用于匹配 Uri, 它的用法如下:

1. 首先把你需要匹配 Uri 路径全部给注册上, 如下:

// 常量 UriMatcher.NO_MATCH 表示不匹配任何路径的返回码(-1)。

```
UriMatcher uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
```

// 如 果 match() 方 法 匹 配

content://com.changcheng.sqlite.provider.contactprovider/contact 路径, 返回匹配码为 1

```
uriMatcher.addURI("com.changcheng.sqlite.provider.contactprovider", "contact", 1);
```

添加需要匹配 uri, 如果匹配就会返回匹配码

// 如 果 match() 方 法 匹 配

content://com.changcheng.sqlite.provider.contactprovider/contact/230 路径, 返回匹配码为 2

```
uriMatcher.addURI("com.changcheng.sqlite.provider.contactprovider", "contact/#", 2);
```

// # 号为通配符

2. 注册完需要匹配的 Uri 后, 就可以使用 uriMatcher.match(uri) 方法对输入的 Uri 进行匹配, 如果匹配就返回匹配码, 匹配码是调用 addURI() 方法传入的第三个参数, 假设匹配 content://com.changcheng.sqlite.provider.contactprovider/contact 路径, 返回的匹配码为 1。

ContentUrist: 用于获取 Uri 路径后面的 ID 部分, 它有两个比较实用的方法:

- withAppendedId(uri, id) 用于为路径加上 ID 部分
- parseId(uri) 方法用于从路径中获取 ID 部分



ContentResolver: 当外部应用需要对 **ContentProvider** 中的数据进行添加、删除、修改和查询操作时, 可以使用 **ContentResolver** 类来完成, 要获取 **ContentResolver** 对象, 可以使用 **Activity** 提供的 **getContentResolver()** 方法。 **ContentResolver** 使用 **insert**、**delete**、**update**、**query** 方法, 来操作数据。

ContentProvider 示例程序

1. 为 SQLite 示例程序添加 ContentProvider 类

```
package com.changcheng.sqlite.provider;
import com.changcheng.sqlite.MyOpenHelper;
import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;

public class ContactContentProvider extends ContentProvider {
    // 通过 UriMatcher 匹配外部请求
    private static UriMatcher uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
    // 通过 openHelper 进行数据库读写
    private MyOpenHelper openHelper;
    // 匹配状态常量
    private static final int CONTACT_LIST = 1;
    private static final int CONTACT = 2;
    // 表名
    private static final String tableName = "contacts";
    // 添加 Uri
    static {
        uriMatcher.addURI("com.changcheng.sqlite.provider", "contact",
            CONTACT_LIST);
        uriMatcher.addURI("com.changcheng.sqlite.provider", "contact/#",
            CONTACT);
    }
    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        SQLiteDatabase db = this.openHelper.getWritableDatabase();
        int result;
        switch (uriMatcher.match(uri)) {
            case CONTACT_LIST:
                result = db.delete(tableName, selection, selectionArgs);
                break;
            case CONTACT:
                long id = ContentUris.parseId(uri);
```



```

        String where = "_id=" + id;
        if (selection != null && !"".equals(selection)) {
            where = where + " and " + selection;
        }
        result = db.delete(tableName, where, selectionArgs);
        break;
    default:
        throw new IllegalArgumentException("Uri IllegalArgument:" + uri);
    }
    return result;
}

@Override
public String getType(Uri uri) {
    switch (uriMatcher.match(uri)) {
        case CONTACT_LIST:// 集合类型必须在前面加上 vnd.android.cursor.dir/
            return "vnd.android.cursor.dir/contactlist";
        case CONTACT:// 非集合类型必须在前面加上 vnd.android.cursor.item/
            return "vnd.android.cursor.item/contact";
        default:
            throw new IllegalArgumentException("Uri IllegalArgument:" + uri);
    }
}

@Override
public Uri insert(Uri uri, ContentValues values) {
    SQLiteDatabase db = this.openHelper.getWritableDatabase();
    long id;
    switch (uriMatcher.match(uri)) {
        case CONTACT_LIST:
            // 因为后台需要生成 SQL 语句，当 values 为 null 时，必须提第二个参数。
            生成的 SQL 语句才不会出错！
            id = db.insert(tableName, "_id", values);
            return ContentUris.withAppendedId(uri, id);
        case CONTACT:
            id = db.insert(tableName, "_id", values);
            String uriPath = uri.toString();
            String path = uriPath.substring(0, uriPath.lastIndexOf("/")) + id;
            return Uri.parse(path);
        default:
            throw new IllegalArgumentException("Uri IllegalArgument:" + uri);
    }
}

@Override
public boolean onCreate() {

```



```

        this.openHelper = new MyOpenHelper(this.getContext());
        return true;
    }

    @Override
    public Cursor query(Uri uri, String[] projection, String selection,
        String[] selectionArgs, String sortOrder) {
        SQLiteDatabase db = this.openHelper.getWritableDatabase();
        switch (uriMatcher.match(uri)) {
            case CONTACT_LIST:
                return db.query(tableName, projection, selection, selectionArgs,
                    null, null, sortOrder);

            case CONTACT:
                long id = ContentUris.parseId(uri);
                String where = "_id=" + id;
                if (selection != null && !"".equals(selection)) {
                    where = where + " and " + selection;
                }
                return db.query(tableName, projection, where, selectionArgs, null,
                    null, sortOrder);

            default:
                throw new IllegalArgumentException("Uri IllegalArgument:" + uri);
        }
    }

    @Override
    public int update(Uri uri, ContentValues values, String selection,
        String[] selectionArgs) {
        SQLiteDatabase db = this.openHelper.getWritableDatabase();
        int result;
        switch (uriMatcher.match(uri)) {
            case CONTACT_LIST:
                result = db.update(selection, values, selection, selectionArgs);
                break;

            case CONTACT:
                long id = ContentUris.parseId(uri);
                String where = "_id=" + id;
                if (selection != null && !"".equals(selection)) {
                    where = where + " and " + selection;
                }
                result = db.update(tableName, values, where, selectionArgs);
                break;

            default:
                throw new IllegalArgumentException("Uri IllegalArgument:" + uri);
        }
        return result;
    }

```



```

    }
}

```

2. 添加 ContentProvider 配置

```

<provider                                android:name=".provider.ContactContentProvider"
    android:authorities="com.changcheng.sqlite.provider.contactprovider"/>

```

6、执行 SQL 语句进行查询

用法 1

```

SQLiteDatabase db;
String args[] = {id};
ContentValues cv = new ContentValues();

cv.put("android123", id);
Cursor c = db.rawQuery("SELECT * FROM table WHERE android123=?", args); 执行本地 SQL 语句查询

if (c.getCount() != 0) {
    //dosomething
    ContentValues cv = new ContentValues();

    cv.put("android123", "cwj");
    db.insert("table", "android123", cv); //插入数据

    String args[] = {id};
    ContentValues cv2= new ContentValues();
    cv2.put("android123", id);
    db.delete("table", "android123=?", args); //删除数据
}

```

其它:

```

//定义数据库
SQLiteDatabase myDB = null;
//打开或建立数据库（当数据库不存在时，自动分创建）
myDB = this.openOrCreateDatabase("MY_DATABASE_NAME", "MODE_PRIVATE", null);

//删除表格、新建表格、插入数据、更新数据，地球人都知道，我也不说了

```



```

myDB.execSQL(".....标准的 SQL 语句.....");

//查询比较麻烦，我这里列出代码
//查询(表格和数据你自己去建立吧)
Cursor c = myDB.rawQuery("SELECT id,name,tel FROM MY_DATABASE_TABLE
WHERE age>1 order by age;", null);
if (c != null) {
    if (c.moveToFirst()) {
        do {
            Log.w("test", "id="+c.getString(c.getColumnIndex("id")));
            Log.w("test", "name="+c.getString(c.getColumnIndex("name")));
            Log.w("test", "tel="+c.getString(c.getColumnIndex("tel")));
            Log.w("test", "-----");
        } while (c.moveToNext());
    }
}
c.close();

//关闭数据库
myDB.close();

```

详解:

一：在 android 系统中除了文件和 sharedPreferences 可以存储数据外，还可以用 SQLite 数据库，它是 android 自带的嵌入式的关系型的数据库，它支持 null, Integer, real, text, blob（二进制数据）五种数据类型，但实际运算和保存数据时它可以接受其它数据类型，只是这时候会转换为对应的五种数据类型。但有一种情况例外：定义为 Integer primary key 的字段只能存储 64 位整数，当向这种字段保存除整数以外的数据时，将会产生错误。另外，SQLite 在解析 Create Table 语句时，会忽略 create table 语句中跟在字段后面的数据类型信息。

二：在程序初始化时候，必须先建立数据库以对数据库进行更新，所以这里需要继承一个 SQLiteOpenHelper 抽象类，这里有两个方法 onCreate(),onUpgrade()两个方法，前者用来创建数据库及对数据库的一些初始化操作，后者是当数据库版本进行更新时候用。下面的例子是创建了一个数据库名为：nyist,版本为 1，的数据库，并在数据库中创建了一个 person 表。这里利用了构造函数传入了 数据库名和版本名常量，参数为 Context 对象。

```

public class DatabaseHelper extends SQLiteOpenHelper {
    public final static String NAME="nyist";
    public final static int VERSION=1;
    public DatabaseHelper(Context context) {
        super(context, NAME, null, VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        // TODO Auto-generated method stub
        db.execSQL("CREATE TABLE person (personid integer primary key autoincrement, name

```

```

varchar(20), age integer)");
}
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // TODO Auto-generated method stub
    db.execSQL("DROP TABLE IF EXISTS person");
    onCreate(db);
}
}

```

三：用 SQLiteDatabase 来操作 SQLite 数据库。

这里主要用 `execSQL()`, `rawQuery()` 分别用来对 `insert`, `delete`, `update` 这些对数据库更新的操作，`select` 查询的操作。下面的例子就是 CRUD 操作。

```

public class DatabaseService {
    private DatabaseHelper dbHelper;
    public DatabaseService(Context context) {
        this.dbHelper = new DatabaseHelper(context);
    }
    //保存数据。
    public void save(Person person){
        SQLiteDatabase db=dbHelper.getWritableDatabase();
        db.execSQL("insert into person(name,age) values(?,?)", new
        Object[] {person.getName(),person.getAge()});

    }
    //更新数据
    public void update(Person person){
        SQLiteDatabase db=dbHelper.getReadableDatabase();
        db.execSQL("update person set name=?,age=? where personid=?", new
        Object[] {person.getName(),person.getAge(),person.getPersonid()});
    }
    //查找数据
    public Person find(Integer personid){
        SQLiteDatabase db=dbHelper.getReadableDatabase();
        Cursor cursor=db.rawQuery("select * from person where personid=?",new
        String[] {personid.toString()} );
        while(cursor.moveToNext()){
            String name=cursor.getString(cursor.getColumnIndex("name"));
            int age=cursor.getInt(cursor.getColumnIndex("age"));
            int id=cursor.getInt(cursor.getColumnIndex("personid"));
            return new Person(id,name,age);
        }
        return null;
    }
}

```



//删除数据

```
public void delete(Integer personid){
    SQLiteDatabase db=dbHelper.getReadableDatabase();
    db.execSQL("delete from person where personid=?",new Object[]{personid});
}

public Long getCount(){
    SQLiteDatabase db=dbHelper.getReadableDatabase();
    Cursor cursor = db.rawQuery("select count(*) from person", null);
    cursor.moveToFirst();
    return cursor.getLong(0);
}
```

//分页显示 返回的是 List 集合

```
public List<Person> getScrollData(int offer,int maxResult){
    List<Person> persons=new ArrayList<Person>();
    SQLiteDatabase db=dbHelper.getReadableDatabase();
    Cursor cursor=db.rawQuery("select * from person limit ?,?",new
String[]{String.valueOf(offer),String.valueOf(maxResult)});
    while(cursor.moveToNext()){
        int id=cursor.getInt(cursor.getColumnIndex("personid"));
        String name=cursor.getString(cursor.getColumnIndex("name"));
        int age=cursor.getInt(cursor.getColumnIndex("age"));
        persons.add(new Person(id,name,age));
    }
    return persons;
}
```

//分页显示 返回的是 Cursor 游标

```
public Cursor getCursorScrollData(int offer,int maxResult){
    List<Person> persons=new ArrayList<Person>();
    SQLiteDatabase db=dbHelper.getReadableDatabase();
    Cursor cursor=db.rawQuery("select personid as _id,name,age from person limit ?,?",new
String[]{String.valueOf(offer),String.valueOf(maxResult)});

    return cursor;
}

}
```

四：除了 `execSQL()`,`rawQuery()`这两个方法外，`SQLiteDatabase` 类也专门提供了一些函数来操作数据库，有 `insert`, `delete`,`update`,`query()`,不过这些函数需要的参数比较多，只适合对 SQL 语句不太懂的菜鸟用，对于熟悉 SQL 语句的程序员最好使用 `execSQL()`,`rawQuery()`,这两个方法比较直观明了。

五：调用 `getWritableDatabase()`或 `getReadableDatabase()`方法后，会缓存 `SQLiteDatabase` 实例，

因为这里是手机应用程序，一般只有一个用户访问数据库，所以建议不关闭数据库，保持连接状态。getWritableDatabase(), getReadableDatabase 的区别是当数据库写满时候，调用前者会报错，调用后者不会，所以如果不是更新数据库的话，最好调用后者来获得数据库连接。

六：关于事务。

当执行两个或多条语句时候，有一条出现错误不能执行，其它的也语句也不能执行。比如银行系统，当转账时候，钱数扣除，转账的号数据错误，这个事务就会中止。

```
public void payment(){
    SQLiteDatabase db = dbOpenHelper.getWritableDatabase();
    db.beginTransaction();//开启事务
    try{
        db.execSQL("update person set amount=amount-10 where personid=?", new Object[]{"1"});
        db.execSQL("update person 55set amount=amount+10 where personid=?", new Object[]{"2"});
        db.setTransactionSuccessful();//设置事务标志为成功，在结束事务将会提交事务
    }finally{
        db.endTransaction();//结束事务,默认是回滚事务
    }
}
```

查看SQLite表格内容

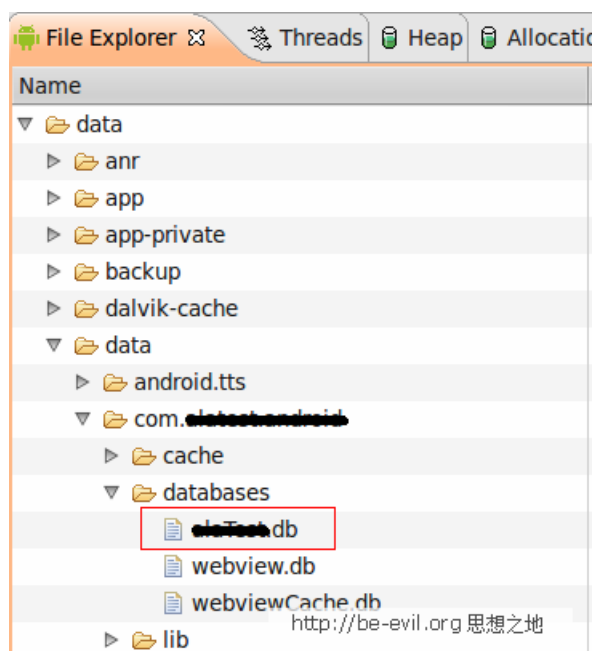
Android中可以采用sqlite数据库来存储数据，是Google却没有直接给我们提供相关工具来管理数据库里的数据。

如果不能直接通过工具来查看，那我们就把数据库从手机/模拟器里面拷贝出来用工具查看，下面是步骤

1.确认数据库的位置

我们可以通过eclipse的DDMS插件来访问手机的部分目录

数据库文件位于/data/data/你的程序的包名/databases/中，下图是一个例子



2. 拷贝出数据库文件 (两种方法)

(1) 我们可以用adb工具来下载数据库文件

命令为 `adb push 手机路径 本地路径`

例如我要把项目下的test.db数据拷贝到我的桌面, 那么运行命令

`./adb pull /data/com.test/databases/test.db ~/Desktop/`

(2) 找找右上角有没有这个, 那就点击保存按钮, 就是左边那个, 指定保存目录即可导出了~~



3. 打开数据库文件

这里不用多说, 去sqlite官方 选一款软件来读取和管理数据即可

4. 更新数据库文件

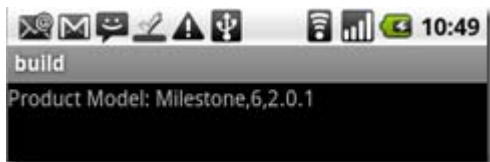
同样使用adb工具上传

命令为 `adb push 本地路径 手机路径`

`./adb push ~/Desktop/alaTest.db /data/data/com.test/databases/test.db`

9、常用功能的实现

9.1、获取手机型号以及系统版本号



有时候需要统计手机的型号和版本号, 利用程序可以获取到相应的手机信息,

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    TextView textView = (TextView) findViewById(R.id.text);
    textView.setText("Product Model: " + android.os.Build.MODEL +
        ", "
        + android.os.Build.VERSION.SDK + ", "
        + android.os.Build.VERSION.RELEASE);
}
```

获得 SDK 版本信息:



```

StringBuffer buf = new StringBuffer();
buf.append("VERSION.RELEASE {" + Build.VERSION.RELEASE + "}");
buf.append("\nVERSION.INCREMENTAL {" + Build.VERSION.INCREMENTAL
);
buf.append("\nVERSION.SDK {" + Build.VERSION.SDK + "}");
buf.append("\nBOARD {" + Build.BOARD + "}");
buf.append("\nBRAND {" + Build.BRAND + "}");
buf.append("\nDEVICE {" + Build.DEVICE + "}");
buf.append("\nFINGERPRINT {" + Build.FINGERPRINT + "}");
buf.append("\nHOST {" + Build.HOST + "}");
buf.append("\nID {" + Build.ID + "}");
Log.d("build", buf);

```

获得手机 UA 的一个方法:

```

public String getUserAgent()
{
    String user_agent = ProductProperties.get(ProductProperties.USER_AGENT_KEY, null);
    return user_agent;
}

```

9.2、更改应用程序图标

在 res 目录下, 把 drawable 所有文件夹下的默认 png 图标以相同名字 (icon.png) 的自定义图标替换掉即可

9.3、迎合不同的手机分辨率

drawable-hdpi、drawable-mdpi、drawable-ldpi 的区别:

(1) drawable-hdpi 里面存放高分辨率的图片, 如 WVGA (480x800), FWVGA (480x854)

(2) drawable-mdpi 里面存放中等分辨率的图片, 如 HVGA (320x480)

(3) drawable-ldpi 里面存放低分辨率的图片, 如 QVGA (240x320)

系统会根据机器的分辨率来分别到这几个文件夹里面去找对应的图片。

在开发程序时为了兼容不同平台不同屏幕, 建议各自文件夹根据需求均存放不同版本图片。



(4) 尽量不适用像素单位而多使用 dip 单位

9.4. Android屏幕适应的四个原则

Best practices for Screen Independence

1. Prefer wrap_content, fill_parent and the dip unit to absolute pixels

When defining the layout_width and layout_height of views in an XML layout file, using wrap_content, fill_parent or the dip will guarantee that the view is given an appropriate size on the current device screen. For instance, a view with a layout_width="100dip" will measure 100 pixels wide on an HVGA@160 density display and 150 pixels on a WVGA@240 density display, but the view will occupy approximately the same physical space.

Similarly, you should prefer the sp (scale-independent pixel, the scale factor depends on a user setting) or dip (if you don't want to allow the user to scale the text) to define font sizes.

2. Avoid AbsoluteLayout

AbsoluteLayout is one of the layout containers offered by the Android UI toolkit. Unlike the other layouts however, AbsoluteLayout enforces the use of fixed positions which might easily lead to user interfaces that do not work well on different displays. Because of this, AbsoluteLayout was deprecated in Android 1.5 (API Level 3).

You can achieve much the same layout by using a FrameLayout instead, and setting layout_margin attributes of the children. This approach is more flexible and will yield better results on different screens.

3. Do not use hard-coded pixel values in your code

编码中定义的 数值默认都是 pixel，所以不要写死。如果写死的要根据 屏幕的 density 值 做转换。

```
width = this.getWindowManager().getDefaultDisplay().getWidth();  
float tempx = 94.0f / 320 * width;
```

For performance reasons and to keep the code simpler, the Android framework API uses pixels as the standard unit for expressing dimension or coordinate values. That means that the dimensions of a View are always expressed in the code in pixels. For instance, if myView.getWidth() returns 10, the view is 10 pixels wide. In some cases, you may need to scale the pixel values that you use in your code. The sections below provide more information.

Converting from dips to pixels

In some cases, you will need to express dimensions in dip and then convert them to pixels. Imagine an application in which a scroll gesture is recognized after the user's finger has moved by at least 16 pixels. On a baseline screen, the user will have to move his finger by 16 pixels / 160 dpi = 1/10th of an inch (or 2.5 mm) before the gesture is

recognized. On a device with a high (240) density display, the user will move his finger by only 16 pixels / 240 dpi = 1/15th of an inch (or 1.7 mm.) The distance is much shorter and the application thus appears more sensitive to the user. To fix this issue, the gesture threshold must be expressed in the code in dip and then converted to actual pixels.

```
// The gesture threshold expressed in dip
```

```
private static final float GESTURE_THRESHOLD_DIP = 16.0f;
```

```
// Convert the dips to pixels
```

```
final float scale = getContext().getResources().getDisplayMetrics().density;  
mGestureThreshold = (int) (GESTURE_THRESHOLD_DIP * scale + 0.5f);
```

```
// Use mGestureThreshold as a distance in pixels
```

The `android.util.DisplayMetrics.density` field specifies the the scale factor you must use to convert dips to pixels according to the current screen density. You can access the current screen's metrics through a `Context` or `Activity`. On a medium (160) density screen, `DisplayMetrics.density` equals "1.0", whereas on a high (240) density screen it equals "1.5". You can refer to the documentation of the `DisplayMetrics` class for details.

Use pre-scaled configuration values

The `ViewConfiguration` class can be used to access the most common distances, speeds, and times used in the Android framework. For instance, the distance in pixels used by the framework as the scroll threshold can be obtained as follows:

```
ViewConfiguration.get(aContext).getScaledTouchSlop()
```

Methods starting with the `getScaled` prefix are guaranteed to return a value in pixels that will display properly regardless of the current screen density.

4. Use density and/or size-specific resources

9.5、android常用单位

1. px (像素)：屏幕上的点。
2. in (英寸)：长度单位。
3. mm (毫米)：长度单位。
4. pt (磅)：1/72 英寸。



5. dp（与密度无关的像素）：一种基于屏幕密度的抽象单位。在每英寸 160 点的显示器上，1dp = 1px。在大于 160 点的显示器上可能增大。
6. dip：与 dp 相同，多用于 Google 示例中。
7. sp（与刻度无关的像素）：与 dp 类似，但是可以根据用户的字体大小首选项进行缩放。
- 8.

9.6、取得屏幕信息

```
DisplayMetrics dm = new DisplayMetrics();
dm = this.getResources().getDisplayMetrics();
//获得屏幕宽度
int screenWidth = dm.widthPixels;
//获得屏幕高度
int screenHeight = dm.heightPixels;
```

9.7、横竖屏

获取屏幕方向

```
Configuration newConfig = getResources().getConfiguration();
if (newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE){
    //横屏
}else if(newConfig.orientation == Configuration.ORIENTATION_PORTRAIT){
    //竖屏
}else if(newConfig.hardwareKeyboardHidden == Configuration.KEYBOARDHIDDEN_NO){
    //键盘没关闭。屏幕方向为横屏
}else if(newConfig.hardwareKeyboardHidden == Configuration.KEYBOARDHIDDEN_YES){
    //键盘关闭。屏幕方向为竖屏
}
```

横竖屏切换问题

很多没有购买真机的网友不知道如何切换 Android 模拟器到横屏显示。常规的显示为 HVGA-P(port)，即分辨率为 320x480 如果使用横屏(land)。如果模拟器尚未启动，可以在 Eclipse 的项目 Run as=>Open Run Dialog 对话框中设置，如果 android 模拟器已经启动后，可以使用快捷键 F12 或 Ctrl+F11 或小键盘 7、8 来切换。当然是用命令行仅仅启动模拟器可以使用参数 emulator.exe -skin HVGA-L 来启动。

需要注意的是，切换 land 或 port 可以通过资源文件来让界面自适应窗体，但程序可能会重载 onCreate，避免的方法可以通过在 androidmanifest.xml 文件中重新定义方向，以及根据 Activity 的重写 onConfigurationChanged(Configuration newConfig)方法来控制，相关的可以在 Android SDK 中获取到。

在开发游戏的时候，有些游戏是只能横屏玩的，所以手机竖立放置的时候，要保持游戏画面依然横屏。要做到这个要求其实很简单，在 `AndroidManifest.xml` 里面配置一下就可以了。加入这一行 `android:screenOrientation="landscape"`。

例如（`landscape` 是横向，`portrait` 是纵向）：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.ray.linkit"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".Main"
            android:label="@string/app_name"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".GamePlay"
            android:screenOrientation="portrait"></activity>
        <activity android:name=".OptionView"
            android:screenOrientation="portrait"></activity>
    </application>
    <uses-sdk android:minSdkVersion="3" />
</manifest>
```

另外，android 中每次屏幕的切换都会重启 Activity，所以应该在 Activity 销毁前保存当前活动的状态，在 Activity 再次 Create 的时候载入配置，那样，进行中的游戏就不会自动重启了！

案例

```
package com.cn;
import android.app.Activity;
import android.content.res.Configuration;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;

public class LayoutTest extends Activity {

    private static final String tag = "LayoutTest";
```



```

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    Log.e(tag, "*****test*****");
    // LOGV("*****test*****");

    if(this.getResources()
.getConfiguration()
.orientation == Configuration.ORIENTATION_LANDSCAPE) {
Log.e(tag, "^^^^^^^^^this is a landscape *****");
    }
    // TextView tx = (TextView)findViewById(R.id.txt);
    // tx.setKeepScreenOn(true);
    // if(this.getResources())

    setContentView(R.layout.main);
}

@Override
public void onConfigurationChanged(Configuration newConfig) {
    // TODO Auto-generated method stub
    super.onConfigurationChanged(newConfig);

    if(this.getResources()
.getConfiguration()
.orientation == Configuration.ORIENTATION_PORTRAIT) {
Log.e(tag, "*****this is a portrait");
    }else if(this.getResources()
.getConfiguration()
.orientation == Configuration.ORIENTATION_LANDSCAPE) {
Log.e(tag, "this is a landscape *****");
    }
}
}

```

AndroidManifest.xml

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.cn"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">

```



```

        <activity android:name=".LayoutTest"
            android:label="@string/app_name"

            android:configChanges="orientation|keyboardHidden" >
        <!--      android:screenOrientation="landscape">
-->

        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
<uses-sdk android:minSdkVersion="3" />
</manifest>

```

9.8、程序完全全屏

```

//不显示标题
requestWindowFeature(Window.FEATURE_NO_TITLE);
//设置窗口全屏显示
getWindow().setFlags(WindowManager.LayoutParams.
FLAG_FULLSCREEN, WindowManager.LayoutParams. FLAG_FULLSCREEN);

```

9.8.1 锁屏锁键盘

从 Android 2.2 开始，加入了一个新的锁屏 API 位于 `android.app.admin.DevicePolicyManager` 包，`DevicePolicyManager` 类的 `lockNow` 方法可以锁住屏幕，查看 Android 源代码发现其实是从 `IDevicePolicyManager` 实现的，整个 AIDL 接口调用代码为：

```

private final IDevicePolicyManager mService;
mService = IDevicePolicyManager.Stub.asInterface(
ServiceManager.getService(Context.DEVICE_POLICY_SERVICE));
if (mService != null) {
    try {
        mService.lockNow();
    } catch (RemoteException e) {
        Log.w(TAG, "Failed talking with device policy service", e);
    }
}

```

加入 `<uses-permission android:name="android.permission.DISABLE_KEYGUARD"></uses-permission>` 权限，使用下面代码可以锁住键盘，但屏幕不行




```
KeyguardManager km =
(KeyguardManager)getSystemService(Context.KEYGUARD_SERVICE);
KeyguardLock kl= km.newKeyguardLock(KEYGUARD_SERVICE);
kl.reenableKeyguard();
DevicePolicyManager 类的 lockNow 方法可以锁住屏幕，查看 Android 源代码发现其实是从
IDevicePolicyManager 实现的，
```

整个 AIDL 接口调用代码为:

```
Private final IDevicePolicyManager mService;

mService =
IDevicePolicyManager.Stub.asInterface(ServiceManager.getService(Context.DEVICE_POLICY_
SERVICE));

if (mService != null ) {
try
{
mService.lockNow();
}
catch
(RemoteException e) {
Log.w(TAG, "Failed talking with device policy service", e);
}
}
```

9.9、程序的开机启动

在 Windows 平台下安装一些软件时，经常会遇到一些软件带有自启动设置。通常会关掉这些功能，除非对于某些重要的程序有必要开启这项功能。在 Android 平台也可以方便的给程序添加自启动设置，下边给出了具体的实现方法：

首先需要在 Manifest 中做如下修改和补充：

[View Code](#) XML

```
<receiver android:enabled="true" android:name=".BootUpReceiver"
android:permission="android.permission.RECEIVE_BOOT_COMPLETED">

<intent-filter>
<action android:name="android.intent.action.BOOT_COMPLETED" />
```

作者:craining (曲阜师范大学) 个人主页:<http://craining.blog.163.com/> 邮箱:craining@163.com 201



```

        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</receiver>

```

程序主体中需要实现在 Manifest 中声明的 BroadcastReceiver 类:

```

public class BootUpReceiver extends BroadcastReceiver{

    @Override
    public void onReceive(Context context, Intent intent) {
        Intent i = new Intent(context, MyActivity.class);
        i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(i);
    }

}

```

1. 定义一个 BroadcastReceiver

代码:

```

public class BootReceiver extends BroadcastReceiver {
    public void onReceive(Context ctx, Intent intent) {
        Log.d("BootReceiver", "system boot completed");
        //start activity
        String action="android.intent.action.MAIN";
        String category="android.intent.category.LAUNCHER";
        Intent myi=new Intent(ctx, CustomDialog.class);
        myi.setAction(action);
        myi.addCategory(category);
        myi.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        ctx.startActivity(myi);
        //start service
        Intent s=new Intent(ctx, MyService.class);
        ctx.startService(s);
    }
}

```

2、配置 Receiver 的许可,允许接收系统启动消息, 在 AndroidManifest.xml 中:

```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
```

3、配置 Receiver,可以接收系统启动消息, 在 AndroidManifest.xml 中



```
<receiver android:name=".app.BootReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
        <category android:name="android.intent.category.HOME" />
    </intent-filter>
</receiver>
```

4、启动模拟器，可以看到系统启动后，弹出一个对话框。

例一

如果您在开发一个需要实时更新数据的应用程序，当有新的数据的时候提醒用户查看新的数据，那么您需要在后台开起一个 Service，然后实时的去网络上获取数据，但是如果用户关机重启，您的 Service 可能就消失了！那么怎么样保证开机后你的 Service 还活跃的在用户的手机里偷偷的从网络上获取数据呢？

很简单，我们只要实现开机自启动即可，Android 实现开机自启动可能是移动操作系统中最简单的了，我们只需要监听一个开机启动的 Broadcast（广播）即可。首先写一个 Receiver（即广播监听器），继承 BroadcastReceiver，如下所示：

```
public class BootReceiver extends BroadcastReceiver {
    private PendingIntent mAlarmSender;
    @Override
    public void onReceive(Context context, Intent intent) {
        // 在这里干你想干的事（启动一个 Service, Activity 等），本例是启动一个定时调度程序，
        // 每 30 分钟启动一个 Service 去更新数据
        mAlarmSender = PendingIntent.getService(context, 0, new Intent(context,
        RefreshDataService.class), 0);
        long firstTime = SystemClock.elapsedRealtime();
        AlarmManageram=(AlarmManager)context.getSystemService(Activity.ALARM_SERVICE);
        am.cancel(mAlarmSender);
        am.setRepeating(AlarmManager.ELAPSED_REALTIME_WAKEUP, firstTime, 30*60*1000,
        mAlarmSender);
    }
}
```

接下来，我们只需要在应用程序配置文件 AndroidManifest.xml 中注册这个 Receiver 来监听系统启动事件即可，如下所示：

```
<receiver android:name=".service.BootReceiver">
    <intent-filter>
        <!-- 系统启动完成后会调用 -->
        <action android:name="android.intent.action.BOOT_COMPLETED">
    </action>
    </intent-filter>
</receiver>
```

例二：

android 开机启动 service，适合闹钟程序 实例中一共三个类

1. public class YourReceiver extends BroadcastReceiver



2. public class ServiceTest extends Service
3. public class ShowActivity extends Activity

1. yourReceiver 类:

```
package radar.com;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class YourReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        Intent i = new Intent(context, ServiceTest.class);
        i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startService(i);
    }

}
```

2. 复制代码ServiceTest 类:

```
package radar.com;

import java.util.Calendar;
import android.app.Service;
import android.content.Intent;
import android.os.Handler;
import android.os.IBinder;
import android.util.Log;

public class ServiceTest extends Service{
    Handler hd1=new Handler();
    /**启动activity的开关*/
    boolean b;
    /**启动一次activity之后的一分钟内将不再重新启动*/
    int time;
    public static final Intent ACTION_START = null;
    private static final String TAG = "TestService";
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
}
```



```

@Override
public boolean onUnbind(Intent i) {
    Log.e(TAG, "=====> TestService.onUnbind");
    return false;
}

@Override
public void onRebind(Intent i) {
    Log.e(TAG, "=====> TestService.onRebind");
}

@Override
public void onCreate() {

    Log.e(TAG, "=====> TestService.onCreate");
    hdl.postDelayed(mTasks, delay);
}

@Override
public void onStart(Intent intent, int startId) {
    Log.e(TAG, "=====> TestService.onStart");
}

@Override
public void onDestroy() {
    Log.e(TAG, "=====> TestService.onDestroy");
}

public void log() {
    Calendar c= Calendar.getInstance();
    int h=c.getTime().getHours();
    int m=c.getTime().getMinutes();
    Log.i("hour", ""+h);
    Log.i("minute", ""+m);
    /**这里的16和10可以自己定义一下 主要是提醒的时间设置, 我不想做的太繁琐, 所有没有
    有些闹钟, 只是用这个测试一下:*/)
    if (h==16&& m==10)
    {

        /**为防止持续调用, 所以用boolean 变量b做了一个小开关*/
        if(!b) {

            Intent i = new Intent();
            i.setClass(ServiceTest.this, ShowActivity.class);
            i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            this.startActivity(i);
            this.stopSelf();

```



```

        b=true;
    }
}

/**开关开启后计时60秒，在这60秒之内就不再重新启动activity了，而60秒一过，上面的h
和m条件肯定就不成立了*/
    if(b){
        time+=5;
        if(time==60){
            time=0;
            b=false;
        }
    }
}

/** 速度控制参数(单位毫秒) */
private int delay = 5000;
/**
 * 控制速度
 */
private Runnable mTasks = new Runnable()
{
    public void run()
    {
        log();

        hdl.postDelayed(mTasks, delay);
    }
};
}

```

3. 复制代码showActivity 类: (. 此类中啥都没有，就是演示一下activity可以被启动)

```

package radar.com;

import radar.com.R;
import android.app.Activity;
import android.os.Bundle;
import android.view.Window;
import android.view.WindowManager;

public class ShowActivity extends Activity{
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,

```



```

        WindowManager.LayoutParams.FLAG_FULLSCREEN);
    setContentView(R.layout.main);
}
}

```

4. 复制代码下面是很重要的AndroidManifest

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="radar.com"
    android:versionCode="1"
    android:versionName="1.0">
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name">
        <service
            android:name=".ServiceTest"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </service>

        <receiver android:name=".yourReceiver" >
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED" />
                <category android:name="android.intent.category.HOME" />
            </intent-filter>
        </receiver>
        <activity android:name=".showActivity"
            android:label="@string/app_name"
            android:configChanges="orientation|keyboardHidden|navigation"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="4" />
</manifest>

```

既可以作为开机启动并隐藏到后台的service，也可以当做activity打开，可以说东西会



很方便。

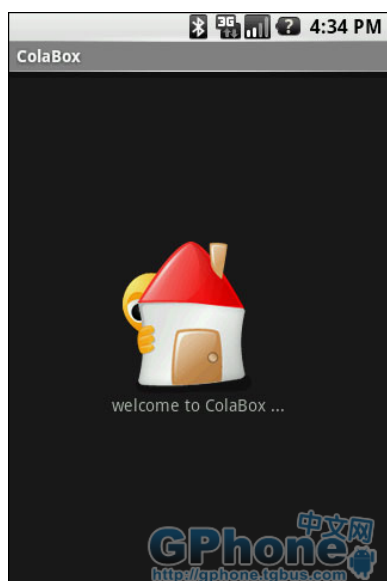
9.10、动态start页面

启动界面的主要功能就是显示一幅启动图像,后台进行系统初始化.

如果是第一次使用本程序,需要初始化本程序的 sqlite 数据库,建库,建 Table,初始化账目数据.

如果不是第一次使用,就进入登记收支记录界面.

界面效果如图:



界面很简单,一个 imageview 和一个 textview

可是如何是 2 个 view 垂直居中显示,我开始使用 linearlayout 就没法完成垂直和横向居中.

后来使用 RelativeLayout 才搞定了横向居中.

界面的具体 xml 如下:

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout android:id="@+id/RelativeLayout01"
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_gravity="center_vertical|center_horizontal"
android:layout_height="wrap_content"
android:layout_width="wrap_content">
<ImageView android:id="@+id/ImageView01"
android:src="@drawable/logo3"
android:layout_width="wrap_content"
android:layout_height="wrap_content">
```




```

</ImageView>
<TextView android:id="@+id/TextView01"
    android:text="@string/welcome"
    android:layout_below="@id/ImageView01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
</TextView>
</RelativeLayout>

```

然后在 String.xml 中添加字符声明;

```
<string name="welcome">欢迎使用·····</string>
```

在这儿我来使用一个小技巧,就是在程序初始化完成后,让图片淡出,然后显示下一个界面.

开始我准备使用一个 timer 来更新图片的 alpha 值,后来程序抛出异常 Only the original thread that created a view hierarchy can touch its views.

这才发现 android 的 ui 控件是线程安全的.

这里我们需要在主线程外,再开一个线程更新界面上的图片.可以使用 imageView.invalidate

关于如何另开一个线程更新界面的相关代码如下.

//给主线程发送消息更新 imageView

```

mHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        imageView.setAlpha(alpha);
        imageView.invalidate();
    }
};

new Thread(new Runnable() {
    public void run() {
        while (b < 2) {
            try {
                //延时 2 秒后,每 50 毫秒更新一次 imageView
                if (b == 0) {
                    Thread.sleep(2000);
                    b = 1;
                } else {
                    Thread.sleep(50);
                }
            } catch (InterruptedException e) {}

            updateApp();
        }
    }
}

```



```
    } catch (InterruptedException e) {  
        e.printStackTrace();  
    }  
}  
  
}  
}).start();  
  
public void updateApp() {  
    alpha -= 5; //每次减少 alpha 5  
  
    if (alpha <= 0) {  
        ib = 2;  
        Intent in = new Intent(this, com.cola.ui.Frm_Addbills.class);  
        startActivity(in); //启动下个界面  
    }  
    mHandler.sendMessage(mHandler.obtainMessage());  
}
```

通过这段代码,我们能够理解 android 里面如何对 ui 视图进行更新.

下篇文章我们来看看 sqlite 的使用.如何初始化程序.

关于 handler, invalidate 的用法,

附 ColaBox.java:

```
package com.cola.ui;  
  
import android.app.Activity;  
import android.content.Intent;  
import android.os.Bundle;  
import android.os.Handler;  
import android.os.Message;  
import android.util.Log;  
import android.view.KeyEvent;  
import android.widget.ImageView;  
import android.widget.TextView;  
  
public class ColaBox extends Activity {  
    private Handler mHandler = new Handler();  
  
    ImageView imageview;  
    TextView textview;  
    int alpha = 255;  
    int b = 0;  
    public void onCreate(Bundle savedInstanceState) {
```



```
super.onCreate(savedInstanceState);
setContentView(R.layout.main);

imageView = (ImageView) this.findViewById(R.id.ImageView01);
textView = (TextView) this.findViewById(R.id.TextView01);

Log.v("ColaBox", "ColaBox start ...");
imageView.setAlpha(alpha);

new Thread(new Runnable() {
    public void run() {
        initApp(); //初始化程序

        while (b < 2) {
            try {
                if (b == 0) {
                    Thread.sleep(2000);
                    b = 1;
                } else {
                    Thread.sleep(50);
                }

                updateApp();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

    }
}).start();

mHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        imageView.setAlpha(alpha);
        imageView.invalidate();
    }
};
}
```



```
public void updateApp() {
    alpha -= 5;

    if (alpha <= 0) {
        b = 2;
        Intent in = new Intent(this, com.cola.ui.Frm_Addbills.class);
        startActivity(in);
    }

    mHandler.sendMessage(mHandler.obtainMessage());

}

public void initApp(){

}

}
```

9.11、彻底退出当前程序

//在 finish 之后， 应用程序正在运行的进程列表中仍存在，可以结束进程将其移除列表：

```
finish();
android.os.Process.killProcess(android.os.Process.myPid());
```

9.12、获取应用程序的名称，包名，版本号和图标

```
class PInfo {
    private String appname = "";
    private String pname = "";
    private String versionName = "";
    private int versionCode = 0;
    private Drawable icon;
    private void prettyPrint() {
        log(appname + "\\t" + pname + "\\t" + versionName + "\\t" + versionCode +
        "\\t");
    }
}

private void listPackages() {
    ArrayList<PInfo> apps = getInstalledApps(false); /* false = no system packages */
```



```

        final int max = apps.size();
        for (int i=0; i<max; i++) {
            apps.get(i).prettyPrint();
        }
    }

    private ArrayList<PInfo> getInstalledApps(boolean getSysPackages) {
        ArrayList<PInfo> res = new ArrayList<PInfo>();
        List<PackageInfo> packs = getPackageManager().getInstalledPackages(0);
        for(int i=0;i<packs.size();i++) {
            PackageInfo p = packs.get(i);
            if ((!getSysPackages) && (p.versionName == null)) {
                continue ;
            }
            PInfo newInfo = new PInfo();
            newInfo.appname = p.applicationInfo.loadLabel(getPackageManager()).toString();
            newInfo.pname = p.packageName;
            newInfo.versionName = p.versionName;
            newInfo.versionCode = p.versionCode;
            newInfo.icon = p.applicationInfo.loadIcon(getPackageManager());
            res.add(newInfo);
        }
        return res;
    }
}

```

使用 android 隐藏 api

前提: 我们需要得到 Android 系统源码编译输出的一个文件

out\target\common\obj\JAVA_LIBRARIES\framework_intermediates\classes.jar

这个包里面包含所有的系统 api, 隐藏的, 公开的

添加 jar 方法

右键功能菜单->Properties->Java Build Path

Libraries 选项卡

这时应该有一个列表, 如果你没有添加过, 应该只有一项, 就是系统自带的 Android SDK, 选中后, 右边有一个删除, 先删除系统添加的 sdk.

点 Add Library -> User Library

选择 User Library 按钮, 新建一个 User Library 将刚才那个文件 classes.jar 和系统本身的文件都导入进来, 调整下顺序, 将 classes.jar 调到前面

这样添加了之后, 就可以使用系统隐藏的 api 了

使用隐藏 api, 有个前提:

许多 api 涉及到系统权限问题, 比如 后台安装文件 api PackageManager.installPackage 要求有安装



程序的权限，而这个安装程序权限不是随便有的，只有经 ROM 签名认证的才可以使用这个权限。虽然可以在配置文件里面添加这个权限，但是悲剧的是你仍然不能拥有这个权限，在这点上，Google 做的真绝..

好了，虽然我们不能安装，但用 api 去查看 apk 总该可以了吧？

Google 没有公开这个 Api，但又有了上面这个方法，我们可以使用了 **//apk 包的文件路径**

```
String apkPath =
    \"/sdcard/qq.apk\";
    //这是一个 Package 解释器，是隐藏的
    //构造函数的参数只有一个，apk 文件的路径
    PackageParser packageParser =
new PackageParser(apkPath);
    //这个是与显示有关的，里面涉及到一些像素显示等等，我们使用默认的情况
    DisplayMetrics metrics =
new DisplayMetrics();
    metrics.setToDefaults();
    //这里就是解析了，四个参数，
    //源文件 File，
    //目的文件路径(这个我也没搞清楚怎么回事，看 Android 安装器源码，用的是源文件路径，但
名字却是 destFileName)
    //显示，DisplayMetrics metrics
    //flags，这个真不知道是啥
    PackageParser.Package mPkgInfo = packageParser.parsePackage(new File(apkPath),
        apkPath, metrics, 0);

    //应用程序信息包，这个公开的，不过有些函数，变量没公开
    ApplicationInfo info = mPkgInfo.applicationInfo;

    //Resources 是用来获取资源的，而这里获取的资源是在本程序之外的
    //至于为什么这么弄，我搞不懂。
    Resources pRes = getResources();
    AssetManager assmgr =
new AssetManager();
    assmgr.addAssetPath(apkPath);
    Resources res =
new Resources(assmgr, pRes.getDisplayMetrics(), pRes.getConfiguration());

    CharSequence label =
null;
    if (info.labelRes !=
0) {
        try {
            label = res.getText(info.labelRes);
        } catch (Resources.NotFoundException e) {
```



```

        }
    }
    if (label ==
null) {
        label = (info.nonLocalizedLabel !=
null) ?
            info.nonLocalizedLabel : info.packageName;
    }

    //这里就是读取一个 apk 程序的图标

    if (info.icon !=
0){
        Drawable icon = res.getDrawable(info.icon);
        ImageView image = (ImageView) findViewById(R.id.iv_test);
        image.setVisibility(View.VISIBLE);
        image.setImageDrawable(icon);
    }
}

```

9.13、调用Android installer 安装和卸载程序

```

view plaincopy to clipboardprint?
Intent intent = new Intent(Intent.ACTION_VIEW);
    intent.setDataAndType(Uri.fromFile(new File("/sdcard/WorldCupTimer.apk")),
"application/vnd.android.package-archive");
    startActivity(intent); //安装 程序

    Uri packageURI = Uri.parse("package:zy.dnh");
    Intent uninstallIntent = new Intent(Intent.ACTION_DELETE, packageURI);
    startActivity(uninstallIntent); //正常卸载程序
    Intent intent = new Intent(Intent.ACTION_VIEW);
        intent.setDataAndType(Uri.fromFile(new File("/sdcard/WorldCupTimer.apk")),
"application/vnd.android.package-archive");
        startActivity(intent); //安装 程序

        Uri packageURI = Uri.parse("package:zy.dnh");
        Intent uninstallIntent = new Intent(Intent.ACTION_DELETE, packageURI);
        startActivity(uninstallIntent); //正常卸载程序

```

安装程序:



```

Intent i = new Intent(Intent.ACTION_VIEW);
String filePath = "/sdcard/XXX.apk";
i.setDataAndType(Uri.parse("file://" + filePath), "application/vnd.android.package-archive");
context.startActivity(i);

2:
Uri installUri = Uri.fromParts("package", "xxx", null);
returnIt = new Intent(Intent.ACTION_PACKAGE_ADDED, installUri);

```

卸载程序

```

Uri uri = Uri.fromParts("package", strPackageName, null);
Intent it = new Intent(Intent.ACTION_DELETE, uri);
startActivity(it);

```

9.14、后台监控应用程序包的安装&卸载

方法一:

```

public class getBroadcast extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {

        if(Intent.ACTION_PACKAGE_ADDED.equals(intent.getAction())){
            Toast.makeText(context, "有应用被添加", Toast.LENGTH_LONG).show();
        }
        else if(Intent.ACTION_PACKAGE_REMOVED.equals(intent.getAction())){
            Toast.makeText(context, "有应用被删除", Toast.LENGTH_LONG).show();
        }

        else if(Intent.ACTION_PACKAGE_REPLACED.equals(intent.getAction())){
            Toast.makeText(context, "有应用被替换", Toast.LENGTH_LONG).show();
        }

        else if(Intent.ACTION_CAMERA_BUTTON.equals(intent.getAction())){
            Toast.makeText(context, "按键", Toast.LENGTH_LONG).show();
        }

    }

}

public class getBroadcast extends BroadcastReceiver {

    @Override

```




```

public void onReceive(Context context, Intent intent) {

    if(Intent.ACTION_PACKAGE_ADDED.equals(intent.getAction())){
        Toast.makeText(context, "有应用被添加", Toast.LENGTH_LONG).show();
    }
    else if(Intent.ACTION_PACKAGE_REMOVED.equals(intent.getAction())){
        Toast.makeText(context, "有应用被删除", Toast.LENGTH_LONG).show();
    }

    else if(Intent.ACTION_PACKAGE_REPLACED.equals(intent.getAction())){
        Toast.makeText(context, "有应用被替换", Toast.LENGTH_LONG).show();
    }

    else if(Intent.ACTION_CAMERA_BUTTON.equals(intent.getAction())){
        Toast.makeText(context, "按键", Toast.LENGTH_LONG).show();
    }

}

}

```

需要声明的权限如下 AndroidManifest.xml

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="zy.Broadcast"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".Broadcast"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name="getBroadcast" android:enabled="true" >
            <intent-filter>
                <action android:name="android.intent.action.PACKAGE_ADDED"></action>
                <!-- <action android:name="android.intent.action.PACKAGE_CHANGED"></action>-->
                <action android:name="android.intent.action.PACKAGE_REMOVED"></action>
                <action android:name="android.intent.action.PACKAGE_REPLACED"></action>
                <!-- <action android:name="android.intent.action.PACKAGE_RESTARTED"></action>-->
                <!-- <action android:name="android.intent.action.PACKAGE_INSTALL"></action>-->
            </intent-filter>
        </receiver>
    </application>
</manifest>

```

```

        <action android:name="android.intent.action.CAMERA_BUTTON"></action>
        <data android:scheme="package"></data>
    </intent-filter>
</receiver>
</application>
<uses-sdk android:minSdkVersion="3" />

</manifest>
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="zy.Broadcast"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".Broadcast"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name="getBroadcast" android:enabled="true" >
            <intent-filter>
                <action android:name="android.intent.action.PACKAGE_ADDED"></action>
                <!-- <action android:name="android.intent.action.PACKAGE_CHANGED"></action>-->
                <action android:name="android.intent.action.PACKAGE_REMOVED"></action>
                <action android:name="android.intent.action.PACKAGE_REPLACED"></action>
                <!-- <action android:name="android.intent.action.PACKAGE_RESTARTED"></action>-->
                <!-- <action android:name="android.intent.action.PACKAGE_INSTALL"></action>-->
                <action android:name="android.intent.action.CAMERA_BUTTON"></action>
                <data android:scheme="package"></data>
            </intent-filter>
        </receiver>
    </application>
    <uses-sdk android:minSdkVersion="3" />

</manifest>

```

方法二:

通过阅读 Android SDK 里关于 `intent.action` 这部分里面的描述,我们可以找到一些与 `package` 相关的系统广播

`android.intent.action.PACKAGE_ADDED`



```
android.intent.action.PACKAGE_CHANGED
android.intent.action.PACKAGE_DATA_CLEARED
android.intent.action.PACKAGE_INSTALL
android.intent.action.PACKAGE_REMOVED
android.intent.action.PACKAGE_REPLACED
android.intent.action.PACKAGE_RESTARTED
android.intent.action.PACKAGE_ADDED
android.intent.action.PACKAGE_CHANGED
android.intent.action.PACKAGE_DATA_CLEARED
android.intent.action.PACKAGE_INSTALL
android.intent.action.PACKAGE_REMOVED
android.intent.action.PACKAGE_REPLACED
android.intent.action.PACKAGE_RESTARTED
```

其中 ACTION_PACKAGE_ADDED 在 SDK 里的描述是：

Broadcast Action: A new application package has been installed on the device.

ACTION_PACKAGE_REMOVED 在 SDK 里的描述是：

Broadcast Action: An existing application package has been removed from the device.

ACTION_PACKAGE_REPLACED 在 SDK 里的描述是：

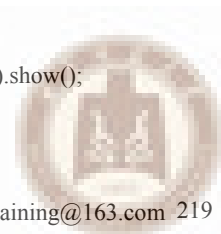
Broadcast Action: A new version of an application package has been installed, replacing an existing version that was previously installed.

通过这三个广播消息 我们已经可以监控到 Android 应用程序的安装和删除
详细的实现代码如下

getBroadcast.java

```
package zy.Broadcast;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;
public class getBroadcast extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {

        if(Intent.ACTION_PACKAGE_ADDED.equals(intent.getAction())){
            Toast.makeText(context, "有应用被添加", Toast.LENGTH_LONG).show();
        }
    }
}
```



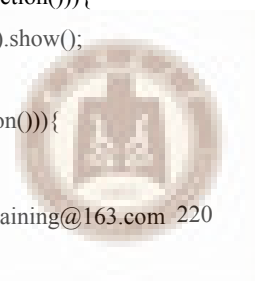
```

        else if(Intent.ACTION_PACKAGE_REMOVED.equals(intent.getAction())){
            Toast.makeText(context, "有应用被删除", Toast.LENGTH_LONG).show();
        }
        /* else if(Intent.ACTION_PACKAGE_CHANGED.equals(intent.getAction())){
            Toast.makeText(context, "有应用被改变", Toast.LENGTH_LONG).show();
        }*/
        else if(Intent.ACTION_PACKAGE_REPLACED.equals(intent.getAction())){
            Toast.makeText(context, "有应用被替换", Toast.LENGTH_LONG).show();
        }
        /* else if(Intent.ACTION_PACKAGE_RESTARTED.equals(intent.getAction())){
            Toast.makeText(context, "有应用被重启", Toast.LENGTH_LONG).show();
        }*/
        /* else if(Intent.ACTION_PACKAGE_INSTALL.equals(intent.getAction())){
            Toast.makeText(context, "有应用被安装", Toast.LENGTH_LONG).show();
        }*/
    }
}

package zy.Broadcast;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;
public class getBroadcast extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {

        if(Intent.ACTION_PACKAGE_ADDED.equals(intent.getAction())){
            Toast.makeText(context, "有应用被添加", Toast.LENGTH_LONG).show();
        }
        else if(Intent.ACTION_PACKAGE_REMOVED.equals(intent.getAction())){
            Toast.makeText(context, "有应用被删除", Toast.LENGTH_LONG).show();
        }
        /* else if(Intent.ACTION_PACKAGE_CHANGED.equals(intent.getAction())){
            Toast.makeText(context, "有应用被改变", Toast.LENGTH_LONG).show();
        }*/
        else if(Intent.ACTION_PACKAGE_REPLACED.equals(intent.getAction())){
            Toast.makeText(context, "有应用被替换", Toast.LENGTH_LONG).show();
        }
        /* else if(Intent.ACTION_PACKAGE_RESTARTED.equals(intent.getAction())){
            Toast.makeText(context, "有应用被重启", Toast.LENGTH_LONG).show();
        }*/
        /* else if(Intent.ACTION_PACKAGE_INSTALL.equals(intent.getAction())){

```



```

        Toast.makeText(context, "有应用被安装", Toast.LENGTH_LONG).show();
    }*/

}

}

```

然后在 AndroidManifest.xml 中声明这几个 Action 的<intent-filter>即可在系统里捕获这些广播消息
具体的源代码如下

```

view plaincopy to clipboardprint?
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="zy.Broadcast"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".Broadcast"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name="getBroadcast" android:enabled="true" >
            <intent-filter>
                <action android:name="android.intent.action.PACKAGE_ADDED"></action>
                <!--                                <action
android:name="android.intent.action.PACKAGE_CHANGED"></action>-->
                <action
android:name="android.intent.action.PACKAGE_REMOVED"></action>
                <action
android:name="android.intent.action.PACKAGE_REPLACED"></action>
                <!--                                <action
android:name="android.intent.action.PACKAGE_RESTARTED"></action>-->
                <!--                                <action
android:name="android.intent.action.PACKAGE_INSTALL"></action>-->
                <data android:scheme="package"></data>
            </intent-filter>
        </receiver>
    </application>
    <uses-sdk android:minSdkVersion="7" />

```



```

</manifest>
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="zy.Broadcast"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".Broadcast"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name="getBroadcast" android:enabled="true" >
            <intent-filter>
                <action android:name="android.intent.action.PACKAGE_ADDED"></action>
                <!--> <action
android:name="android.intent.action.PACKAGE_CHANGED"></action>-->
                <action
android:name="android.intent.action.PACKAGE_REMOVED"></action>
                <action
android:name="android.intent.action.PACKAGE_REPLACED"></action>
                <!--> <action
android:name="android.intent.action.PACKAGE_RESTARTED"></action>-->
                <!--> <action
android:name="android.intent.action.PACKAGE_INSTALL"></action>-->
                <data android:scheme="package"></data>
            </intent-filter>
        </receiver>
    </application>
    <uses-sdk android:minSdkVersion="7" />
</manifest>

```

把程序安装之后，系统就会注册这个 BroadcastReceiver

然后有应用安装删除替换操作时时，就会弹出 Toast 提示

删除应用





添加应用



有应用被替换





以上这样，我们就可以实现监控 Android 应用程序的安装过程

至于拦截安装过程，我也正在研究中，大家有好的 idea 可以与我 分享，谢谢

9.15、显示应用详细列表

```
Uri uri = Uri.parse("market://details?id=app_id");
Intent it = new Intent(Intent.ACTION_VIEW, uri);
startActivity(it);
//where app_id is the application ID, find the ID
//by clicking on your application on Market home
//page, and notice the ID from the address bar
```

刚才找 app id 未果，结果发现用 package name 也可以

```
Uri uri = Uri.parse("market://details?id=<packagename>");
```

这个简单多了

9.16、寻找应用

```
Uri uri = Uri.parse("market://search?q=pname:pkg_name");
Intent it = new Intent(Intent.ACTION_VIEW, uri);
```




```
startActivity(it);  
//where pkg_name is the full package path for an application
```

9.17、注册一个BroadcastReceiver

```
registerReceiver(mMasterResetReciever, new IntentFilter(" OMS.action.MASTERRESET" ));  
private BroadcastReceiver mMasterResetReciever = new BroadcastReceiver() {  
    public void onReceive(Context context, Intent intent){  
        String action = intent.getAction();  
        if(" oms.action.MASTERRESET".equals(action)){  
            RecoverDefaultConfig();  
        }  
    }  
};
```

9.18、打开另一程序

```
Intent i = new Intent();  
    ComponentName cn = new ComponentName("com.yellowbook.android2",  
        "com.yellowbook.android2.AndroidSearch");  
i.setComponent(cn);  
i.setAction("android.intent.action.MAIN");  
startActivityForResult(i, RESULT_OK);
```

9.19、播放默认铃声

```
Uri alert = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_ALARM);  
mMediaPlayer = new MediaPlayer();  
mMediaPlayer.setDataSource(this, alert);  
final AudioManager audioManager = (AudioManager) getSystemService(Context.AUDIO_SERVICE);  
if (audioManager.getStreamVolume(AudioManager.STREAM_ALARM) != 0) {  
    player.setAudioStreamType(AudioManager.STREAM_ALARM);  
    player.setLooping(true);  
    player.prepare();  
    player.start();  
}
```



9.20、设置默认来电铃声

```
public void setMyRingtone()
{
    File k = new File("/sdcard/Shall We Talk.mp3"); // 设置歌曲路径
    ContentValues values = new ContentValues();
    values.put(MediaStore.MediaColumns.DATA, k.getAbsolutePath());
    values.put(MediaStore.MediaColumns.TITLE, "Shall We Talk");
    values.put(MediaStore.MediaColumns.SIZE, 8474325);
    values.put(MediaStore.MediaColumns.MIME_TYPE, "audio/mp3");
    values.put(MediaStore.Audio.Media.ARTIST, "Madonna");
    values.put(MediaStore.Audio.Media.DURATION, 230);
    values.put(MediaStore.Audio.Media.IS_RINGTONE, true);
    values.put(MediaStore.Audio.Media.IS_NOTIFICATION, false);
    values.put(MediaStore.Audio.Media.IS_ALARM, false);
    values.put(MediaStore.Audio.Media.IS_MUSIC, false);
    // Insert it into the database
    Uri uri = MediaStore.Audio.Media.getContentUriForPath(k.getAbsolutePath());
    Uri newUri = this.getContentResolver().insert(uri, values);
    RingtoneManager.setActualDefaultRingtoneUri(this, RingtoneManager.TYPE_RINGTONE, newUri);
};

public void setMyRingtone()
{
    File k = new File("/sdcard/Shall We Talk.mp3"); // 设置歌曲路径
    ContentValues values = new ContentValues();
    values.put(MediaStore.MediaColumns.DATA, k.getAbsolutePath());
    values.put(MediaStore.MediaColumns.TITLE, "Shall We Talk");
    values.put(MediaStore.MediaColumns.SIZE, 8474325);
    values.put(MediaStore.MediaColumns.MIME_TYPE, "audio/mp3");
    values.put(MediaStore.Audio.Media.ARTIST, "Madonna");
    values.put(MediaStore.Audio.Media.DURATION, 230);
    values.put(MediaStore.Audio.Media.IS_RINGTONE, true);
    values.put(MediaStore.Audio.Media.IS_NOTIFICATION, false);
    values.put(MediaStore.Audio.Media.IS_ALARM, false);
    values.put(MediaStore.Audio.Media.IS_MUSIC, false);
    // Insert it into the database
    Uri uri = MediaStore.Audio.Media.getContentUriForPath(k.getAbsolutePath());
    Uri newUri = this.getContentResolver().insert(uri, values);
    RingtoneManager.setActualDefaultRingtoneUri(this, RingtoneManager.TYPE_RINGTONE, newUri);
};
```

需要的权限

```
<uses-permission android:name="android.permission.WRITE_SETTINGS"></uses-permission>
```

作者: craining (曲阜师范大学) 个人主页: <http://craining.blog.163.com/> 邮箱: craining@163.com 226



9.21、位图旋转

(来自 Android123)

今天有关 Android 游戏开发的基础，我们说下 Bitmap 相关的实用操作，这里我们就说下位图旋转。在 Android 中图形的旋转和变化提供了方便的矩阵 Maxtrix 类,Maxtrix 类的 setRotate 方法接受图形的变换角度和缩放，最终 Bitmap 类的 createBitmap 方法中其中的重载函数，可以接受 Maxtrix 对象，方法原型如下 `public static Bitmap createBitmap (Bitmap source, int x, int y, int width, int height, Matrix m, boolean filter)`

参数的具体意思

source 源 bitmap 对象

x 源坐标 x 位置

y 源坐标 y 位置

width 宽度

height 高度

m 接受的 maxtrix 对象，如果没有可以设置为 null

filter 该参数仅对 maxtrix 包含了超过一个翻转才有效。

下面 Android123 给大家一个比较经典的例子，rotate 方法是静态方法可以直接调用，参数为源 Bitmap 对象，参数二为旋转的角度，从 0~360，返回值为新的 Bitmap 对象。其中具体的宽高可以调整。

```
public static Bitmap rotate(Bitmap b, int degrees) {
    if (degrees != 0 && b != null) {
        Matrix m = new Matrix();
        m.setRotate(degrees,
            (float) b.getWidth() / 2, (float) b.getHeight() / 2);
        try {
            Bitmap b2 = Bitmap.createBitmap(
                b, 0, 0, b.getWidth(), b.getHeight(), m, true);
            if (b != b2) {
                b.recycle(); //Android 开发网再次提示 Bitmap 操作完应该显示的释放
                b = b2;
            }
        } catch (OutOfMemoryError ex) {
            // Android123 建议大家如何出现了内存不足异常，最好 return 原始的 bitmap 对象。
        }
    }
    return b;
}
```



9.22、手机震动控制

android api 提供了 Vibrator 对象全权负责震动的处理。

其构造形式为: `vibrate(long[] pattern, int repeat)`; 其中 `pattern` 参数取值为: `{x,y,z,t}`, `x,y,z` 分别表示了手机在不同方向上震动的幅度, `t` 表示了一次震动的时长。参数 `repeat=0` 时手机会一直保持震动。

构造方法:

```
Vibrator vb=(Vibrator)getApplicationContext().getSystemService(Service.VIBRATOR_SERVICE);
```

产生震动: `vb.vibrate(new long[]{x,y,z,t},-1);`

取消震动: `vb.cancel();`

添加权限: `<uses-permission android:name="android.permission.VIBRATE" />`

代码片段:

```
btn.setOnClickListener(new OnClickListener()
{
    public void onClick(View v)
    {
        if (checkVb.isChecked())
        {
            /*设定震动的周期*/
            vb.vibrate( new long[]{1000,50,1000,50,1000},-1);
            ///others
        }
        else
        {
            /*取消震动*/
            vb.cancel();
            //.....
        }
    }
});
```

9.23、Sensor2D感应实例

(本文来自 Android123)

有关 Android 游戏开发中的 Sensor 感应示例今天我们将一起来讨论,对于目前最新的 Android 2.2 平台而言仍然没有具体的感应判断逻辑,下面我们一起定义下常用的感应动作事件。首先 Android123 提醒大家由于是三轴的立体空间感应所以相对于轨迹球、导航键的上下左右外,还提供了前后的感应,所以我们定义最基本的六种空间方向。 `public static final int CWJ_UP = 0;`

`public static final int CWJ_DOWN = 1;`

`public static final int CWJ_LEFT = 2;`

`public static final int CWJ_RIGHT = 4;`



```
public static final int CWJ_FORWARD = 8; //向前
public static final int CWJ_BACKWARD = 16; //向后
```

下面我们做精确的角度旋转修正值定义，我们用到 yaw、pitch 和 roll，相信学过 3D 开发的网友不会对这些陌生的，我们就把他们对应为绕 y、x、z 轴的角度好了，如果你们没有学过 3D 相关的知识这里 Android 开发网推荐大家可以通过 Cube 例子自定义 Render 来观察这三个值对应立方体的旋转角度。

Yaw 在 (0,0,0) 中，以 xOz 的坐标平面中围绕 y 轴旋转，如果是负角则我们定义为 CWJ_YAW_LEFT 即往左边倾斜，同理我们定义如下：

```
public static final int CWJ_YAW_LEFT = 0;
public static final int CWJ_YAW_RIGHT = 1;
public static final int CWJ_PITCH_UP = 2;
public static final int CWJ_PITCH_DOWN = 4;
public static final int CWJ_ROLL_LEFT = 8;
public static final int CWJ_ROLL_RIGHT = 16;
```

我们通过加速感应器可以获得 SensorEvent 的四个值，今天 Android123 给大家一个简单示例，不考虑其他因素，在 public int accuracy、public Sensor sensor、public long timestamp 和 public final float[] values 中，我们获取 values 的浮点数组来判断方向。

```
int nAndroid123=CWJ_UP //向上
    float ax = values[0];
    float ay = values[1];
    float az = values[2];

    float absx = Math.abs(ax);
    float absy = Math.abs(ay);
    float absz = Math.abs(az);

    if (absx > absy && absx > absz) {

        if (ax > 0) {
            nAndroid123 = CWJ_RIGHT;
        } else {
            nAndroid123 = CWJ_LEFT;
        }
    } else if (absy > absx && absy > absz) {

        if (ay > 0) {
            nAndroid123= CWJ_FORWARD;
        } else {
            nAndroid123= CWJ_BACKWARD;
        }
    } else if (absz > absx && absz > absy) {

        if (az > 0) {
```



```

        nAndroid123 = CWJ_UP;
    } else {
        nAndroid123 = CWJ_DOWN;
    }
} else {
    nAndroid123 = CWJ_UNKNOWN;
}
}

```

9.24、运用java mail包实现发Gmail邮件

第一个类：MailSenderInfo.java

```

package com.util.mail;

/**
 * 发送邮件需要使用的基本信息
 */

import java.util.Properties;

public class MailSenderInfo {
    // 发送邮件的服务器的 IP 和端口
    private String mailServerHost;
    private String mailServerPort = "25";
    // 邮件发送者的地址
    private String fromAddress;
    // 邮件接收者的地址
    private String toAddress;
    // 登陆邮件发送服务器的用户名和密码
    private String userName;
    private String password;
    // 是否需要身份验证
    private boolean validate = false;
    // 邮件主题
    private String subject;
    // 邮件的文本内容
    private String content;
    // 邮件附件的文件名
    private String[] attachFileNames;

    /**
     * 获得邮件会话属性
     */

    public Properties getProperties(){
        Properties p = new Properties();
        p.put("mail.smtp.host", this.mailServerHost);
        p.put("mail.smtp.port", this.mailServerPort);
        p.put("mail.smtp.auth", validate ? "true" : "false");
    }
}

```



```
    return p;
}
public String getMailServerHost() {
    return mailServerHost;
}
public void setMailServerHost(String mailServerHost) {
    this.mailServerHost = mailServerHost;
}
public String getMailServerPort() {
    return mailServerPort;
}
public void setMailServerPort(String mailServerPort) {
    this.mailServerPort = mailServerPort;
}
public boolean isValid() {
    return validate;
}
public void setValidate(boolean validate) {
    this.validate = validate;
}
public String[] getAttachFileNames() {
    return attachFileNames;
}
public void setAttachFileNames(String[] fileNames) {
    this.attachFileNames = fileNames;
}
public String getFromAddress() {
    return fromAddress;
}
public void setFromAddress(String fromAddress) {
    this.fromAddress = fromAddress;
}
public String getPassword() {
    return password;
}
public void setPassword(String password) {
    this.password = password;
}
public String getToAddress() {
    return toAddress;
}
public void setToAddress(String toAddress) {
    this.toAddress = toAddress;
}
```



```
public String getUserName() {
    return userName;
}
public void setUserName(String userName) {
    this.userName = userName;
}
public String getSubject() {
    return subject;
}
public void setSubject(String subject) {
    this.subject = subject;
}
public String getContent() {
    return content;
}
public void setContent(String textContent) {
    this.content = textContent;
}
}
```

第二个类: SimpleMailSender.java

```
package com.util.mail;

import java.util.Date;
import java.util.Properties;
import javax.mail.Address;
import javax.mail.BodyPart;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Multipart;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeBodyPart;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;

/**
 * 简单邮件（不带附件的邮件）发送器
 */
public class SimpleMailSender {
    /**
     * 以文本格式发送邮件
     * @param mailInfo 待发送的邮件的信息
     */
}
```




```
public boolean sendTextMail(MailSenderInfo mailInfo) {
    // 判断是否需要身份认证
    MyAuthenticator authenticator = null;
    Properties pro = mailInfo.getProperties();
    if (mailInfo.isValidate()) {
        // 如果需要身份认证，则创建一个密码验证器
        authenticator = new MyAuthenticator(mailInfo.getUserName(), mailInfo.getPassword());
    }
    // 根据邮件会话属性和密码验证器构造一个发送邮件的 session
    Session sendMailSession = Session.getDefaultInstance(pro,authenticator);
    try {
        // 根据 session 创建一个邮件消息
        Message mailMessage = new MimeMessage(sendMailSession);
        // 创建邮件发送者地址
        Address from = new InternetAddress(mailInfo.getFromAddress());
        // 设置邮件消息的发送者
        mailMessage.setFrom(from);
        // 创建邮件的接收者地址，并设置到邮件消息中
        Address to = new InternetAddress(mailInfo.getToAddress());
        mailMessage.setRecipient(Message.RecipientType.TO,to);
        // 设置邮件消息的主题
        mailMessage.setSubject(mailInfo.getSubject());
        // 设置邮件消息发送的时间
        mailMessage.setSentDate(new Date());
        // 设置邮件消息的主要内容
        String mailContent = mailInfo.getContent();
        mailMessage.setText(mailContent);
        // 发送邮件
        Transport.send(mailMessage);
        return true;
    } catch (MessagingException ex) {
        ex.printStackTrace();
    }
    return false;
}

/**
 * 以 HTML 格式发送邮件
 * @param mailInfo 待发送的邮件信息
 */
public static boolean sendHtmlMail(MailSenderInfo mailInfo){
    // 判断是否需要身份认证
    MyAuthenticator authenticator = null;
    Properties pro = mailInfo.getProperties();
```



```

//如果需要身份认证，则创建一个密码验证器
if (mailInfo.isValidate()) {
    authenticator = new MyAuthenticator(mailInfo.getUserName(), mailInfo.getPassword());
}
// 根据邮件会话属性和密码验证器构造一个发送邮件的 session
Session sendMailSession = Session.getDefaultInstance(pro,authenticator);
try {
    // 根据 session 创建一个邮件消息
    Message mailMessage = new MimeMessage(sendMailSession);
    // 创建邮件发送者地址
    Address from = new InternetAddress(mailInfo.getFromAddress());
    // 设置邮件消息的发送者
    mailMessage.setFrom(from);
    // 创建邮件的接收者地址，并设置到邮件消息中
    Address to = new InternetAddress(mailInfo.getToAddress());
    // Message.RecipientType.TO 属性表示接收者的类型为 TO
    mailMessage.setRecipient(Message.RecipientType.TO,to);
    // 设置邮件消息的主题
    mailMessage.setSubject(mailInfo.getSubject());
    // 设置邮件消息发送的时间
    mailMessage.setSentDate(new Date());
    // MiniMultipart 类是一个容器类，包含 MimeBodyPart 类型的对象
    Multipart mainPart = new MimeMultipart();
    // 创建一个包含 HTML 内容的 MimeBodyPart
    BodyPart html = new MimeBodyPart();
    // 设置 HTML 内容
    html.setContent(mailInfo.getContent(), "text/html; charset=utf-8");
    mainPart.addBodyPart(html);
    // 将 MiniMultipart 对象设置为邮件内容
    mailMessage.setContent(mainPart);
    // 发送邮件
    Transport.send(mailMessage);
    return true;
} catch (MessagingException ex) {
    ex.printStackTrace();
}
return false;
}
}

```

第三个类：MyAuthenticator.java

```
package com.util.mail;
```

```
import javax.mail.*;
```



```

public class MyAuthenticator extends Authenticator{
    String userName=null;
    String password=null;

    public MyAuthenticator(){
    }
    public MyAuthenticator(String username, String password) {
        this.userName = username;
        this.password = password;
    }
    protected PasswordAuthentication getPasswordAuthentication(){
        return new PasswordAuthentication(userName, password);
    }
}

```

主函数代码

```

public static void main(String[] args){
    //这个类主要是设置邮件
    MailSenderInfo mailInfo = new MailSenderInfo();
    mailInfo.setMailServerHost("smtp.163.com");
    mailInfo.setMailServerPort("25");
    mailInfo.setValidate(true);
    mailInfo.setUserName("han2000lei@163.com");
    mailInfo.setPassword("*****");//您的邮箱密码
    mailInfo.setFromAddress("han2000lei@163.com");
    mailInfo.setToAddress("han2000lei@163.com");
    mailInfo.setSubject("设置邮箱标题");
    mailInfo.setContent("设置邮箱内容");
    //这个类主要来发送邮件
    SimpleMailSender sms = new SimpleMailSender();
    sms.sendTextMail(mailInfo);//发送文体格式
    sms.sendHtmlMail(mailInfo);//发送 html 格式
}

```

注意：

- 1、使用此代码你可以完成你的 javamail 的邮件发送功能。三个类缺一不可。
- 2、这三个类我打包是用的 **com.util.mail** 包，如果不喜欢，你可以自己改，但三个类文件必须在同一个包中
- 3、不要使用你刚刚注册过的邮箱在程序中发邮件，如果你的 163 邮箱是刚注册不久，那你就不要使用“**smtp.163.com**”。因为你发不出去。刚注册的邮箱是不会给你这种权限的，也就是你不能通过验证。要使用你经常用的邮箱，而且时间比较长的。
- 4、另一个问题就是 **mailInfo.setMailServerHost("smtp.163.com");** 与 **mailInfo.setFromAddress("han2000lei@163.com");**这两句话。即如果你使用 163smtp 服务器，那么发送邮件地址就必须用 163 的邮箱，如果不的话，是不会发送成功的。
- 5、关于 javamail 验证错误的问题，网上的解释有很多，但我看见的只有一个。就是我的第

三个类。你只要复制全了代码，我想是不会有问题的。

9.26、Android键盘响应

按键抬起:

```
public boolean onKeyUp(int keyCode, KeyEvent event) {
    switch (keyCode) {
        case KeyEvent.KEYCODE_DPAD_LEFT:
            //添加操作
            break;

        case KeyEvent.KEYCODE_DPAD_RIGHT:
            //添加操作

            break;

        case KeyEvent.KEYCODE_1:
            //添加操作

            break;
    }
    return true;
}
```

按键按下:

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if (keyCode == KeyEvent.KEYCODE_BACK && event.getRepeatCount() == 0) {
        // do something on back.
        return true;
    }

    return super.onKeyDown(keyCode, event);
}
```

返回键:

```
@Override
public void onBackPressed() {
    // do something on back.
    return;
}
```

按键长按:

```
@Override
public boolean onKeyLongPress(int keyCode, KeyEvent event) {
    if (keyCode == KeyEvent.KEYCODE_CALL) {
        // a long press of the call key.
    }
}
```



```
        // do our work, returning true to consume it.  by
        // returning true, the framework knows an action has
        // been performed on the long press, so will set the
        // canceled flag for the following up event.
        return true;
    }
    return super.onKeyLongPress(keyCode, event);
}
```

```
@Override
public boolean onKeyUp(int keyCode, KeyEvent event) {
    if (keyCode == KeyEvent.KEYCODE_CALL && event.isTracking()
        && !event.isCanceled()) {
        // if the call key is being released, AND we are tracking
        // it from an initial key down, AND it is not canceled,
        // then handle it.
        return true;
    }
    return super.onKeyUp(keyCode, event);
}
```

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if (keyCode == KeyEvent.KEYCODE_0) {
        // this tells the framework to start tracking for
        // a long press and eventual key up.  it will only
        // do so if this is the first down (not a repeat).
        event.startTracking();
        return true;
    }
    return super.onKeyDown(keyCode, event);
}
```

```
@Override
public boolean dispatchKeyEvent(KeyEvent event) {
    if (event.getKeyCode() == KeyEvent.KEYCODE_BACK) {
        if (event.getAction() == KeyEvent.ACTION_DOWN
            && event.getRepeatCount() == 0) {

            // Tell the framework to start tracking this event.
            getKeyDispatcherState().startTracking(event, this);
            return true;

        } else if (event.getAction() == KeyEvent.ACTION_UP) {
```



```

        getKeyDispatcherState().handleUpEvent(event);
        if (event.isTracking() && !event.isCanceled()) {

            // DO BACK ACTION HERE
            return true;
        }
    }
    return super.dispatchKeyEvent(event);
} else {
    return super.dispatchKeyEvent(event);
}
}

```

9.27、后台监听某个按键

It's relatively easy to add headset button support to your application. For example you want to play/pause media playback in your super media player.

But here are a few moments that you should take into consideration:

- 1、 You should register your broadcast receiver inside your application (not in manifest file). Otherwise Google Music player will catch your broadcast and aboard it.
<http://android.git.kernel.org/?p=platform/packages/apps/Music.git;a=blob;f=src/com/android/music/MediaButtonIntentReceiver.java>
- 2、 Your IntentFilter priority should be higher that other media players priorities in your phone:) That' s kind of tricky thing.

The code snippet will look like:

```

MediaButtonIntentReceiver mMediaButtonReceiver = new MediaButtonIntentReceiver
IntentFilter mediaFilter = new IntentFilter(Intent.ACTION_MEDIA_BUTTON);
mediaFilter.setPriority(MEDIA_BUTTON_INTENT_EMPIRICAL_PRIORITY_VALUE);
registerReceiver(mMediaButtonReceiver, mediaFilter);

```

And MediaButtonIntentReceiver.java

```

public class MediaButtonIntentReceiver extends BroadcastReceiver {

    public MediaButtonIntentReceiver() {

        super ();
    }

    @Override

```



```

public void onReceive(Context context, Intent intent) {
    String intentAction = intent.getAction();
    if (!Intent.ACTION_MEDIA_BUTTON.equals(intentAction)) {
        return;
    }

    KeyEvent event = (KeyEvent)intent.getParcelableExtra(Intent.EXTRA_KEY_EVENT);
    if(event == null) {

        return;
    }

    int action=event.getAction();

    if(action == KeyEvent.ACTION_DOWN) {

        //do something

    }

    abortBroadcast();
}
}

```

1.Don't forget to unregister your broadcast receiver

2.Add “android.permission.BLUETOOTH” permission if you want to support bluetooth headset

9.28、Vector用法

java.util.vector 中的 vector 的详细用法

ArrayList 会比 Vector 快，他是非同步的，如果设计涉及到多线程，还是用 Vector 比较好一些

```
import java.util.*;
```

```
/**
```

```
* 演示 Vector 的使用。包括 Vector 的创建、向 Vector 中添加元素、从 Vector 中删除元素、
```

```
* 统计 Vector 中元素的个数和遍历 Vector 中的元素。
```

```
*/
```

```
public class VectorDemo{
```

```
public static void main(String[] args){
```

```
//Vector 的创建
```

```
//使用 Vector 的构造方法进行创建
```

```
Vector v = new Vector(4);
```



```

//向 Vector 中添加元素
//使用 add 方法直接添加元素
v.add("Test0");
v.add("Test1");
v.add("Test0");
v.add("Test2");
v.add("Test2");
//从 Vector 中删除元素
v.remove("Test0"); //删除指定内容的元素
v.remove(0); //按照索引号删除元素
//获得 Vector 中已有元素的个数
int size = v.size();
System.out.println("size:" + size);
//遍历 Vector 中的元素
for(int i = 0; i < v.size(); i++){
    System.out.println(v.get(i));
}
}
}

```

Vector 类提供了实现可增长数组的功能，随着更多元素加入其中，数组变的更大。在删除一些元素之后，数组变小。

Vector 有三个构造函数，

```

public Vector(int initialCapacity,int capacityIncrement)
public Vector(int initialCapacity)
public Vector()

```

Vector 运行时创建一个初始的存储容量 initialCapacity，存储容量是以 capacityIncrement 变量定义的增量增长。初始的存储容量和 capacityIncrement 可以在 Vector 的构造函数中定义。第二个构造函数只创建初始存储容量。第三个构造函数既不指定初始的存储容量也不指定 capacityIncrement。

Vector 类提供的访问方法支持类似数组运算和与 Vector 大小相关的运算。类似数组的运算允许向量中增加，删除和插入元素。它们也允许测试矢量的内容和检索指定的元素，与大小相关的运算允许判定字节大小和矢量中元素不数目。

现针对经常用到的对向量增，删，插功能举例描述：

addElement(Object obj)

把组件加到向量尾部，同时大小加 1，向量容量比以前大 1

insertElementAt(Object obj, int index)

把组件加到所定索引处，此后的内容向后移动 1 个单位

setElementAt(Object obj, int index)

把组件加到所定索引处，此处的内容被代替。



removeElement(Object obj)

把向量中含有本组件内容移走。

removeAllElements()

把向量中所有组件移走，向量大小为 0。

例如：

```
import java.lang.System;
```

```
import java.util.Vector;
```

```
import java.util.Enumeration;
```

```
public class Avector{
    public static void main(String args[]) {

        Vector v=new Vector();
        v.addElement("one");
        addElement("two");
        v.addElement("three");
        v.insertElementAt("zero",0);
        v.insertElementAt("oop",3);
        v.setElementAt("three",3);
        v.setElementAt("four",4);
        v.removeAllElements();
    }
}
```

Vector 中的变化情况：

1. one	2. one	3. one	4. zero	5. zero	6. zero	7. zero	8.
two	two	one	one	one		three	two
two	two	two			three	oop	three
three				three	three	four	另

外，Vector 在参数传递中发挥着举足轻重的作用。在 Applet 中有一块画布(Canvas) 和一个(Panel)，而 Panel 中放着用户要输入的信息，根据这些信息把参数传递到 canvas 中，这时在 Java 中用一个接口（Interface），而在接口中需用一个 Vector 去传递这些参数。

另外，在一个类向另一个类参数传递就可以用这种方法。 例如：

```
import java.util.Vector
```

```
interface codeselect{
```

```
    Vector codeselect=new Vector();
```

```
} 显示数学信息
```

Vector(0)存入学生编号

Vector(1)存入学科

在 Panel 中当用户在 TextField 和 Choice 中选择自己所要求的内容，程序中通过事件响应把值传到向量 Vector 中。



9.29、Cursor

使用过 SQLite 数据库的开发者对 Cursor 应该不陌生，如果你是搞.net 开发你大可以把 Cursor 理解成 ADO.NET 中的数据集合相当于 dataReader。今天特地将它单独拿出来谈，加深自己和大家对 Android 中使用 Cursor 的理解。

关于 Cursor

在你理解和使用 Android Cursor 的时候你必须先知道关于 Cursor 的几件事情：

Cursor 是每行的集合。

使用 moveToFirst() 定位第一行。

你必须知道每一列的名称。

你必须知道每一列的数据类型。

Cursor 是一个随机的数据源。

所有的数据都是通过下标取得。

关于 Cursor 的重要方法：

`close()`

关闭游标，释放资源

`copyStringToBuffer(int columnIndex, CharArrayBuffer buffer)`

在缓冲区中检索请求的列的文本，并将其存储

`getColumnCount()`

返回所有列的总数

`getColumnIndex(String columnName)`

返回指定列的名称，如果不存在返回-1

`getColumnIndexOrThrow(String columnName)`

从零开始返回指定列名称，如果不存在将抛出 `IllegalArgumentException` 异常。

`getColumnName(int columnIndex)`

从给定的索引返回列名

`getColumnNames()`

返回一个字符串数组的列名

`getCount()`

返回 Cursor 中的行数

`moveToFirst()`

移动光标到第一行

`moveToLast()`

移动光标到最后一行

`moveToNext()`

移动光标到下一行

`moveToPosition(int position)`

移动光标到一个绝对的位置

`moveToPrevious()`

移动光标到上一行



下面来看看一小段代码:

```
if (cur.moveToFirst() == false)
{
//为空的 Cursor
return;
}
```

访问 Cursor 的下标获得其中的数据

```
int nameColumnIndex = cur.getColumnIndex(People.NAME);
String name = cur.getString(nameColumnIndex);
```

现在让我们看看如何循环 Cursor 取出我们需要的数据

```
while(cur.moveToNext())
{
//光标移动成功
//把数据取出
}
```

当 cur.moveToNext() 为假时将跳出循环, 即 Cursor 数据循环完毕。

如果你喜欢用 for 循环而不想用 While 循环可以使用 Google 提供的几下方法:

isBeforeFirst()

返回游标是否指向之前第一行的位置

isAfterLast()

返回游标是否指向第最后一行的位置

isClosed()

如果返回 true 即表示该游戏标已关闭

有了以上的方法, 可以如此取出数据

AbstractCursor

```
for(cur.moveToFirst();!cur.isAfterLast();cur.moveToNext())
{
    int nameColumn = cur.getColumnIndex(People.NAME);
    int phoneColumn = cur.getColumnIndex(People.NUMBER);
    String name = cur.getString(nameColumn);
    String phoneNumber = cur.getString(phoneColumn);
}
```



Tip:在 Android 查询数据是通过 Cursor 类来实现的。当我们使用 SQLiteDatabase.query()方法时, 就会得到 Cursor 对象, Cursor 所指向的就是每一条数据。结合 ADO.net 的知识可能好理解一点。

Cursor 位于 android.database.Cursor 类, 可见出它的设计是基于数据库服务产生的。

另外, 还有几个已知的子类, 分别为:

AbstractWindowedCursor

CrossProcessCursor

CursorWrapper

MatrixCursor

MergeCursor

MockCursor

SQLiteCursor

具体详细的使用方法和解释可以去参照 API, 这里就不过多讲述。

9.30、把一个字符串写进文件

方法一:

```
/**
 * 向 file 中写数据
 *
 * @param fileName
 * @param toSave
 * @return
 */
public boolean save(String fileName, String toSave) {
    Properties properties = new Properties();

    properties.put("string", toSave);
    try {
        FileOutputStream stream = this.openFileOutput(fileName,
            Context.MODE_WORLD_WRITEABLE);
        properties.store(stream, "");
    } catch (FileNotFoundException e) {
        return false;
    } catch (IOException e) {
        return false;
    }

    return true;
}
```



```
}
```

方法二:

```
public void writefile(String str,String path )
{
    File file;
    FileOutputStream out;
    try {
        //创建文件
        file = new File(path);
        file.createNewFile();

        //打开文件 file 的 OutputStream
        out = new FileOutputStream(file);
        String infoToWrite = str;
        //将字符串转换成 byte 数组写入文件
        out.write(infoToWrite.getBytes());
        //关闭文件 file 的 OutputStream
        out.close();

    } catch (IOException e) {
        //将出错信息打印到 Logcat
        DisplayToast(e.toString());
    }
}
```

9.31、把文件内容读出到一个字符串

方法一:

```
/**
 * 读取 file 中的数据
 *
 * @param fileName
 * @return
 */
public String load(String fileName) {
    Properties properties = new Properties();
    try {
        FileInputStream stream = this.openFileInput(fileName);
        properties.load(stream);
    } catch (FileNotFoundException e) {
        return null;
    }
}
```



```

    } catch (IOException e) {
        return null;
    }

    return properties.get("string").toString();
}

```

方法二:

```

public String getinfo(String path)
{
    File file;
    String str="";
    FileInputStream in;
    try{
        //打开文件 file 的 InputStream
        file = new File(path);
        in = new FileInputStream(file);
        //将文件内容全部读入到 byte 数组
        int length = (int)file.length();
        byte[] temp = new byte[length];
        in.read(temp, 0, length);
        //将 byte 数组用 UTF-8 编码并存入 display 字符串中
        str = EncodingUtils.getString(temp,TEXT_ENCODING);
        //关闭文件 file 的 InputStream

        in.close();
    }
    catch (IOException e) {

        DisplayToast(e.toString());

    }
    return str;
}

```

9.32、扫描wifi热点演示实例教程

1、首先新建了布局模板 XML 文件 `vifi.xml`,代码很简单,如下:

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <TextView android:id="@+id/wifi"
        android:layout_width="fill_parent"

```



```

        android:layout_height="wrap_content"
        android:text="@string/hello" />

```

```
</LinearLayout>
```

2、写 java 代码，新建个 Activity，代码如下：

```

package com.eoeandroid.demo.testcode;

import java.util.List;

import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.net.wifi.ScanResult;
import android.net.wifi.WifiManager;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;

public class WifiTester extends Activity {
    TextView mainText;
    WifiManager mainWifi;
    WifiReceiver receiverWifi;
    List<ScanResult> wifiList;
    StringBuilder sb = new StringBuilder();

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.vifi);
        setTitle("eoe 教程: Wifi Test. -by:IceskYsl");
        mainText = (TextView) findViewById(R.id.wifi);
        mainWifi = (WifiManager) getSystemService(Context.WIFI_SERVICE);

        receiverWifi = new WifiReceiver();
        registerReceiver(receiverWifi, new IntentFilter(
            WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));
        mainWifi.startScan();
        mainText.setText("\nStarting Scan...\n");
    }

    public boolean onCreateOptionsMenu(Menu menu) {
        menu.add(0, 0, 0, "Refresh");
    }
}

```



```

        return super.onCreateOptionsMenu(menu);
    }

    public boolean onOptionsItemSelected(int featureId, MenuItem item) {
        mainWifi.startScan();
        mainText.setText("Starting Scan");
        return super.onOptionsItemSelected(featureId, item);
    }

    protected void onPause() {
        unregisterReceiver(receiverWifi);
        super.onPause();
    }

    protected void onResume() {
        registerReceiver(receiverWifi, new IntentFilter(
            WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));
        super.onResume();
    }

    class WifiReceiver extends BroadcastReceiver {

        public void onReceive(Context c, Intent intent) {
            sb = new StringBuilder();
            wifiList = mainWifi.getScanResults();
            for (int i = 0; i < wifiList.size(); i++) {
                sb.append(new Integer(i + 1).toString() + ".");
                sb.append((wifiList.get(i)).toString());
                sb.append("\n\n");
            }
            mainText.setText(sb);
        }
    }
}

```

3、申请相关权限，代码如下：

```

<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"></uses-permission>
<uses-permission android:name="android.permission.ACCESS_CHECKIN_PROPERTIES"></uses-permission>
<uses-permission android:name="android.permission.WAKE_LOCK"></uses-permission>
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"></uses-permission>
<uses-permission android:name="android.permission.MODIFY_PHONE_STATE"></uses-permission>

```



9.33、调用google搜索

```
Intent intent = new Intent();
intent.setAction(Intent.ACTION_WEB_SEARCH);
intent.putExtra(SearchManager.QUERY,"searchString")
startActivity(intent);
```

9.34、调用浏览器 载入某网址

```
Uri uri = Uri.parse("http://www.google.hk.cn");
Intent it = new Intent(Intent.ACTION_VIEW,uri);
startActivity(it);
```

```
Uri uri = Uri.parse("http://www.google.com");
Intent it = new Intent(Intent.ACTION_VIEW,uri);
startActivity(it);
```

9.35、获取 IP地址

Android 获取 Ip 的一些方法,在我们开发中,有判断手机是否联网,或者想获得当前手机的 Ip 地址,当然 WIFI 连接的和

我们 3G 卡的 Ip 地址当然是不一样的.

首先我尝试了如下方法:

1. `WifiManager wifiManager = (WifiManager) getSystemService(WIFI_SERVICE);`
2. `WifiInfo wifiInfo = wifiManager.getConnectionInfo();`
3. `int ipAddress = wifiInfo.getIpAddress();`

但是获得的居然是一个整数,我尝试了用些数学方法都没有成功!,所以这种方法不可取!

最后查了一些资料,发现如下方法是比较通用的,我尝试了 WIFI 和 G3 卡,都获取了争取的 Ip 地址:代码如下:

1. `public String getLocalIpAddress() {`
2. `try {`



```

3.      for (Enumeration<NetworkInterface> en = NetworkInterface.getNetworkInterfaces();
en.hasMoreElements();) {
4.          NetworkInterface intf = en.nextElement();
5.          for (Enumeration<InetAddress> enumIpAddr = intf.getInetAddresses();
enumIpAddr.hasMoreElements();) {
6.              InetAddress inetAddress = enumIpAddr.nextElement();
7.              if (!inetAddress.isLoopbackAddress()) {
8.                  return inetAddress.getHostAddress().toString();
9.              }
10.         }
11.     }
12. } catch (SocketException ex) {
13.     Log.e(LOG_TAG, ex.toString());
14. }
15. return null;
16. }

```

不联网的情况下获得的是空值

方法 2:

```

WifiManager wifiManager = (WifiManager) getSystemService(WIFI_SERVICE);
WifiInfo wifiInfo = wifiManager.getConnectionInfo();
int ipAddress = wifiInfo.getIpAddress();
String ip = intToIp(ipAddress);

```

```

public String intToIp(int i) {

return ((i >> 24 ) & 0xFF ) + "." +
((i >> 16 ) & 0xFF) + "." +
((i >> 8 ) & 0xFF) + "." +
( i & 0xFF) ;
}

```

9.36、从输入流中获取数据并以字节数组返回

```

import java.io.ByteArrayOutputStream;
import java.io.InputStream;
public class StreamTool {
/**
* 从输入流获取数据
* @param inputStream

```



```

* @return
* @throws Exception
*/
public static byte[] readInputStream(InputStream inputStream) throws Exception {
    byte[] buffer = new byte[1024]; // 你可以根据实际需要调整缓存大小
    int len = -1;
    ByteArrayOutputStream outSteam = new ByteArrayOutputStream();
    while( (len = inputStream.read(buffer)) != -1 ){
        outSteam.write(buffer, 0, len);
    }
    outSteam.close();
    inputStream.close();
    return outSteam.toByteArray();
}
}

```

9.37、通过Android 客户端上传数据到服务器

```

import java.io.DataOutputStream;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
public class HttpRequester {
    /**
     * 直接通过 HTTP 协议提交数据到服务器,实现如下面表单提交功能:
     * <FORM METHOD=POST
     * ACTION="http://192.168.0.200:8080/ssi/fileload/test.do" enctype="multipart/form-data">
     * <INPUT TYPE="text" NAME="name">
     * <INPUT TYPE="text" NAME="id">
     * <input type="file" name="imagefile"/>
     * <input type="file" name="zip"/>
     * </FORM>

```



* @param actionUrl 上传路径(注: 避免使用 localhost 或 127.0.0.1 这样的路径测试, 因为它会指向手机模拟器, 你可以使用 <http://www.itcast.cn> 或 <http://192.168.1.10:8080> 这样的路径测试)

* @param params 请求参数 key 为参数名,value 为参数值

* @param file 上传文件

*/

```
public static String post(String actionUrl, Map<String, String> params, FormFile[]
files) {
    try {
        String BOUNDARY = "-----7d4a6d158c9"; //数据分隔线
        String MULTIPART_FORM_DATA = "multipart/form-data";
        URL url = new URL(actionUrl);
        HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
        conn.setConnectTimeout(5* 1000);
        conn.setDoInput(true); //允许输入
        conn.setDoOutput(true); //允许输出
        conn.setUseCaches(false); //不使用 Cache
        conn.setRequestMethod("POST");
        conn.setRequestProperty("Connection", "Keep-Alive");
        conn.setRequestProperty("Charset", "UTF-8");
        conn.setRequestProperty("Content-Type", MULTIPART_FORM_DATA
+ "; boundary=" + BOUNDARY);
        StringBuilder sb = new StringBuilder();
        for (Map.Entry<String, String> entry : params.entrySet()) { //构建表单
            字段内容
            sb.append("--");
            sb.append(BOUNDARY);
            sb.append("\r\n");
            sb.append("Content-Disposition: form-data; name=\"" +
entry.getKey() + "\"\r\n\r\n");
            sb.append(entry.getValue());
            sb.append("\r\n");
        }
        DataOutputStream outputStream = new
DataOutputStream(conn.getOutputStream());
        outputStream.write(sb.toString().getBytes()); //发送表单字段数据
        for (FormFile file : files) { //发送文件数据
            StringBuilder split = new StringBuilder();
            split.append("--");
            split.append(BOUNDARY);
            split.append("\r\n");
            split.append("Content-Disposition: form-data; name=\"" +
file.getFormname() + "\"; filename=\"" + file.getFilename() + "\"\r\n\r\n");
```



```

split.append("Content-Type: "+ file.getContentType()+"\r\n\r\n");
outStream.write(split.toString().getBytes());
if(file.getInStream()!=null){
byte[] buffer = new byte[1024];
int len = 0;
while((len = file.getInStream().read(buffer))!=-1){
outStream.write(buffer, 0, len);
}
file.getInStream().close();
}else{
outStream.write(file.getData(), 0, file.getData().length);
}
outStream.write("\r\n".getBytes());
}
byte[] end_data = ("--" + BOUNDARY + "--\r\n").getBytes();//数据结
束标志
outStream.write(end_data);
outStream.flush();
int cah = conn.getResponseCode();
if (cah != 200) throw new RuntimeException("请求 url 失败");
InputStream is = conn.getInputStream();
int ch;
StringBuilder b = new StringBuilder();
while( (ch = is.read()) != -1 ){
b.append((char)ch);
}
outStream.close();
conn.disconnect();
return b.toString();
} catch (Exception e) {
throw new RuntimeException(e);
}
}
}
/**
 * 提交数据到服务器
 * @param actionUrl 上传路径(注: 避免使用 localhost 或 127.0.0.1 这样的路径
测试, 因为它会指向手机模拟器, 你可以使用 http://www.itcast.cn 或
http://192.168.1.10:8080 这样的路径测试)
 * @param params 请求参数 key 为参数名,value 为参数值
 * @param file 上传文件
 */
public static String post(String actionUrl, Map<String, String> params, FormFile
file) {
return post(actionUrl, params, new FormFile[]{file});
}

```



```

}
public static byte[] postFromHttpClient(String path, Map<String, String> params,
String encode) throws Exception{
List<NameValuePair> formparams = new ArrayList<NameValuePair>();// 用
于存放请求参数
for(Map.Entry<String, String> entry : params.entrySet()){
formparams.add(new BasicNameValuePair(entry.getKey(),
entry.getValue()));
}
UrlEncodedFormEntity entity = new UrlEncodedFormEntity(formparams,
"UTF-8");
HttpPost httppost = new HttpPost(path);
httppost.setEntity(entity);
HttpClient httpclient = new DefaultHttpClient();//看作是浏览器
HttpResponse response = httpclient.execute(httppost);//发送 post 请求
return StreamTool.readInputStream(response.getEntity().getContent());
}
/**
 * 发送请求
 * @param path 请求路径
 * @param params 请求参数 key 为参数名称 value 为参数值
 * @param encode 请求参数的编码
 */
public static byte[] post(String path, Map<String, String> params, String encode)
throws Exception{
//String params = "method=save&name="+ URLEncoder.encode(" 老毕",
"UTF-8")+ "&age=28&";//需要发送的参数
StringBuilder parambuilder = new StringBuilder("");
if(params!=null && !params.isEmpty()){
for(Map.Entry<String, String> entry : params.entrySet()){
parambuilder.append(entry.getKey()).append("=")
.append(URLEncoder.encode(entry.getValue(),
encode)).append("&");
}
parambuilder.deleteCharAt(parambuilder.length()-1);
}
byte[] data = parambuilder.toString().getBytes();
URL url = new URL(path);
URLConnection conn = (URLConnection)url.openConnection();
conn.setDoOutput(true);//允许对外发送请求参数
conn.setUseCaches(false);//不进行缓存
conn.setConnectTimeout(5 * 1000);
conn.setRequestMethod("POST");
//下面设置 http 请求头

```



```

conn.setRequestProperty("Accept", "image/gif, image/jpeg, image/pjpeg,
image/pjpeg, application/x-shockwave-flash, application/xhtml+xml,
application/vnd.ms-xpsdocument, application/x-ms-xbap, application/x-ms-application,
application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*");
conn.setRequestProperty("Accept-Language", "zh-CN");
conn.setRequestProperty("User-Agent", "Mozilla/4.0 (compatible; MSIE 8.0;
Windows NT 5.2; Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR
3.0.04506.30; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)");
conn.setRequestProperty("Content-Type",
"application/x-www-form-urlencoded");
conn.setRequestProperty("Content-Length", String.valueOf(data.length));
conn.setRequestProperty("Connection", "Keep-Alive");
//发送参数
DataOutputStream outputStream = new
DataOutputStream(conn.getOutputStream());
outputStream.write(data);//把参数发送出去
outputStream.flush();
outputStream.close();
if(conn.getResponseCode()==200){
return StreamTool.readInputStream(conn.getInputStream());
}
return null;
}
}
}

```

9.38、文件下载类

支持文件的多线程断点续传，使用该类的即可安全、高效的下载任何类型的二进制文件：

1) FileDownloader

```

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.Properties;
import java.util.UUID;
import java.util.concurrent.ConcurrentHashMap;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import cn.itcast.service.FileService;

```



```
import android.content.Context;
import android.util.Log;
/**
 * 文件下载器
 */
public class FileDownloader {
    private Context context;
    private FileService fileService;
    private static final String TAG = "FileDownloader";
    /* 已下载文件大小*/
    private int downloadSize = 0;
    /* 原始文件大小*/
    private int fileSize = 0;
    /* 线程数*/
    private DownloadThread[] threads;
    /* 下载路径*/
    private URL url;
    /* 本地保存文件*/
    private File saveFile;
    /* 下载记录文件*/
    private File logFile;
    /* 缓存各线程最后下载的位置*/
    private Map<Integer, Integer> data = new ConcurrentHashMap<Integer, Integer>();
    /* 每条线程下载的大小*/
    private int block;
    private String downloadUrl;//下载路径
    /**
     * 获取线程数
     */
    public int getThreadSize() {
        return threads.length;
    }
    /**
     * 获取文件大小
     * @return
     */
    public int getFileSize() {
        return fileSize;
    }
    /**
     * 累计已下载大小
     * @param size
     */
    protected synchronized void append(int size) {
```




```

downloadSize += size;
}

/**
 * 更新指定线程最后下载的位置
 * @param threadId 线程id
 * @param pos 最后下载的位置
 */
protected void update(int threadId, int pos) {
    this.data.put(threadId, pos);
}

/**
 * 保存记录文件
 */
protected synchronized void saveLogFile() {
    this.fileService.update(this.downloadUrl, this.data);
}

/**
 * 构建文件下载器
 * @param downloadUrl 下载路径
 * @param fileSaveDir 文件保存目录
 * @param threadNum 下载线程数
 */
public FileDownloader(Context context, String downloadUrl, File fileSaveDir, int
threadNum) {
    try {
        this.context = context;
        this.downloadUrl = downloadUrl;
        fileService = new FileService(context);
        this.url = new URL(downloadUrl);
        if(!fileSaveDir.exists()) fileSaveDir.mkdirs();
        this.threads = new DownloadThread[threadNum];
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setConnectTimeout(6*1000);
        conn.setRequestMethod("GET");
        conn.setRequestProperty("Accept", "image/gif, image/jpeg, image/pjpeg,
image/pjpeg, application/x-shockwave-flash, application/xhtml+xml,
application/vnd.ms-xpsdocument, application/x-ms-xbap, application/x-ms-application,
application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*");
        conn.setRequestProperty("Accept-Language", "zh-CN");
        conn.setRequestProperty("Referer", downloadUrl);
        conn.setRequestProperty("Charset", "UTF-8");
        conn.setRequestProperty("User-Agent", "Mozilla/4.0 (compatible; MSIE 8.0;
Windows NT 5.2; Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR
3.0.04506.30; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)");
    }
}

```



```

conn.setRequestProperty("Connection", "Keep-Alive");
conn.connect();
printResponseHeader(conn);
if (conn.getResponseCode()==200) {
this.fileSize = conn.getContentLength();//根据响应获取文件大小
if (this.fileSize <= 0) throw new RuntimeException("1无法获知文件大小");
String filename = getFileName(conn);
this.saveFile = new File(fileSaveDir, filename);/* 保存文件*/
Map<Integer, Integer> logdata = fileService.getData(downloadUrl);
if(logdata.size()>0){
for(Map.Entry<Integer, Integer> entry : logdata.entrySet())
data.put(entry.getKey(), entry.getValue()+1);
}
this.block = this.fileSize / this.threads.length + 1;
if(this.data.size()==this.threads.length){
for (int i = 0; i < this.threads.length; i++) {
this.downloadSize += this.data.get(i+1)-(this.block * i);
}
print("已经下载的长度"+ this.downloadSize);
}
}else{
throw new RuntimeException("2服务器响应错误");
}
} catch (Exception e) {
print(e.toString());
throw new RuntimeException("3连接不到下载路径");
}
}
/**
 * 获取文件名
 */
private String getFileName(HttpURLConnection conn) {
String filename = this.url.toString().substring(this.url.toString().lastIndexOf('/') + 1);
if(filename==null || "".equals(filename.trim())){//如果获取不到文件名称
for (int i = 0;; i++) {
String mine = conn.getHeaderField(i);
if (mine == null) break;
if("content-disposition".equals(conn.getHeaderFieldKey(i).toLowerCase())){
Matcher m =
Pattern.compile(".*filename=(.*)").matcher(mine.toLowerCase());
if(m.find()) return m.group(1);
}
}
}
}

```



```

}
filename = UUID.randomUUID()+ ".tmp";//默认取一个文件名
}
return filename;
}
/**
 * 开始下载文件
 * @param listener 监听下载数量的变化, 如果不需要了解实时下载的数量, 可以设置为null
 * @return 已下载文件大小
 * @throws Exception
 */
public int download(DownloadProgressListener listener) throws Exception{
try {
if(this.data.size() != this.threads.length){
this.data.clear();
for (int i = 0; i < this.threads.length; i++) {
this.data.put(i+1, this.block * i);
}
}
for (int i = 0; i < this.threads.length; i++) {
int downLength = this.data.get(i+1) - (this.block * i);
if(downLength < this.block && this.data.get(i+1)<this.fileSize){ //该线程未完成下载时, 继续下载
RandomAccessFile randOut = new
RandomAccessFile(this.saveFile, "rw");
if(this.fileSize>0) randOut.setLength(this.fileSize);
randOut.seek(this.data.get(i+1));
this.threads[i] = new DownloadThread(this, this.url, randOut,
this.block, this.data.get(i+1), i+1);
this.threads[i].setPriority(7);
this.threads[i].start();
}else{
this.threads[i] = null;
}
}
this.fileService.save(this.downloadUrl, this.data);
boolean notFinish = true;//下载未完成
while (notFinish) {// 循环判断是否下载完毕
Thread.sleep(900);
notFinish = false;//假定下载完成
for (int i = 0; i < this.threads.length; i++){
if (this.threads[i] != null && !this.threads[i].isFinish()) {
notFinish = true;//下载没有完成

```



```

if(this.threads[i].getDownLength() == -1) { //如果下载失败, 再
重新下载
RandomAccessFile randOut = new
RandomAccessFile(this.saveFile, "rw");
randOut.seek(this.data.get(i+1));
this.threads[i] = new DownloadThread(this, this.url,
randOut, this.block, this.data.get(i+1), i+1);
this.threads[i].setPriority(7);
this.threads[i].start();
}
}
}

if(listener!=null) listener.onDownloadSize(this.downloadSize);
}

fileService.delete(this.downloadUrl);
} catch (Exception e) {
print(e.toString());
throw new Exception("下载失败");
}

return this.downloadSize;
}

/**
 * 获取Http 响应头字段
 * @param http
 * @return
 */
public static Map<String, String> getHttpResponseHeader(HttpURLConnection http) {
Map<String, String> header = new LinkedHashMap<String, String>();
for (int i = 0;; i++) {
String mine = http.getHeaderField(i);
if (mine == null) break;
header.put(http.getHeaderFieldKey(i), mine);
}
return header;
}

/**
 * 打印Http 头字段
 * @param http
 */
public static void printResponseHeader(HttpURLConnection http) {
Map<String, String> header = getHttpResponseHeader(http);
for(Map.Entry<String, String> entry : header.entrySet()) {
String key = entry.getKey()!=null ? entry.getKey()+ ":" : "";
print(key+ entry.getValue());
}
}

```



```

}
}
private static void print(String msg){
Log.i(TAG, msg);
}
public static void main(String[] args) {
/* FileDownloader loader = new FileDownloader(context,
"http://browse.babasport.com/ejb3/ActivePort.exe",
new File("D:\\androidsoft\\test"), 2);
loader.getFileSize();//得到文件总大小
try {
loader.download(new DownloadProgressListener() {
public void onDownloadSize(int size) {
print("已经下载: "+ size);
}
});
} catch (Exception e) {
e.printStackTrace();
}*/
}
}

```

2) 下面的类是真正支持下载的线程类:

```

import java.io.InputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URL;
import android.util.Log;
public class DownloadThread extends Thread {
private static final String TAG = "DownloadThread";
private RandomAccessFile saveFile;
private URL downUrl;
private int block;
/* 下载开始位置*/
private int threadId = -1;
private int startPos;
private int downLength;
private boolean finish = false;
private FileDownloader downloader;
public DownloadThread(FileDownloader downloader, URL downUrl,
RandomAccessFile saveFile, int block, int startPos, int threadId) {
this.downUrl = downUrl;
this.saveFile = saveFile;
this.block = block;

```



```

this.startPos = startPos;
this.downloader = downloader;
this.threadId = threadId;
this.downLength = startPos - (block * (threadId - 1));
}
@Override
public void run() {
    if(downLength < block) { //未下载完成
        try {
            HttpURLConnection http = (HttpURLConnection)
            downUrl.openConnection();
            http.setRequestMethod("GET");
            http.setRequestProperty("Accept", "image/gif, image/jpeg,
            image/pjpeg, image/pjpeg, application/x-shockwave-flash, application/xhtml+xml,
            application/vnd.ms-xpsdocument, application/x-ms-xbap, application/x-ms-application,
            application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*");
            http.setRequestProperty("Accept-Language", "zh-CN");
            http.setRequestProperty("Referer", downUrl.toString());
            http.setRequestProperty("Charset", "UTF-8");
            http.setRequestProperty("Range", "bytes=" + this.startPos + "-");
            http.setRequestProperty("User-Agent", "Mozilla/4.0 (compatible;
            MSIE 8.0; Windows NT 5.2; Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET
            CLR 3.0.04506.30; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)");
            http.setRequestProperty("Connection", "Keep-Alive");
            InputStream inStream = http.getInputStream();
            int max = 1024 * 1024;
            byte[] buffer = new byte[max];
            int offset = 0;
            print("线程" + this.threadId + "从位置" + this.startPos + "开始下
            载");
            while (downLength < block && (offset = inStream.read(buffer, 0,
            max)) != -1) {
                saveFile.write(buffer, 0, offset);
                downLength += offset;
                downloader.update(this.threadId, block * (threadId - 1) +
                downLength);
                downloader.saveLogFile();
                downloader.append(offset);
                int spare = block - downLength; //求剩下的字节数
                if(spare < max) max = (int) spare;
            }
            saveFile.close();
            inStream.close();
            print("线程" + this.threadId + "完成下载");
        }
    }
}

```



```

this.finish = true;
this.interrupt();
} catch (Exception e) {
this.downLength = -1;
print("线程"+ this.threadId+ "："+ e);
}
}
}

private static void print(String msg){
Log.i(TAG, msg);
}

/**
 * 下载是否完成
 * @return
 */
public boolean isFinish() {
return finish;
}

/**
 * 已经下载的内容大小
 * @return 如果返回值为-1, 代表下载失败
 */
public long getDownLength() {
return downLength;
}
}

```

3) 下面为监听器接口，会实时显示下载的大小，在实际使用的时候建议采用匿名类的方式构建此接口：

```

public interface DownloadProgressListener {
public void onDownloadSize(int size);
}

```

上面的三个文件在一起就构建起了一个迷你型的Android 下载框架，这个下载框架可以用于下载任何类型的二进制文件，以后需要下载的时候直接使用即可。其中IoC 非常直接的体现就是DownloadProgressListener ，在使用的时候只需要只需要传入该接口一个实现实例即可自动的获取实时的下载长度。

9.39、下载文件的进度条提示

```

try {
//set the download URL, a url that points to a file on the internet

```



```
//this is the file to be downloaded
URL url = new URL("http://somewhere.com/some/webhosted/file");

//create the new connection
URLConnection urlConnection = (URLConnection) url.openConnection();

//set up some things on the connection
urlConnection.setRequestMethod("GET");
urlConnection.setDoOutput(true);

//and connect!
urlConnection.connect();

//set the path where we want to save the file
//in this case, going to save it on the root directory of the
//sd card.
File SDCardRoot = Environment.getExternalStorageDirectory();
//create a new file, specifying the path, and the filename
//which we want to save the file as.
File file = new File(SDCardRoot,"somefile.ext");

//this will be used to write the downloaded data into the file we created
FileOutputStream fileOutput = new FileOutputStream(file);

//this will be used in reading the data from the internet
InputStream inputStream = urlConnection.getInputStream();

//this is the total size of the file
int totalSize = urlConnection.getContentLength();
//variable to store total downloaded bytes
int downloadedSize = 0;

//create a buffer...
byte[] buffer = new byte[1024];
int bufferLength = 0; //used to store a temporary size of the buffer

//now, read through the input buffer and write the contents to the file
while ( (bufferLength = inputStream.read(buffer)) > 0 ) {
//add the data in the buffer to the file in the file output stream (the file on the sd card
fileOutput.write(buffer, 0, bufferLength);
//add up the size so we know how much is downloaded
downloadedSize += bufferLength;
//this is where you would do something to report the progress, like this maybe
updateProgress(downloadedSize, totalSize);
```




```
}  
//close the output stream when done  
fileOutput.close();  
  
//catch some possible errors...  
} catch (MalformedURLException e) {  
    e.printStackTrace();  
} catch (IOException e) {  
    e.printStackTrace();  
}  
}
```

9.40、通过HttpClient从指定server获取数据

```
DefaultHttpClient httpClient = new DefaultHttpClient();  
HttpGet method = new HttpGet("http://www.baidu.com/1.html");  
HttpResponse resp;  
Reader reader = null;  
try {  
    // AllClientPNames.TIMEOUT  
    HttpParams params = new BasicHttpParams();  
    params.setIntParameter(AllClientPNames.CONNECTION_TIMEOUT, 10000);  
    httpClient.setParams(params);  
    resp = httpClient.execute(method);  
    int status = resp.getStatusLine().getStatusCode();  
    if (status != HttpStatus.SC_OK) return false;  
    // HttpStatus.SC_OK;  
    return true;  
} catch (ClientProtocolException e) {  
    // TODO Auto-generated catch block  
    e.printStackTrace();  
} catch (IOException e) {  
    // TODO Auto-generated catch block  
    e.printStackTrace();  
} finally {  
    if (reader != null) try {  
        reader.close();  
    } catch (IOException e) {  
        // TODO Auto-generated catch block  
        e.printStackTrace();  
    }  
}
```



9.41、通过FTP传输文件，关闭UI获得返回码

```

Intent intent = new Intent();
intent.setAction(Intent.ACTION_PICK);
// FTP URL (Starts with ftp://, sftp:// or ftps:// followed by hostname and port).
Uri ftpUri = Uri.parse("ftp://yourftpserver.com");
intent.setDataAndType(ftpUri, "vnd.android.cursor.dir/lysesoft.andftp.uri");
// FTP credentials (optional)
intent.putExtra("ftp_username", "anonymous");
intent.putExtra("ftp_password", "something@somewhere.com");
//intent.putExtra("ftp_keyfile", "/sdcard/dsakey.txt");
//intent.putExtra("ftp_keypass", "optionalkeypassword");
// FTP settings (optional)
intent.putExtra("ftp_pasv", "true");
//intent.putExtra("ftp_resume", "true");
//intent.putExtra("ftp_encoding", "UTF8");
// Download
intent.putExtra("command_type", "download");
// Activity title
intent.putExtra("progress_title", "Downloading files ...");
// Close activity after transfer (optional)
// intent.putExtra("close_ui", "true");
// Remote files to download.
intent.putExtra("remote_file1", "/remotefolder/subfolder/file1.zip");
intent.putExtra("remote_file2", "/remotefolder/subfolder/file2.zip");
// Target local folder where files will be downloaded.
intent.putExtra("local_folder", "/sdcard/localfolder");
startActivityForResult(intent, 0);
...
...
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
    Log.i(TAG, "Result: "+resultCode+ " from request: "+requestCode);
    if (data != null)
    {
        Log.i(TAG, "Status: "+ data.getStringExtra("TRANSFERSTATUS"));
    }
}

```

9.42、激活JavaScript打开内部链接

```
public class WebViewTest extends Activity {
```



```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    /* Set the Content View */
    setContentView(R.layout.main);

    /* Get the WebView */
    WebView wv1 = (WebView) findViewById(R.id.wv1);

    /* Activate JavaScript */
    wv1.getSettings().setJavaScriptEnabled(true);

    /* Prevent WebView from Opening the Browser */
    wv1.setWebViewClient(new InsideWebViewClient());
}

/* Class that prevents opening the Browser */
private class InsideWebViewClient extends WebViewClient {
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
        view.loadUrl(url);
        return true;
    }
}
```

9.43、清空手机cookies

```
CookieSyncManager.createInstance(getApplicationContext());
CookieManager.getInstance().removeAllCookie();
```

9.44、检查SD卡是否存在并且可以写入

```
public static boolean isSdPresent() {
    return
    android.os.Environment.getExternalStorageState().equals(android.os.Environment.MEDIA_MOUNTED);
}
```

如：向 SDcard 中存储数据：

1、



在 AndroidManifest.xml 中加入访问 SDCard 的权限如下:

```
<!-- 在 SDCard 中创建与删除文件权限 -->
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<!-- 往 SDCard 写入数据权限 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

2、Java 代码:

```
if(Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)){
    File sdCardDir = Environment.getExternalStorageDirectory();//获取 SDCard 目录
    File saveFile = new File(sdCardDir, "android.txt");
    FileOutputStream outputStream = new FileOutputStream(saveFile);
    outputStream.write("安桌".getBytes());
    outputStream.close();
}
```

9.45、获取SD卡的路径和存储空间

```
/** 获取存储卡路径 */
File sdcardDir=Environment.getExternalStorageDirectory();
/** StatFs 看文件系统空间使用情况 */
StatFs statFs=new StatFs(sdcardDir.getPath());
/** Block 的 size*/
Long blockSize=statFs.getBlockSize();
/** 总 Block 数量 */
Long totalBlocks=statFs.getBlockCount();
/** 已使用的 Block 数量 */
Long availableBlocks=statFs.getAvailableBlocks();
```

9.46、将程序安装到SD卡

Android 2.2 系统的一大改进就是通过Move to SD Card功能让用户可以安装程序到SD卡，不用担心手机内存不足的问题。但是最近很多安装Nexus One更新的用户都反映Move to SD Card功能不能使用，有人甚至担心自己安装的Android 2.2 升级是阉割版。

其实问题既不是出在Android 2.2 系统也不是用户安装的Android 2.2 更新，而是出在开发者那里。开发者如果想让自己的程序可以安装在运行Android 2.2 系统收集的SD里，必须在程序Manifest文件里添上下面这两行内容：

```
xmlns:android="http://schemas.android.com/apk/res/android"
android:installLocation="auto"
```



... >

如果每个程序都需要进行上面的修改,可以想见接下来 Android Market 里的程序将有一波规模空前的升级热潮,不过我觉得还有一个可能性就是 Google 对 Android 2.2 发布一个小的更新,让所有程序自动添加这个功能,在系统层面解决,不要麻烦开发者和用户了。

9.47、创建一个SD映像

很多网友不知道如何创建一个 SD 映像,使用 Android 模拟器时在安装软件、拍照等操作都需要 SD 卡辅助,我们可以通过 `mksdcard` 命令来创建,首先在 SDK 的根目录中找到 Tools 文件夹,输入 `mksdcard` 可以看到详细的参数说明,其中方框号的为可选,尖括号的为必选, label 卷标可以忽略,最终创建的文件格式将为 FAT32, 详细参数

`mksdcard`: create a blank FAT32 image to be used with the Android emulator

usage: `mksdcard [-l label] <size> <file>`

if <size> is a simple integer, it specifies a size in bytes

if <size> is an integer followed by 'K', it specifies a size in KiB

if <size> is an integer followed by 'M', it specifies a size in MiB

我们可以 `mksdcard -l android123 128M e:\cwj.image` 这行来创建一个 128M 的 sd 映像,卷标为 android123 这个是可选的,而最后是文件生成的路径,如图:



```

C:\WINDOWS\system32\cmd.exe

E:\android-sdk-windows-1.6_r1\android-sdk-windows-1.6_r1\tools>mksdcard
mksdcard: create a blank FAT32 image to be used with the Android emulator
usage: mksdcard [-l label] <size> <file>

    if <size> is a simple integer, it specifies a size in bytes
    if <size> is an integer followed by 'K', it specifies a size in KiB
    if <size> is an integer followed by 'M', it specifies a size in MiB

E:\android-sdk-windows-1.6_r1\android-sdk-windows-1.6_r1\tools>mksdcard -l andro
id123 128M e:\cwj.image
android123.com.cn
E:\android-sdk-windows-1.6_r1\android-sdk-windows-1.6_r1\tools>
  
```

不过在 ADT 插件中选择的 SD 卡文件时,注意数字后面的单位有 M 和 K。

9.48、查看手机内存存储

```
import java.io.File;
```

```
import android.os.Environment;
```

作者:craining (曲阜师范大学) 个人主页: <http://craining.blog.163.com/> 邮箱: craining@163.com 269



```
import android.os.StatFs;

public class MemoryStatus {

    static final int ERROR = -1;

    static public boolean externalMemoryAvailable() {
        return android.os.Environment.getExternalStorageState().equals(android.os.Environment.MEDIA_MOUNTED);
    }

    static public long getAvailableInternalMemorySize() {
        File path = Environment.getDataDirectory();
        StatFs stat = new StatFs(path.getPath());
        long blockSize = stat.getBlockSize();
        long availableBlocks = stat.getAvailableBlocks();
        return availableBlocks * blockSize;
    }

    static public long getTotalInternalMemorySize() {
        File path = Environment.getDataDirectory();
        StatFs stat = new StatFs(path.getPath());
        long blockSize = stat.getBlockSize();
        long totalBlocks = stat.getBlockCount();
        return totalBlocks * blockSize;
    }

    static public long getAvailableExternalMemorySize() {
        if(externalMemoryAvailable()) {
            File path = Environment.getExternalStorageDirectory();
            StatFs stat = new StatFs(path.getPath());
            long blockSize = stat.getBlockSize();
            long availableBlocks = stat.getAvailableBlocks();
            return availableBlocks * blockSize;
        } else {
            return ERROR;
        }
    }

    static public long getTotalExternalMemorySize() {
        if(externalMemoryAvailable()) {
            File path = Environment.getExternalStorageDirectory();
            StatFs stat = new StatFs(path.getPath());
            long blockSize = stat.getBlockSize();
            long totalBlocks = stat.getBlockCount();
```



```
return totalBlocks * blockSize;
} else {
return ERROR;
}
}

static public String formatSize(long size) {
String suffix = null;

if (size >= 1024) {
suffix = "KiB";
size /= 1024;
if (size >= 1024) {
suffix = "MiB";
size /= 1024;
}
}

StringBuilder resultBuffer = new StringBuilder(Long.toString(size));

int commaOffset = resultBuffer.length() - 3;
while (commaOffset > 0) {
resultBuffer.insert(commaOffset, ',');
commaOffset -= 3;
}

if (suffix != null)
resultBuffer.append(suffix);
return resultBuffer.toString();
}
}
```

9.49、在模拟器上调试Google Maps

1.在 android SDK 中预装的 add-on 中提供了一个 Map 扩展库 com.google.android.maps,利用它就可以在 android 的应用程序中加上强大的地图功能了。它位于 F:\android-sdk-windows\add-ons\google_apis-7_r01\libs (也就是你的 SDK 的安装路径下面,这里只是一个例子,我把它安装在了 F 盘下面)。不过在使用 Android Maps API 功能之前,你还需要申请一个 Android Maps API Key。

2.申请一个 Android Maps API Key

对于这个,网上有很多解决的办法,我也曾试着用了一下,但是在取得 debug.keystore 的 MD5 值的时候,

作者:craining (曲阜师范大学) 个人主页:<http://craining.blog.163.com/> 邮箱:craining@163.com 271



有些麻烦，而且有时候会出现很多错误，不用也罢。以下是我解决的步骤：

1) 找到你的 debug.keystore 文件所在的路径。

证书的一般路径为：打开 eclipse, 选择 Windows——>Preference——>Android——>Build, 其中 Default debug keystore 的值便是 debug.keystore 的路径，当然别的途径也可以得到，这里就不提了，因为这就是捷径，有捷径干嘛不用？

2) 取得 debug.keystore 的 MD5 值

首先在 DOS 下进入 debug.keystore 文件所在的路径，上面的一步已经得到，然后执行命令：

keytool -list -keystore debug.keystore (这个命令和网上说的解决办法不太一样，但简单了很多，而且也不容易出错)

这时可能会提示你输入密码，这里输入默认的密码 “android”，即可取得 MD5 的值

3) 获取 Maps API Key

打开浏览器，输入网址：
<http://code.google.com/intl/zh-CN/android/add-ons/google-apis/maps-api-signup.html> (当然，网上很多，也是这样干的，但是他们的网址似乎都有一些问题，书上的有些也不对，这是自己通过实践，得出的这样的网址，绝对没有问题)。当然，前提是你必须有一个 google 账户，如果没有，可以临时注册一个。然后，打开 google 页面，输入 code，一步步的查找，也可以找到，最终就会得到我一开始写的那个网址。(我建议大家还是这么做比较好，吃现成的东西，永远不会学到东西，而且也不会记得很牢，所以，你还是按我说的一步步找就可以了)。在打开的页面上，输入你在步骤 2 里得到的 MD5 认证指纹，勾选同意协议，按下 “Generate API Key” 按钮，即可得到我们申请到的 API Key

4) 在 main.xml 配置文件里添加如下的代码：

```
<com.google.android.maps.MapView

    android:id="@+id/MapView01"

    android:layout_width="fill_parent"

    android:layout_height="fill_parent"

    android:apiKey="0VCtgYqXDXHfzETwYIVNs-4IHkt8phKbBmhv2Vg" (注意这里要添的就是我们刚刚申请得到的 Maps API Key，我这里添的就是我得到的 Key 值)

/>
```

5) 同时我们也要在 AndroidManifest.xml 添加如下几行代码：



在 `<application/>` 前面添加如下代码：

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

```
<uses-permission android:name="android.permission.INTERNET"/>
```

这都是一些用户的权限，具体的你可以去 androidAPI 里去查，它里面已经给我们说的很清楚了。

同时，在 `<application/>` 里面添加如下代码：

```
<uses-library android:name="com.google.android.maps"/>
```

（用户库，记住，我们使用的是谷歌公司的地图，所以要在这下面找，而这些在 android 的 SDK 里面是没有的）

以上，就是我在学习使用 Android Maps API 开发地图应用程序的时候的一些解决办法，借助了 SDK 文档，也借助了网上的一些资源，当然也有不少是通过看书学到的。

下面我说说我遇到的问题以及错误的原因：

1> 按照如上步骤，我在获取取得 debug.keystore 的 MD5 值时候，很容易，也很快就获取了，但在获取 Key 值的时候，由于我的大意，获取的 API Key 其实是不对的，导致程序在模拟器上运行的时候，会显示一些个灰格子，而不是我所希望看到的地图，这就是由于你的 Map API Key 值错误的原因导致的；

2> 对于 Android SDK2.1 或者以后的开发，我建议用：
<http://code.google.com/intl/zh-CN/android/add-ons/google-apis/maps-api-signup.html> 这个网址去获取 Maps API Key，否则会出现一些问题。我就是因为一开始，没有使用上面的网址，导致 Map API Key 的取值错误，从而效果不出来。

9.50、建立GPRS连接

//Dial the [GPRS](#) link.

```
private boolean openDataConnection() {
    // Set up data connection.
    DataConnection conn = DataConnection.getInstance();
    if (connectMode == 0) {
        ret = conn.openConnection(mContext, "cmwap", "cmwap", "cmwap");
    } else {
        ret = conn.openConnection(mContext, "cmnet", "", "");
    }
}
```



9.51、获取手机位置

```
private double[] getGPS() {
    LocationManager lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
    List<String> providers = lm.getProviders(true);

    /* Loop over the array backwards, and if you get an accurate location, then break out the loop*/
    Location l = null;

    for (int i=providers.size()-1; i>=0; i--) {
        l = lm.getLastKnownLocation(providers.get(i));
        if (l != null) break;
    }

    double[] gps = new double[2];
    if (l != null) {
        gps[0] = l.getLatitude();
        gps[1] = l.getLongitude();
    }
    return gps;
}
```

9.5* 获得经纬度，地名标注在地图上

```
/**
 * @author Tony Shen
 *
 */
public class Main extends MapActivity {
    // 地图显示控制相关变量定义
    private MapView map = null;
    private MapController mapCon;
    private Geocoder geo;
    private static final int ERROR_DIALOG = 1;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        geo = new Geocoder(this, Locale.CHINA);
    }
}
```

=====黑软基地 [手机](#) 资讯频道=====



```
// 获取MapView
map = (MapView) findViewById(R.id.map);
// 设置显示模式
map.setTraffic(true);
map.setSatellite(false);
map.setStreetView(true);
// 设置可以缩放
map.setBuiltInZoomControls(true);
List
addresses = null;
try {
    addresses = geo.getLocationName("江苏省苏州市寒山寺", 1);
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
if(addresses.size() == 0) {
    showDialog(ERROR_DIALOG);
    GeoPoint geoBeijing = new GeoPoint(
        (int) (39.906033 * 1000000),
        (int) (116.397700 * 1000000));
    mapCon = map.getController();
    mapCon.setCenter(geoBeijing);
    mapCon.setZoom(4);
} else {
    Address address = addresses.get(0);
    // 设置初始地图的中心位置
    GeoPoint geoPoint = new GeoPoint(
        (int) (address.getLatitude() * 1000000),
        (int) (address.getLongitude() * 1000000));
    mapCon = map.getController();
    mapCon.setCenter(geoPoint);
    mapCon.setZoom(16);
    List overlays = this.map.getOverlays();
    overlays.add(new PositionOverlay(geoPoint, this, R.drawable.ic_red_pin));
}
}

@Override
protected boolean isRouteDisplayed() {
    return false;
}

@Override
protected Dialog onCreateDialog(int id) {
```



```

return new AlertDialog.Builder(this).setTitle("查询出错哦")
.setMessage("路名/地名出错，请重新输入!").create();
}

class PositionOverlay extends Overlay {
private GeoPoint geoPoint;
private Context context;
private int drawable;
public PositionOverlay(GeoPoint geoPoint, Context context, int drawable)
=====黑软基地 手机 资讯频道=====

{
    super();
    this.geoPoint = geoPoint;
    this.context = context;
    this.drawable = drawable;
}

@Override
public void draw(Canvas canvas, MapView mapView, boolean shadow) {
    Projection projection = mapView.getProjection();
    Point point = new Point();
    projection.toPixels(geoPoint, point);
    Bitmap bitmap = BitmapFactory.decodeResource(context.getResources(),
    drawable);
    canvas.drawBitmap(bitmap, point.x-bitmap.getWidth()/2, point.y-bitmap.getHeight(), null);
    super.draw(canvas, mapView, shadow);
}
}
}

效果图如下：

```

9.52、获得两个GPS坐标之间的距离

```

private double gps2m(float lat_a, float lng_a, float lat_b, float lng_b) {
float pk = (float) (180/3.14169);

float a1 = lat_a / pk;
float a2 = lng_a / pk;
float b1 = lat_b / pk;
float b2 = lng_b / pk;

float t1
FloatMath.cos(a1)*FloatMath.cos(a2)*FloatMath.cos(b1)*FloatMath.cos(b2);
float t2
FloatMath.cos(a1)*FloatMath.sin(a2)*FloatMath.cos(b1)*FloatMath.sin(b2);

```

```
float t3 = FloatMath.sin(a1)*FloatMath.sin(b1);
double tt = Math.acos(t1 + t2 + t3);

return 6366000*tt;
}
```

9.53、通过经纬度显示地图

```
Uri uri = Uri.parse("geo:38.899533,-77.036476");
Intent it = new Intent(Intent.ACTION_VIEW,uri);
startActivity(it);
```

9.54、路径规划

```
Uri uri
Uri.parse("\http://maps.google.com/maps?f=d&saddr=startLat%20startLng&daddr=endLat%20endLng&hl=en");
Intent it = new Intent(Intent.ACTION_VIEW,URI);
startActivity(it);
```

9.55、将坐标传递到google Map并显示

你将使用 Google 地图来显示用户的当前位置。在 main.xml 文件中的唯一修改指出就是为 MpaView 增加一个布局。在目前版本的 Android SDK 中，MapView 被建立为一个类 View。可能在将来的版本中 MapView 会相当于这个布局。

```
<view class="com.google.android.maps.MapView"
android:id="@+id/myMap"
```

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>
```

完成后的 main.xml 文件应当像这样：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android=http://schemas.android.com/apk/res/android
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<Button
android:id="@+id/gpsButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
```



```

android:text="Where Am I"
/>
<LinearLayout xmlns:android=http://schemas.android.com/apk/res/android
android:layout_width="wrap_content"
android:layout_height="wrap_content"
>
<TextView
android:id="@+id/latLabel"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Latitude: "
/>
<TextView
android:id="@+id/latText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
/>
</LinearLayout>
<LinearLayout xmlns:android=http://schemas.android.com/apk/res/android
android:layout_width="wrap_content"
android:layout_height="wrap_content"
>
<TextView
android:id="@+id/lngLabel"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Longitude: "
/>
<TextView
android:id="@+id/lngText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
/>
</LinearLayout>
<view class="com.google.android.maps.MapView"
android:id="@+id/myMap"
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>
</LinearLayout>

```

因为在这个活动中嵌入 `MapView`，你需要改变类的定义。现在，主要类扩展了活动。但是要正确的使用 `Google MapView`，你必须扩展 `MapActivity`。因此，你需要输入 `MapActivity` 包装并且替换在头部的 `Activity` 包装。

输入下列包装：

```
import com.google.android.maps.MapActivity;
```



```
import com.google.android.maps.MapView;
```

```
import com.google.android.maps.Point;
```

```
import com.google.android.maps.MapController
```

Point 包装将被用于保留 point 的值，它就是展示地图坐标的，而 MapController 将你的 point 置于地图中央。这两个包装在使用 MapView 时非常的关键。

现在准备增加建立地图并传递坐标的代码。首先，设置一个 MapView，并且从 main.xml 文件中把它赋值到布局：

```
MapView myMap = (MapView) findViewById(R.id.myMap);
```

下一步，设置一个 Point 并且把从 GPS 检索的数值赋值给 latPoint 和 lngPoint：

```
Point myLocation = new Point(latPoint.intValue(),lngPoint.intValue());
```

现在，可以创建 MapController 了，它将被用于移动 Google 地图来定位你定义的 Point。从 MapView 使用 getController()方法在定制的地图中建立一个控制器：

```
MapController myMapController = myMap.getController();
```

唯一剩下的工作就是使用控制器来移动地图到你的位置（要让地图更容易辨认，把 zoom 设定为 9）：

```
myMapController.centerMapTo(myLocation, false);
```

```
myMapController.zoomTo(9);
```

你刚才所写的所有代码就是从活动中利用 Google 地图。完整的类应当像这样：

```
package android_programmers_guide.AndroidLBS;

import android.os.Bundle;
import android.location.LocationManager;
import android.view.View;
import android.widget.TextView;
import android.content.Context;
import android.widget.Button;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapView;
import com.google.android.maps.Point;
import com.google.android.maps.MapController;

public class AndroidLBS extends MapActivity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle icle) {
        super.onCreate(icle);
        setContentView(R.layout.main);
        final Button gpsButton = (Button) findViewById(R.id.gpsButton);
        gpsButton.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v){
                LoadProviders();
            }
        });
    }

    public void LoadProviders(){
        TextView latText = (TextView) findViewById(R.id.latText);
        TextView lngText = (TextView) findViewById(R.id.lngText);
```



```

LocationManager myManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
Double latPoint =
myManager.getCurrentLocation("gps").getLatitude()*1E6;
Double lngPoint =
myManager.getCurrentLocation("gps").getLongitude()*1E6;
latText.setText(latPoint.toString());
lngText.setText(lngPoint.toString());
MapView myMap = (MapView) findViewById(R.id.myMap);
Point myLocation = new Point(latPoint.intValue(),lngPoint.intValue());
MapController myMapController = myMap.getController();
myMapController.centerMapTo(myLocation, false);
myMapController.zoomTo(9);
}
}

```

在模拟器中运行活动。活动应当打开一个空白的地图。点击“Where Am I”按钮，应当会看到地图聚焦并且放大到旧金山。

9.56、获取本机电话号码

```

private String getMyPhoneNumber(){
    TelephonyManager mTelephonyMgr;
    mTelephonyMgr = (TelephonyManager)
    getSystemService(Context.TELEPHONY_SERVICE);
    return mTelephonyMgr.getLine1Number();
}

private String getMy10DigitPhoneNumber(){
    String s = getMyPhoneNumber();
    return s.substring(2);
}

```

9.57、获得手机联系人

注意在</application>后要加上

```
<uses-permission android:name="android.permission.READ_CONTACTS"></uses-permission>
```

貌似从android2.0开始，联系人的API做了很大的调整。People接口由ContactsContract.Contacts代替。

闲话不多说看代码。

[Java](#)代码




```

/**
 * 获得联系人
 */
public void getContact(){
    //获得所有的联系人
    Cursor cur =
getContentResolver().query(ContactsContract.Contacts.CONTENT_URI,
null, null, null, null);
    //循环遍历
    if (cur.moveToFirst()) {
        int idColumn =
cur.getColumnIndex(ContactsContract.Contacts._ID);
        int displayNameColumn =
cur.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME);
        do {
            //获得联系人的ID号
            String contactId = cur.getString(idColumn);
            //获得联系人姓名
            String displayName = cur.getString(displayNameColumn);
            //查看该联系人有多少个电话号码。如果没有这返回值为0
            int phoneCount =
cur.getInt(cur.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUM
BER));

            if(phoneCount>0){
                //获得联系人的电话号码
                Cursor phones =
getContentResolver().query(ContactsContract.CommonDataKinds.Phone.CON
TENT_URI,null,ContactsContract.CommonDataKinds.Phone.CONTACT_ID+ " = "
+ contactId, null, null);
                if(phones.moveToFirst()){
                    do{
                        //遍历所有的电话号码
                        String phoneNum=
phones.getString(phones.getColumnIndex(ContactsContract.CommonDataKin
ds.Phone.NUMBER));

                        System.out.println(phoneNumber);
                    }while(phones.moveToNext());
                }
            } while (cur.moveToNext());
        }
    }
}

```



在联系人的电话号码中有很多种，如果只想获得手机号码。代码如下：

```
Cursor phones = mContext.getContentResolver().query(
    ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
    null,
    ContactsContract.CommonDataKinds.Phone.CONTACT_ID
    + " = " + contactId + " and
"+ContactsContract.CommonDataKinds.Phone.TYPE+"=" +ContactsContract.CommonDataKi
nds.Phone.TYPE_MOBILE, null, null);
```

打开联系人列表

<1>

```
Intent i = new Intent();
i.setAction(Intent.ACTION_GET_CONTENT);
i.setType("vnd.android.cursor.item/phone");
startActivityForResult(i, REQUEST_TEXT);
```

<2>

```
Uri uri = Uri.parse("content://contacts/people");
Intent it = new Intent(Intent.ACTION_PICK, uri);
startActivityForResult(it, REQUEST_TEXT);
```

9.58、2.0 以上版本查询联系人详细信息

<http://www.cmd100.com/bbs/forum.php?mod=viewthread&tid=1204&extra=page%3D1>

首先是查询联系人基本信息，2.0 查询的联系人信息 URI 为 ContactsContract.Contacts.CONTENT_URI。

```
ContentResolver cr = getContentResolver();
Cursor cur = cr.query(ContactsContract.Contacts.CONTENT_URI,
    null, null, null, null);
if (cur.getCount() > 0) {
    while (cur.moveToNext()) {
        String id = cur.getString(
            cur.getColumnIndex(ContactsContract.Contacts._ID)); //联系人 ID
        String name = cur.getString(
            cur.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME)); //联系人名字
        if
(Integer.parseInt(cur.getString(cur.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER))) > 0) {
// 号 码 查 询 ( 相 关 API :
http://androidappdocs.appspot.com/reference/android/provider/ContactsContract.CommonDataKinds.Phone.html)
// 判断是否有联系人号码 ContactsContract.Contacts.HAS_PHONE_NUMBER=1 为至少有一个，0
为没有，接下去就可以决定是否查询号码啦。
```

```
Cursor pCur = cr.query(
    ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null,
    ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = ?",
```



```

        new String[]{id}, null);
        while (pCur.moveToNext()) {
            // 做些操作
            String phoneNumber=
phones.getString(phones.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));
        }
        pCur.close();
    }
}
}
}

```

//Email 查 询 (相 关 API :
<http://androidappdocs.appspot.com/reference/android/provider/ContactsContract.CommonDataKinds.Email.html>)

```

Cursor emailCur = cr.query(
    ContactsContract.CommonDataKinds.Email.CONTENT_URI,
    null,
    ContactsContract.CommonDataKinds.Email.CONTACT_ID + " = ?",
    new String[]{id}, null);
while (emailCur.moveToNext()) {
    // 邮箱
    String email =
emailCur.getString(emailCur.getColumnIndex(ContactsContract.CommonDataKinds.Email.DATA));
    // 邮箱类型 (HOME,WORK,OTHOR.CUSTOM,MOBILE)
    String emailType = emailCur.getString(
        emailCur.getColumnIndex(ContactsContract.CommonDataKinds.Email.TYPE));
    }
    emailCur.close();
}

```

//Note 查 询 (相 关 API :
<http://androidappdocs.appspot.com/reference/android/provider/ContactsContract.CommonDataKinds.Note.html>)
String noteWhere = ContactsContract.Data.CONTACT_ID + " = ? AND " + ContactsContract.Data.MIMETYPE +
" = ?";

```

String[] noteWhereParams = new String[]{id,
    ContactsContract.CommonDataKinds.Note.CONTENT_ITEM_TYPE};
Cursor noteCur = cr.query(ContactsContract.Data.CONTENT_URI, null, noteWhere,
noteWhereParams, null);
if (noteCur.moveToFirst()) {
    String note =
noteCur.getString(noteCur.getColumnIndex(ContactsContract.CommonDataKinds.Note.NOTE));
}
noteCur.close();
}

```

// 邮 政 地 址 查 询 (相 关 API :
<http://androidappdocs.appspot.com/reference/android/provider/ContactsContract.CommonDataKinds.StructuredPo>



```

    String addrWhere = ContactsContract.Data.CONTACT_ID + " = ? AND " +
    ContactsContract.Data.MIMETYPE + " = ?";
    String[] addrWhereParams = new String[]{id,
        ContactsContract.CommonDataKinds.StructuredPostal.CONTENT_ITEM_TYPE};
    Cursor addrCur = cr.query(ContactsContract.Data.CONTENT_URI,
        null, where, whereParameters, null);
    while(addrCur.moveToNext()) {
//邮政信箱
        String poBox = addrCur.getString(

addrCur.getColumnIndex(ContactsContract.CommonDataKinds.StructuredPostal.POBBOX));
//街道
        String street = addrCur.getString(

addrCur.getColumnIndex(ContactsContract.CommonDataKinds.StructuredPostal.STREET));
//城市
        String city = addrCur.getString(

addrCur.getColumnIndex(ContactsContract.CommonDataKinds.StructuredPostal.CITY));
//地区
        String state = addrCur.getString(

addrCur.getColumnIndex(ContactsContract.CommonDataKinds.StructuredPostal.REGION));
//邮政编码
        String postalCode = addrCur.getString(

addrCur.getColumnIndex(ContactsContract.CommonDataKinds.StructuredPostal.POSTCODE));
//国家
        String country = addrCur.getString(

addrCur.getColumnIndex(ContactsContract.CommonDataKinds.StructuredPostal.COUNTRY));
//类型
        String type = addrCur.getString(

addrCur.getColumnIndex(ContactsContract.CommonDataKinds.StructuredPostal.TYPE));
    }
    addrCur.close();

//IM 查 询 ( 相 关 API :
http://androidappdocs.appspot.com/reference/android/provider/ContactsContract.CommonDataKinds.Im.html)
    String imWhere = ContactsContract.Data.CONTACT_ID + " = ? AND " + ContactsContract.Data.MIMETYPE + "
    = ?";
    String[] imWhereParams = new String[]{id,
        ContactsContract.CommonDataKinds.Im.CONTENT_ITEM_TYPE};

```



```

Cursor imCur = cr.query(ContactsContract.Data.CONTENT_URI,
    null, imWhere, imWhereParams, null);
if (imCur.moveToFirst()) {
    String imName = imCur.getString(
        imCur.getColumnIndex(ContactsContract.CommonDataKinds.Im.DATA));
    String imType;
    imType = imCur.getString(
        imCur.getColumnIndex(ContactsContract.CommonDataKinds.Im.TYPE));
}
imCur.close();

// 组 织 查 询 ( 相 关 API :
http://androidappdocs.appspot.com/reference/android/provider/ContactsContract.CommonDataKinds.Organization
.html)
String orgWhere = ContactsContract.Data.CONTACT_ID + " = ? AND " + ContactsContract.Data.MIMETYPE +
    " = ?";

String[] orgWhereParams = new String[] {id,
    ContactsContract.CommonDataKinds.Organization.CONTENT_ITEM_TYPE};

Cursor orgCur = cr.query(ContactsContract.Data.CONTENT_URI,
    null, orgWhere, orgWhereParams, null);
if (orgCur.moveToFirst()) {
//组织名
    String orgName =
    orgCur.getString(orgCur.getColumnIndex(ContactsContract.CommonDataKinds.Organization.DATA));
//职位
    String title =
    orgCur.getString(orgCur.getColumnIndex(ContactsContract.CommonDataKinds.Organization.TITLE));
}
orgCur.close();

```

9.59、2.0 以上版本添加联系人

据我个人理解，以前的联系人信息都是放在不同的表中，现在都是放在一张表中，但是分了很多的数据类型。自己的理解，仅供参考。

```

//联系人名字
ContentValues values = new ContentValues();
values.put(People.NAME, name);
Uri uri = contentResolver.insert(People.CONTENT_URI, values);
values.clear();
values.put(ContactsContract.Data.RAW_CONTACT_ID, uri.getLastPathSegment());
contentResolver.insert(ContactsContract.Data.CONTENT_URI, values);
//号码
Uri phoneUri = null;

```



```

phoneUri = Uri.withAppendedPath(uri, People.Phones.CONTENT_DIRECTORY);
for(int index=0;index <PHONE.SIZE();INDEX++)
{
values.clear();
values.put(People.Phones.TYPE, phone.get(index).getType());
values.put(People.Phones.NUMBER, phone.get(index).getNumber());
contentResolver.insert(phoneUri, values);
} //邮箱地址
values.clear();
values.put(ContactsContract.Data.RAW_CONTACT_ID, uri.getLastPathSegment());
values.put(ContactsContract.CommonDataKinds.Email.DATA, email);
values.put(ContactsContract.CommonDataKinds.Email.TYPE,
ContactsContract.CommonDataKinds.Email.TYPE_HOME);
values.put(ContactsContract.Data.MIMETYPE, ContactsContract.CommonDataKinds.StructuredPostal.CONTENT_ITEM_TYPE);
contentResolver.insert(ContactsContract.Data.CONTENT_URI, values);

//邮政地址
values.clear();
values.put(ContactsContract.Data.RAW_CONTACT_ID, uri.getLastPathSegment());
values.put(ContactsContract.CommonDataKinds.StructuredPostal.STREET, “街道”);
values.put(ContactsContract.CommonDataKinds.StructuredPostal.CITY, “城市”);
values.put(ContactsContract.CommonDataKinds.StructuredPostal.REGION, “地区”);
values.put(ContactsContract.CommonDataKinds.StructuredPostal.POSTCODE, “邮政编码”);
values.put(ContactsContract.CommonDataKinds.StructuredPostal.TYPE, ContactsContract.CommonDataKinds.StructuredPostal.TYPE_HOME);
values.put(ContactsContract.Data.MIMETYPE, ContactsContract.CommonDataKinds.StructuredPostal.CONTENT_ITEM_TYPE);
contentResolver.insert(ContactsContract.Data.CONTENT_URI, values);
}

//公司组织
ContentValues organisationValues = new ContentValues();
Uri orgUri = ContactsContract.Data.CONTENT_URI;
organisationValues.put(ContactsContract.Data.RAW_CONTACT_ID, uri.getLastPathSegment());
values.put(ContactsContract.CommonDataKinds.Organization.DATA, org.companyName); //公司名
values.put(ContactsContract.CommonDataKinds.Organization.TITLE, org.positionName); //职位
organisationValues.put(ContactsContract.Data.MIMETYPE, ContactsContract.CommonDataKinds.Organization.CONTENT_ITEM_TYPE);
organisationValues.put(ContactsContract.Organizations.TYPE, ContactsContract.Organizations.TYPE_WORK);
contentResolver.insert(orgUri, organisationValues);
}

```



9.60、拨打电话

```
Uri uri = Uri.parse("tel:xxxxxx");
Intent it = new Intent(Intent.ACTION_CALL, uri);
//Intent call = new Intent(Intent.ACTION_DIAL, uri); //只是拨号, 并未呼叫
startActivity(it);
```

注册权限

```
<uses-permission android:name="android.permission.CALL_PHONE"/>
```

9.61、发送SMS、MMS

```
Intent it = new Intent(Intent.ACTION_VIEW); it.putExtra("sms_body", "The SMS text");
it.setType("vnd. Android-dir/mms-sms");
startActivity(it);
```

发送短信

```
Uri uri = Uri.parse("smsto:0800000123");
Intent it = new Intent(Intent.ACTION_SENDTO, uri);
it.putExtra("sms_body", "The SMS text");
startActivity(it);
```

发送彩信

```
Uri uri = Uri.parse("content://media/external/images/media/23");
Intent it = new Intent(Intent.ACTION_SEND);
it.putExtra("sms_body", "some text");
it.putExtra(Intent.EXTRA_STREAM, uri);
it.setType("image/png");
startActivity(it);
```

发送短信:

```
String body=" this is mms demo" ;
Intent mmsintent = new Intent(Intent.ACTION_SENDTO, Uri.fromParts(" smsto" ,
number, null));
mmsintent.putExtra(Messaging.KEY_ACTION_SENDTO_MESSAGE_BODY, body);
mmsintent.putExtra(Messaging.KEY_ACTION_SENDTO_COMPOSE_MODE, true);
mmsintent.putExtra(Messaging.KEY_ACTION_SENDTO_EXIT_ON_SENT, true);
startActivity(mmsintent);
```

发送彩信:

```
StringBuilder sb = new StringBuilder();
sb.append("file://");
```



```

sb.append(fd.getAbsolutePath());
Intent intent = new Intent(Intent.ACTION_SENDTO, Uri.fromParts("mmsto", number, null));
// Below extra datas are all optional.
intent.putExtra(Messaging.KEY_ACTION_SENDTO_MESSAGE_SUBJECT, subject);
intent.putExtra(Messaging.KEY_ACTION_SENDTO_MESSAGE_BODY, body);
intent.putExtra(Messaging.KEY_ACTION_SENDTO_CONTENT_URI, sb.toString());
intent.putExtra(Messaging.KEY_ACTION_SENDTO_COMPOSE_MODE, composeMode);
intent.putExtra(Messaging.KEY_ACTION_SENDTO_EXIT_ON_SENT, exitOnSent);
startActivity(intent);

```

9.62、监听电话被呼叫状态

开发应用程序的时候，我们希望能够监听电话的呼入，以便执行暂停音乐播放器等操作，当电话结束之后，再次恢复播放。在 Android 平台可以通过 `TelephonyManager` 和 `PhoneStateListener` 来完成此任务。

`TelephonyManager` 作为一个 Service 接口提供给用户查询电话相关的内容，比如 IMEI, `LineNumber1` 等。通过下面的代码即可获得 `TelephonyManager` 的实例。

```

TelephonyManager mTelephonyMgr = (TelephonyManager) this
.getSystemService(Context.TELEPHONY_SERVICE);

```

在 Android 平台中，`PhoneStateListener` 是个很有用的监听器，用来监听电话的状态，比如呼叫状态和连接服务等。其方法如下所示：

```

public void onCallForwardingIndicatorChanged(boolean cfi)
public void onCallStateChanged(int state, String incomingNumber)
public void onCellLocationChanged(CellLocation location)
public void onDataActivity(int direction)
public void onDataConnectionStateChanged(int state)
public void onMessageWaitingIndicatorChanged(boolean mwi)
public void onServiceStateChanged(ServiceState serviceState)
public void onSignalStrengthChanged(int asu)

```

这里我们只需要覆盖 `onCallStateChanged()` 方法即可监听呼叫状态。在 `TelephonyManager` 中定义了三种状态，分别是振铃（RINGING），摘机（OFFHOOK）和空闲（IDLE），我们通过 `state` 的值就知道现在的电话状态了。

获得了 `TelephonyManager` 接口之后，调用 `listen()` 方法即可监听电话状态。

```

mTelephonyMgr.listen(new TeleListener(),
    PhoneStateListener.LISTEN_CALL_STATE);

```

监听实例：

下面是个简单的测试例子，只是把呼叫状态追加到 `TextView` 之上。

```

package com.j2medev;
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.telephony.PhoneStateListener;

```




```

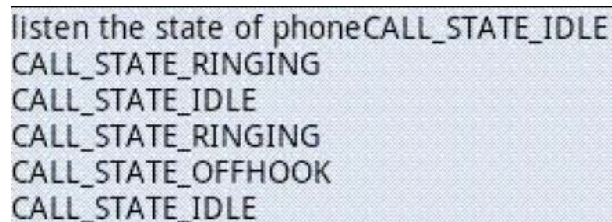
import android.telephony.TelephonyManager;
import android.util.Log;
import android.widget.TextView;
public class Telephony extends Activity {
    private static final String TAG = "Telephony";
    TextView view = null;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TelephonyManager mTelephonyMgr = (TelephonyManager) this
            .getSystemService(Context.TELEPHONY_SERVICE);
        mTelephonyMgr.listen(new TeleListener(),
            PhoneStateListener.LISTEN_CALL_STATE);
        view = new TextView(this);
        view.setText("listen the state of phone\n");
        setContentView(view);
    }
    class TeleListener extends PhoneStateListener {
        @Override
        public void onCallStateChanged(int state, String incomingNumber) {
            super.onCallStateChanged(state, incomingNumber);
            switch (state) {
                case TelephonyManager.CALL_STATE_IDLE: {
                    Log.e(TAG, "CALL_STATE_IDLE");
                    view.append("CALL_STATE_IDLE " + "\n");
                    break;
                }
                case TelephonyManager.CALL_STATE_OFFHOOK: {
                    Log.e(TAG, "CALL_STATE_OFFHOOK");
                    view.append("CALL_STATE_OFFHOOK " + "\n");
                    break;
                }
                case TelephonyManager.CALL_STATE_RINGING: {
                    Log.e(TAG, "CALL_STATE_RINGING");
                    view.append("CALL_STATE_RINGING " + "\n");
                    break;
                }
                default:
                    break;
            }
        }
    }
}

```

不要忘记在 AndroidManifest.xml 里面添加个 permission.



```
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```



```
listen the state of phoneCALL_STATE_IDLE
CALL_STATE_RINGING
CALL_STATE_IDLE
CALL_STATE_RINGING
CALL_STATE_OFFHOOK
CALL_STATE_IDLE
```

9.63、监听要拨打的电话(可以后台进行修改号码)

添加广播接收类，名字可任意，在 Manifest.xml 中添加 receiver:

```
<receiver android:name=".ReceiveCallOut">
    <intent-filter>
        <action android:name="android.intent.action.NEW_OUTGOING_CALL" />
    </intent-filter>
</receiver>
```

当然，还要写明监听权限:

```
<uses-permission
android:name="android.permission.PROCESS_OUTGOING_CALLS"></uses-permission>
```

然后在实现类中加上前缀 12593:

类的源代码: ReceiveCallOut.java

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class ReceiveCallOut extends BroadcastReceiver {

    @Override
    public void onReceive(Context arg0, Intent arg1) {

        this.setResultData( "12593" + this.getResultData() );
    }
}
```

这样，当系统监听到呼出电话时，会在呼出的号码前加上“12593”。

作者: craining (曲阜师范大学) 个人主页: <http://craining.blog.163.com/> 邮箱: craining@163.com 290



但是在测试后发现，手动拨数字呼出时，会加好 12593，通讯记录呼出时，也会拨出 12593，但是在进入联系人详细，选择呼叫联系人时，不会加 12593，难道这里是监听不到，还是这里本身不是 NEW_OUTGOING_CALL？

9.64、后台监听短信内容

AndroidManifest.xml 中添加

```
<receiver android:name=".receive">
    <intent-filter>
        <action android:name="android.provider.Telephony.SMS_RECEIVED" />
    </intent-filter>
</receiver>

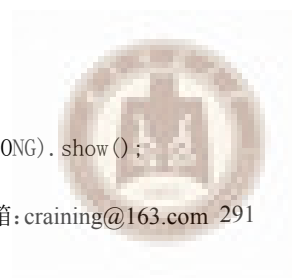
<uses-permission android:name="android.permission.RECEIVE_SMS"></uses-permission>
<uses-permission android:name="android.permission.READ_SMS"></uses-permission>
```

再写一个广播监听

```
public class receive extends BroadcastReceiver
{
    String receiveMsg = "";
    public void onReceive(Context context, Intent intent)
    {
        SmsMessage[] msg= null;

        if (intent.getAction().equals("android.provider.Telephony.SMS_RECEIVED"))
        {
            //StringBuilder buf = new StringBuilder();
            Bundle bundle = intent.getExtras();
            if (bundle != null) {
                Object[] pdusObj = (Object[]) bundle.get("pdus");
                msg= new SmsMessage[pdusObj.length];
                for (int i = 0; i<pdusObj.length; i++)
                    msg[i] = SmsMessage.createFromPdu ((byte[])
pdusObj[i]);
            }

            for(int i = 0; i < msg.length; i++)
            {
                String msgTxt = msg[i].getMessageBody();
                if (msgTxt.equals("Testing!"))
                {
                    Toast.makeText(context, "success!", Toast.LENGTH_LONG).show();
                }
            }
        }
    }
}
```



```

        return;
    }
    else
    {
        Toast.makeText(context, msgTxt, Toast.LENGTH_LONG).show();
        return;
    }
}
return;
}
}
}

```

9.65、删除最近收到的一条短信

1. 得到最近一条短信的 ID

```

private long getThreadId() {
    long threadId = 0;
    String SMS_READ_COLUMN = "read";
    String WHERE_CONDITION = SMS_READ_COLUMN + "= 0";
    String SORT_ORDER = "date DESC";
    int count = 0;
    Cursor cursor = mContext.getContentResolver().query(
        Uri.parse("content://sms/inbox"),
        new String[] { "_id", "thread_id", "address", "person", "date", "body" },
        WHERE_CONDITION,
        null,
        SORT_ORDER);
    if (cursor != null) {
        try {
            count = cursor.getCount();
            if (count > 0) {
                cursor.moveToFirst();
                threadId = cursor.getLong(1);
            }
        } finally {
            cursor.close();
        }
    }
    Log.i("threadId", String.valueOf(threadId));
    return threadId;
}

```

2. 删除操作:



```

long id = getThreadId();
Uri mUri=Uri.parse("content://sms/conversations/" + id);
mContext.getContentResolver().delete(mUri, null, null);

```

9.66、调用发短信的程序

注册权限

```
<uses-permission android:name="android.permission.SEND_SMS" />
```

方式一:

```

Intent it = new Intent(Intent.ACTION_VIEW);
it.putExtra("sms_body", "The SMS text");
it.setType("vnd.android-dir/mms-sms");
startActivity(it);

```

方式二:

```

Uri uri = Uri.parse("smsto:0800000123");
Intent it = new Intent(Intent.ACTION_SENDTO, uri);
it.putExtra("sms_body", "The SMS text");
startActivity(it);
String body="this is sms demo";
Intent mmsintent = new Intent(Intent.ACTION_SENDTO, Uri.fromParts("smsto", number, null));
mmsintent.putExtra(Messaging.KEY_ACTION_SENDTO_MESSAGE_BODY, body);
mmsintent.putExtra(Messaging.KEY_ACTION_SENDTO_COMPOSE_MODE, true);
mmsintent.putExtra(Messaging.KEY_ACTION_SENDTO_EXIT_ON_SENT, true);
startActivity(mmsintent);

```

9.67、后台发送短信

注册权限

```
<uses-permission android:name="android.permission.SEND_SMS" />
```

代码实现：

```

// 获取信息内容
String message ;
// 移动运营商允许每次发送的字节数据有限，我们可以使用 Android 给我们提供 的短信工具。
if (message != null) {
    SmsManager sms = SmsManager.getDefault();
    // 如果短信没有超过限制长度，则返回一个长度的 List。
    List<String> texts = sms.divideMessage(message);
    for (String text : texts) {
        sms.sendTextMessage("10086", "这里是发送者电话号码", "text", null, null);
    }
}

```



```
// (“这里是接收者电话号码”， “这里是发送者电话号码”， “这里是短信内容”， null, null);
}
}
//说明
sms.sendTextMessage(destinationAddress, scAddress, text, sentIntent, deliveryIntent):
destinationAddress:接收方的手机号码
scAddress:发送方的手机号码
text:信息内容
sentIntent:发送是否成功的回执，
DeliveryIntent:接收是否成功的回执，
```

9.68、调用发送彩信程序

```
Uri uri = Uri.parse("content://media/external/images/media/23");
Intent it = new Intent(Intent.ACTION_SEND);
it.putExtra("sms_body", "some text");
it.putExtra(Intent.EXTRA_STREAM, uri);
it.setType("image/png");
startActivity(it);
StringBuilder sb = new StringBuilder();
sb.append("file:///");
sb.append(fd.getAbsolutePath());
Intent intent = new Intent(Intent.ACTION_SENDTO, Uri.fromParts("mmsto", number, null));
// Below extra datas are all optional.
intent.putExtra(Messaging.KEY_ACTION_SENDTO_MESSAGE_SUBJECT, subject);
intent.putExtra(Messaging.KEY_ACTION_SENDTO_MESSAGE_BODY, body);
intent.putExtra(Messaging.KEY_ACTION_SENDTO_CONTENT_URI, sb.toString());
intent.putExtra(Messaging.KEY_ACTION_SENDTO_COMPOSE_MODE, composeMode);
intent.putExtra(Messaging.KEY_ACTION_SENDTO_EXIT_ON_SENT, exitOnSent);
startActivity(intent);
```

9.69、发送Email

```
Uri uri = Uri.parse("mailto:xxx@abc.com");
Intent it = new Intent(Intent.ACTION_SENDTO, uri);
startActivity(it);
```

或者:

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.putExtra(Intent.EXTRA_EMAIL, address);
intent.putExtra(Intent.EXTRA_SUBJECT, filename);
```



```
intent.putExtra(Intent.EXTRA_STREAM, Uri.parse("file://" + filename)); ;
intent.setType("text/csv");
startActivity(Intent.createChooser(intent, "Email File"));
```

或者:

```
Intent it = new Intent(Intent.ACTION_SEND);
it.putExtra(Intent.EXTRA_EMAIL, "me@abc.com");
it.putExtra(Intent.EXTRA_TEXT, "The email body text");
it.setType("text/plain");
startActivity(Intent.createChooser(it, "Choose Email Client"));
```

或者:

```
Intent it=new Intent(Intent.ACTION_SEND);
String[] tos={"me@abc.com"};
String[] ccs={"you@abc.com"};
it.putExtra(Intent.EXTRA_EMAIL, tos);
it.putExtra(Intent.EXTRA_CC, ccs);
it.putExtra(Intent.EXTRA_TEXT, "The email body text");
it.putExtra(Intent.EXTRA_SUBJECT, "The email subject text");
it.setType("message/rfc822");
startActivity(Intent.createChooser(it, "Choose Email Client"));
```

或者:

```
Intent it = new Intent(Intent.ACTION_SEND);
it.putExtra(Intent.EXTRA_SUBJECT, "The email subject text");
it.putExtra(Intent.EXTRA_STREAM, "file:///sdcard/mysong.mp3");
sendIntent.setType("audio/mp3");
startActivity(Intent.createChooser(it, "Choose Email Client"));
```

9.70、播放多媒体

```
Intent it = new Intent(Intent.ACTION_VIEW);
Uri uri = Uri.parse("file:///sdcard/song.mp3");
it.setDataAndType(uri, "audio/mp3");
startActivity(it);
```

或者

```
Uri uri = Uri.withAppendedPath(MediaStore.Audio.Media.INTERNAL_CONTENT_URI, "1");
Intent it = new Intent(Intent.ACTION_VIEW, uri);
startActivity(it);
```

或者

```
Uri playUri = Uri.parse("file:///sdcard/download/everything.mp3");
returnIt = new Intent(Intent.ACTION_VIEW, playUri);
```



9.71、控制音量

一个好的 Android 应用免不了会自带背景音乐，比如游戏或者一款比较不错的书本阅读器。一些好的应用在自带音乐的时候会多添加一款小功能即可以帮助用户设置声音大小或者改变应用的声音模式。

本篇基于 Android API 中的 AudioManager 作讲述，使看过本篇的读者可以迅速的掌握这个类的实现过程。下面是本篇大纲：

- 1、认识 AudioManager
- 2、AudioManager 主要方法介绍
- 3、程序逻辑实现过程

1、认识 AudioManager

AudioManager 类位于 android.Media 包中，该类提供访问控制音量和铃声模式的操作。

2、AudioManager 主要方法介绍

由于 AudioManager 该类方法过多，这里只讲述几个比较常用到的方法：

- 方法：adjustVolume(int direction, int flags)

解释：这个方法用来控制手机音量大小，当传入的第一个参数为 AudioManager.ADJUST_LOWER 时，可将音量调小一个单位，传入 AudioManager.ADJUST_RAISE 时，则可以将音量调大一个单位。

- 方法：getMode()

解释：返回当前音频模式。

- 方法：getRingerMode()

解释：返回当前的铃声模式。

- 方法：getStreamVolume(int streamType)

解释：取得当前手机的音量，最大值为 7，最小值为 0，当为 0 时，手机自动将模式调整为“震动模式”。

- 方法：setRingerMode(int ringerMode)

解释：改变铃声模式

3、程序逻辑实现过程

界面上设置了一个图片，表示当前铃声状态，一个进度条表示当前音量大小，五个图片按钮，用来表示增加/减小音量、普通模式、静音模式和震动模式。下面是界面的 XML 布局代码：

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/layout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/white"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
    android:id="@+id/myText1"
    android:layout_width="wrap_content"
```




```
        android:layout_height="wrap_content"
        android:text="@string/str_text1"
        android:textSize="16sp"
        android:textColor="@drawable/black"
        android:layout_x="20px"
        android:layout_y="42px"
    >
</TextView>
<ImageView
    android:id="@+id/myImage"
    android:layout_width="48px"
    android:layout_height="48px"
    android:layout_x="110px"
    android:layout_y="32px"
>
</ImageView>
<TextView
    android:id="@+id/myText2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/str_text2"
    android:textSize="16sp"
    android:textColor="@drawable/black"
    android:layout_x="20px"
    android:layout_y="102px"
>
</TextView>
<ProgressBar
    android:id="@+id/myProgress"
    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="160dip"
    android:layout_height="wrap_content"
    android:max="7"
    android:progress="5"
    android:layout_x="110px"
    android:layout_y="102px"
>
</ProgressBar>
<ImageButton
    android:id="@+id/downButton"
    android:layout_width="100px"
    android:layout_height="100px"
    android:layout_x="50px"
    android:layout_y="162px"
```



```
        android:src="@drawable/down"
    >
</ImageButton>
<ImageButton
    android:id="@+id/upButton"
    android:layout_width="100px"
    android:layout_height="100px"
    android:layout_x="150px"
    android:layout_y="162px"
    android:src="@drawable/up"
    >
</ImageButton>
<ImageButton
    android:id="@+id/normalButton"
    android:layout_width="60px"
    android:layout_height="60px"
    android:layout_x="50px"
    android:layout_y="272px"
    android:src="@drawable/normal"
    >
</ImageButton>
<ImageButton
    android:id="@+id/muteButton"
    android:layout_width="60px"
    android:layout_height="60px"
    android:layout_x="120px"
    android:layout_y="272px"
    android:src="@drawable/mute"
    >
</ImageButton>
<ImageButton
    android:id="@+id/vibrateButton"
    android:layout_width="60px"
    android:layout_height="60px"
    android:layout_x="190px"
    android:layout_y="272px"
    android:src="@drawable/vibrate"
    >
</ImageButton>
</AbsoluteLayout>
```

程序类分别为:

1、viewHolder



界面上的所有控件和元素都在这里静态声明

```
package com.terry;

import android.media.AudioManager;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.ProgressBar;

public
class viewHolder {

    public
    static ImageButton downButton;
    public
    static ImageButton upButton;
    public
    static ImageButton normalButton;
    public
    static ImageButton muteButton;
    public
    static ImageButton vibrateButton;
    public
    static ProgressBar myProgressBar;

    public
    static ImageView myImageView;

    public
    static AudioManager audiomanage;
}
```

2、AudioManagerActivity

程序入口处，和主要逻辑代码的处理，程序开头以 (AudioManager) getSystemService(AUDIO_SERVICE); 取得 AudioManager 对象。然后再利用该对象来对铃声进行调整。声明了一个返回 ImageButton 的方法，用来处理各自按钮点击所执行的不同事件和对图片的状态进行调整设置。以下把代码提供给大家：

```
package com.terry;

import android.app.Activity;
import android.media.AudioManager;
import android.os.Bundle;
import android.view.View;
```



```
import android.view.View.OnClickListener;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.ProgressBar;

public
class AudioManagerActivity extends Activity {
    //音量变量

private
int volume=0;
    //声音模式

private
int mode;
    /** Called when the activity is first created. */
    @Override
    public
void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    findview();
    //通过 getStreamVolume 获得当前音量大小
    volume=viewHolder.audiomanage.getStreamVolume(AudioManager.STREAM_RING);
    //把当前音量的值 设置给进度条
    viewHolder.myProgressBar.setProgress(volume);
    //得到当前的声音模式
    mode=viewHolder.audiomanage.getRingerMode();
    setImageState();
    viewHolder.downButton=btnListener(viewHolder.downButton);
    viewHolder.upButton=btnListener(viewHolder.upButton);
    viewHolder.muteButton=btnListener(viewHolder.muteButton);
    viewHolder.normalButton=btnListener(viewHolder.normalButton);
    viewHolder.vibrateButton=btnListener(viewHolder.vibrateButton);
}

    //找到控件

void findview(){
    viewHolder.downButton=(ImageButton)findViewById(R.id.downButton);
    viewHolder.upButton=(ImageButton)findViewById(R.id.upButton);
    viewHolder.muteButton=(ImageButton)findViewById(R.id.muteButton);
    viewHolder.normalButton=(ImageButton)findViewById(R.id.normalButton);
    viewHolder.vibrateButton=(ImageButton)findViewById(R.id.vibrateButton);
```



```

viewHolder.myImageView=(ImageView)findViewById(R.id.myImage);
viewHolder.myProgressBar=(ProgressBar)findViewById(R.id.myProgress);
viewHolder.audiomanage=(AudioManager)getSystemService(AUDIO_SERVICE);
}

//按钮 的单击事件
ImageButton btnListener(ImageButton b){
    b.setOnClickListener(new OnClickListener() {

        @Override
        public
        void onClick(View v) {
            // TODO Auto-generated method stub

            switch (v.getId()) {
                case R.id.downButton:
                    viewHolder.audiomanage.adjustVolume(AudioManager.ADJUST_LOWER, 0);
                    volume=viewHolder.audiomanage.getStreamVolume(AudioManager.STREAM_RING);
                    viewHolder.myProgressBar.setProgress(volume);
                    mode=viewHolder.audiomanage.getRingerMode();
                    setImageState();
                    break;
                case R.id.upButton:
                    viewHolder.audiomanage.adjustVolume(AudioManager.ADJUST_RAISE, 0);
                    volume=viewHolder.audiomanage.getStreamVolume(AudioManager.STREAM_RING);
                    viewHolder.myProgressBar.setProgress(volume);
                    mode=viewHolder.audiomanage.getRingerMode();
                    setImageState();
                    break;
                case R.id.muteButton:
                    viewHolder.audiomanage.setRingerMode(AudioManager.RINGER_MODE_SILENT);
                    volume=viewHolder.audiomanage.getStreamVolume(AudioManager.STREAM_RING);
                    viewHolder.myProgressBar.setProgress(volume);

viewHolder.myImageView.setImageDrawable(getResources().getDrawable(R.drawable.mute));
                    break;
                case R.id.normalButton:
                    viewHolder.audiomanage.setRingerMode(AudioManager.RINGER_MODE_NORMAL);
                    volume=viewHolder.audiomanage.getStreamVolume(AudioManager.STREAM_RING);
                    viewHolder.myProgressBar.setProgress(volume);

viewHolder.myImageView.setImageDrawable(getResources().getDrawable(R.drawable.normal));
                    break;
            }
        }
    });
}

```



```

        case R.id.vibrateButton:
            viewHolder.audiomanage.setRingerMode(AudioManager.RINGER_MODE_VIBRATE);
            volume=viewHolder.audiomanage.getStreamVolume(AudioManager.STREAM_RING);
            viewHolder.myProgressBar.setProgress(volume);

viewHolder.myImageView.setImageDrawable(getResources().getDrawable(R.drawable.vibrate));
            break;
        }

    }
});
return b;
}

//设置图片状态

void setImageState(){
    if(mode==AudioManager.RINGER_MODE_NORMAL)
    {
        viewHolder.myImageView.setImageDrawable(getResources().getDrawable(R.drawable.normal));
    }
    else
    if(mode==AudioManager.RINGER_MODE_SILENT)
    {
        viewHolder.myImageView.setImageDrawable(getResources().getDrawable(R.drawable.mute));
    }
    else
    if(mode==AudioManager.RINGER_MODE_VIBRATE)
    {
        viewHolder.myImageView.setImageDrawable(getResources().getDrawable(R.drawable.vibrate));
    }
}
}

```

9.72、定义ContentObserver，监听某个数据表

```

private ContentObserver mDownloadsObserver = new
DownloadsChangeObserver(Downloads.CONTENT_URI);
private class DownloadsChangeObserver extends ContentObserver {
    public DownloadsChangeObserver(Uri uri) {
        super(new Handler());
    }
    @Override

```



```
public void onChange(boolean selfChange) {}  
}
```

9.73、打开照相机

```
<1>Intent i = new Intent(Intent.ACTION_CAMERA_BUTTON, null);  
    this.sendBroadcast(i);  
  
<2>long dateTaken = System.currentTimeMillis();  
    String name = createName(dateTaken) + ".jpg";  
    fileName = folder + name;  
    ContentValues values = new ContentValues();  
    values.put(Image.Media.TITLE, fileName);  
    values.put("_data", fileName);  
    values.put(Image.Media.PICASA_ID, fileName);  
    values.put(Image.Media.DISPLAY_NAME, fileName);  
    values.put(Image.Media.DESCRPTION, fileName);  
    values.put(Image.Columns.BUCKET_DISPLAY_NAME, fileName);  
    Uri photoUri = getContentResolver().insert(  
        MediaStore.Images.Media.EXTERNAL_CONTENT_URI, values);  
  
    Intent inttPhoto = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);  
    inttPhoto.putExtra(MediaStore.EXTRA_OUTPUT, photoUri);  
    startActivityForResult(inttPhoto, 10);
```

9.74、从gallery选取图片

```
Intent i = new Intent();  
    i.setType("image/*");  
    i.setAction(Intent.ACTION_GET_CONTENT);  
    startActivityForResult(i, 11);
```

9.75、打开录音机

```
Intent mi = new Intent(Media.RECORD_SOUND_ACTION);  
    startActivity(mi);
```

9.76、语音朗读

```
package com.terry;
```



```
import java.util.Locale;
import android.app.Activity;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.speech.tts.TextToSpeech.OnInitListener;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class speechActivity extends Activity {
    private TextToSpeech mSpeech;
    private Button btn;
    private EditText mEditText;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        btn = (Button) findViewById(R.id.Button01);
        mEditText = (EditText) findViewById(R.id.EditText01);
        btn.setEnabled(false);
        mSpeech = new TextToSpeech(this, new OnInitListener() {

            @Override
            public void onInit(int status) {
                // TODO Auto-generated method stub
                if (status == TextToSpeech.SUCCESS) {
                    int result = mSpeech.setLanguage(Locale.ENGLISH);
                    if (result == TextToSpeech.LANG_MISSING_DATA
                        || result == TextToSpeech.LANG_NOT_SUPPORTED) {
                        Log.e("lanageTag", "not use");
                    } else {
                        btn.setEnabled(true);
                        mSpeech.speak("i love you", TextToSpeech.QUEUE_FLUSH,
                                    null);
                    }
                }
            }
        });
        btn.setOnClickListener(new OnClickListener() {
```




```
@Override
public void onClick(View v) {
    // TODO Auto-generated method stub
    mSpeech.speak(mEditText.getText().toString(),
        TextToSpeech.QUEUE_FLUSH, null);
}

});

}

@Override
protected void onDestroy() {
    // TODO Auto-generated method stub
    if (mSpeech != null) {
        mSpeech.stop();
        mSpeech.shutdown();
    }
    super.onDestroy();
}
}
```

9.77、手机获取视频流显示在电脑上

1. Android 端

```
package com.ruin.book.AndroidVideo;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.Socket;

import android.app.Activity;
import android.content.res.Configuration;
import android.graphics.PixelFormat;
import android.hardware.Camera;
import android.os.Bundle;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.view.SurfaceHolder.Callback;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
```



```
import com.ruin.book.AndroidVideo.R;

public class AndroidVideo extends Activity implements Callback, OnClickListener {
    private SurfaceView mSurfaceView = null;
    private SurfaceHolder mSurfaceHolder = null;
    private Camera mCamera = null;
    private boolean mPreviewRunning = false;

    //连接相关
    private EditText remoteIP = null;
    private Button connect = null;
    private String remoteIPStr = null;

    //视频数据
    private StreamIt streamIt = null;
    public static Kit kit = null;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        getWindow().setFormat(PixelFormat.TRANSLUCENT);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN);

        setContentView(R.layout.main);

        mSurfaceView = (SurfaceView) this.findViewById(R.id.surface_camera);
        mSurfaceHolder = mSurfaceView.getHolder();
        mSurfaceHolder.addCallback(this);
        mSurfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);

        remoteIP=(EditText)this.findViewById(R.id.EditText_remoteIP);
        connect=(Button)this.findViewById(R.id.Btn_connect);
        connect.setOnClickListener(this);
    }

    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
        if (mPreviewRunning) {
            mCamera.stopPreview();
        }
    }
}
```



```

        Camera.Parameters p = mCamera.getParameters();
        p.setPreviewSize(width, height);
        streamIt=new StreamIt();
        kit=new Kit();
        mCamera.setPreviewCallback(streamIt);

        mCamera.setParameters(p);
        try {
            mCamera.setPreviewDisplay(holder);
        } catch (Exception ex) {
        }
        mCamera.startPreview();
        mPreviewRunning = true;
    }

    public void surfaceCreated(SurfaceHolder holder) {
        mCamera = Camera.open();
    }

    public void surfaceDestroyed(SurfaceHolder holder) {
        mCamera.stopPreview();
        mPreviewRunning = false;
        mCamera.release();
    }

    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        try {
            super.onConfigurationChanged(newConfig);
            if (this.getResources().getConfiguration().orientation ==
Configuration.ORIENTATION_LANDSCAPE) {

            } else if (this.getResources().getConfiguration().orientation ==
Configuration.ORIENTATION_PORTRAIT) {

            }
        } catch (Exception ex) {
        }
    }

    class Kit implements Runnable {
        private boolean run = true;

```



```
//      private final int dataLen=57600; //307200 OR 230400 76800 OR 57600
private final int tt = 28800;

public void run() {
    // TODO Auto-generated method stub
    try {
        Socket socket = new Socket(remoteIPStr, 8899);
        DataOutputStream dos = new DataOutputStream(socket
            .getOutputStream());
        DataInputStream dis = new DataInputStream(socket
            .getInputStream());
        while (run) {
            dos.write(streamIt.yuv420sp, 0, tt);
            dos.write(streamIt.yuv420sp, tt, tt);

            dis.readBoolean();
            Thread.sleep(155);
        }
    } catch (Exception ex) {
        run=false;
        ex.printStackTrace();
    }
}

}

@Override
public void onClick(View view) {
    // TODO Auto-generated method stub
    if(view == connect){//连接函数
        remoteIPStr = remoteIP.getText().toString();
        new Thread(AndroidVideo.kit).start();
    }
}
}
```

```
class StreamIt implements Camera.PreviewCallback {
    public byte[] yuv420sp = null;
    private boolean t =true;

    public void onPreviewFrame(byte[] data, Camera camera) {
        // TODO Auto-generated method stub
        //      if(t){
        //          t=false;
        //      }
    }
}
```



```

        //          new Thread(AndroidVideo.kit).start();
        //      }
        yuv420sp=data;
    }
}

```

2. PC 端

```
package com.ruin.getAndroidVideo;
```

```

import java.awt.Frame;
import java.awt.Graphics;
import java.awt.Point;
import java.awt.Transparency;
import java.awt.color.ColorSpace;
import java.awt.image.BufferedImage;
import java.awt.image.ComponentColorModel;
import java.awt.image.DataBuffer;
import java.awt.image.DataBufferByte;
import java.awt.image.PixelInterleavedSampleModel;
import java.awt.image.Raster;
import java.awt.image.SampleModel;
import java.awt.image.WritableRaster;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.ServerSocket;
import java.net.Socket;

```

```

public class FlushMe extends Frame {
    private static final long serialVersionUID = 1L;
    private BufferedImage im;
    // 图像信息
    //     private final int width = 480;
    //     private final int height = 320;
    private static final int width = 240;
    private static final int height = 160;
    private static final int numBands = 3;
    private static final int dataLen = 57600;//307200 OR 230400//57600 76800
    private static final int tt = 28800;//14400//28800;
    // 图像数组
    private byte[] byteArray = new byte[width * height * numBands];// 图像 RGB 数组
    private byte[] yuv420sp = new byte[dataLen];// 图像 YUV 数组

    private static final int[] bandOffsets = new int[] { 0, 1, 2 };

```



```

private static final SampleModel sampleModel = new PixelInterleavedSampleModel(
    DataBuffer.TYPE_BYTE, width, height, 3, width * 3, bandOffsets);
// ColorModel
private static final ColorSpace cs=ColorSpace.getInstance(ColorSpace.CS_sRGB);
private static final ComponentColorModel cm=new ComponentColorModel(cs, false, false,
    Transparency.OPAQUE, DataBuffer.TYPE_BYTE);

public FlushMe() {
    super( "Android Camorra" );
    updateIM();
    setSize(480, 320);
    // 窗口关闭方法
    this.addWindowListener(new java.awt.event.WindowAdapter() {
        public void windowClosing(java.awt.event.WindowEvent e) {
            System.exit(0);
        }
    });
    // 窗口居中
    this.setLocationRelativeTo(null);
    this.setResizable(false);
    this.setVisible(true);
    this.getData();
}

public void update(Graphics g){
    paint(g);
}

public void paint(Graphics g) {
    g.drawImage(im, 0, 0, 480, 320, this);
}

public void getData() {
    try {
        ServerSocket server = new ServerSocket(8899);
        Socket socket = server.accept();
        DataInputStream dis = new DataInputStream(socket.getInputStream());
        DataOutputStream dos = new DataOutputStream(socket.getOutputStream());
        while (true) {
            for (int i = 0; i < dataLen / tt; i++) {
                dis.read(yuv420sp, i * tt, tt);
            }
            // 得到数据之后立即更新显示
            updateIM();
        }
    }
}

```



```

        im.flush();
        repaint();

        dos.writeBoolean(true);
    }
} catch (Exception ex) {
    ex.printStackTrace();
}
}

private void updateIM() {
    try {
        // 解析 YUV 成 RGB 格式
        decodeYUV420SP(byteArray, yuv420sp, width, height);
        DataBuffer dataBuffer = new DataBufferByte(byteArray, numBands);
        WritableRaster wr = Raster.createWritableRaster(sampleModel,
            dataBuffer, new Point(0, 0));
        im = new BufferedImage(cm, wr, false, null);
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}

private static void decodeYUV420SP(byte[] rgbBuf, byte[] yuv420sp, int width, int height) {
    final int frameSize = width * height;
    if (rgbBuf == null)
        throw new NullPointerException("buffer 'rgbBuf' is null");
    if (rgbBuf.length < frameSize * 3)
        throw new IllegalArgumentException("buffer 'rgbBuf' size "
            + rgbBuf.length + " < minimum " + frameSize * 3);

    if (yuv420sp == null)
        throw new NullPointerException("buffer 'yuv420sp' is null");

    if (yuv420sp.length < frameSize * 3 / 2)
        throw new IllegalArgumentException("buffer 'yuv420sp' size "
            + yuv420sp.length + " < minimum " + frameSize * 3 / 2);

    int i = 0, y = 0;
    int uvp = 0, u = 0, v = 0;
    int y1192 = 0, r = 0, g = 0, b = 0;

    for (int j = 0, yp = 0; j < height; j++) {
        uvp = frameSize + (j >> 1) * width;

```



```

        u = 0;
        v = 0;
        for (i = 0; i < width; i++, yp++) {
            y = (0xff & ((int) yuv420sp[yp])) - 16;
            if (y < 0)
                y = 0;
            if ((i & 1) == 0) {
                v = (0xff & yuv420sp[uvp++]) - 128;
                u = (0xff & yuv420sp[uvp++]) - 128;
            }

            y1192 = 1192 * y;
            r = (y1192 + 1634 * v);
            g = (y1192 - 833 * v - 400 * u);
            b = (y1192 + 2066 * u);

            if (r < 0)
                r = 0;
            else if (r > 262143)
                r = 262143;
            if (g < 0)
                g = 0;
            else if (g > 262143)
                g = 262143;
            if (b < 0)
                b = 0;
            else if (b > 262143)
                b = 262143;

            rgbBuf[yp * 3] = (byte) (r >> 10);
            rgbBuf[yp * 3 + 1] = (byte) (g >> 10);
            rgbBuf[yp * 3 + 2] = (byte) (b >> 10);
        }
    }
}

public static void main(String[] args) {
    Frame f = new FlushMe();
    f.setTitle("Android Camorra ");
}
}

```



9.78、蓝牙的使用

首先，要操作蓝牙，先要在 AndroidManifest.xml 里加入权限

```
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

```
<uses-permission android:name="android.permission.BLUETOOTH" />
```

一、常用类

然后，看下 api，Android 所有关于蓝牙开发的类都在 android.bluetooth 包下，如下图，只有 8 个类

Classes

```
BluetoothAdapter
BluetoothClass
BluetoothClass.Device
BluetoothClass.Device.Major
BluetoothClass.Service
BluetoothDevice
BluetoothServerSocket
BluetoothSocket
```

而我们需要用到了就只有几个而已：

1. BluetoothAdapter

顾名思义，蓝牙适配器，直到我们建立 bluetoothSocket 连接之前，都要不断操作它

BluetoothAdapter 里的方法很多，常用的有以下几个：

cancelDiscovery() 根据字面意思，是取消发现，也就是说当我们正在搜索设备的时候调用这个方法将不再继续搜索

disable() 关闭蓝牙

enable() 打开蓝牙，这个方法打开蓝牙不会弹出提示，更多的时候我们需要问下用户是否打开，一下这两行代码同样是打开蓝牙，不过会提示用户：

```
Intent enabler=new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
```

```
startActivityForResult(enabler,requestCode);//同 startActivity(enabler);
```

getAddress() 获取本地蓝牙地址

getDefaultAdapter() 获取默认 BluetoothAdapter，实际上，也只有这一种方法获取 BluetoothAdapter

getName() 获取本地蓝牙名称

getRemoteDevice(String address) 根据蓝牙地址获取远程蓝牙设备

getState() 获取本地蓝牙适配器当前状态（感觉可能调试的时候更需要）

isDiscovering() 判断当前是否正在查找设备，是返回 true

isEnabled() 判断蓝牙是否打开，已打开返回 true，否则，返回 false

listenUsingRfcommWithServiceRecord(String name, UUID uuid) 根据名称，UUID 创建并返回 BluetoothServerSocket，这是创建 BluetoothSocket 服务器端的第一步

startDiscovery() 开始搜索，这是搜索的第一步

2. BluetoothDevice



看名字就知道，这个类描述了一个蓝牙设备

`createRfcommSocketToServiceRecord(UUIDuuid)`根据 UUID 创建并返回一个 `BluetoothSocket`

这个方法也是我们获取 `BluetoothDevice` 的目的——创建 `BluetoothSocket`

这个类其他的方法，如 `getAddress()`,`getName()`,同 `BluetoothAdapter`

3. BluetoothServerSocket

如果去除了 `Bluetooth` 相信大家一定再熟悉不过了，既然是 `Socket`，方法就应该都差不多，这个类一种只有三个方法

两个重载的 `accept()`,`accept(int timeout)`两者的区别在于后面的方法指定了过时时间，需要注意的是，执行这两个方法的时候，直到接收到了客户端的请求（或是过期之后），都会阻塞线程，应该放在新线程里运行！

还有一点需要注意的是，这两个方法都返回一个 `BluetoothSocket`，最后的连接也是服务器端与客户端的两个 `BluetoothSocket` 的连接

`close()`这个就不用说了吧，翻译一下——关闭！

4. BluetoothSocket

跟 `BluetoothServerSocket` 相对，是客户端

一共 5 个方法，不出意外，都会用到

`close()`,关闭

`connect()`连接

`getInputStream()`获取输入流

`getOutputStream()`获取输出流

`getRemoteDevice()`获取远程设备，这里指的是获取 `bluetoothSocket` 指定连接的那个远程蓝牙设备。

二、用法:

1、获取本地蓝牙适配器

`BluetoothAdapter`

`mAdapter= BluetoothAdapter.getDefaultAdapter();`

2、打开蓝牙

`if(!mAdapter.isEnabled()){`

//弹出对话框提示用户是否后打开

`Intent enabler = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);`

`startActivityForResult(enabler, REQUEST_ENABLE);`

//不做提示，强行打开

// `mAdapter.enable();`

`}`

3、搜索设备

1)刚才说过了 `mAdapter.startDiscovery()`

是第一步,你会发现没有返回的蓝牙设备，怎么知道查找到了呢？往下看，不要急

2)定义 `BroadcastReceiver`,关于 `BroadcastReceiver` 不多讲了，不是今天的讨论内容，代码如下

```

BroadcastReceiver mReceiver = new BroadcastReceiver() {
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        //找到设备
        if (BluetoothDevice.ACTION_FOUND.equals(action)) {
            BluetoothDevice device = intent
                .getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            if (device.getBondState() != BluetoothDevice.BOND_BONDED) {
                Log.v(TAG, "find device:" + device.getName()
                    + device.getAddress());
            }
        }
        //搜索完成
        else if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED
            .equals(action)) {
            setTitle("搜索完成");
            if (mNewDevicesAdapter.getCount() == 0) {

                Log.v(TAG, "find over");
            }
        }
        //执行更新列表的代码

    }
};

```

这样，没当查找到新设备或是搜索完成，相应的操作都在上段代码的两个 if 里执行了，不过前提是你要先注册 `BroadcastReceiver`，具体代码如下

```

IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
registerReceiver(mReceiver, filter);
filter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
registerReceiver(mReceiver, filter);

```

（这段代码，一般写在 `onCreate()` 里..）

4、建立连接

首先 Android sdk（2.0 以上版本）支持的蓝牙连接是通过 `BluetoothSocket` 建立连接（说的不对请高人指正），服务器端（`BluetoothServerSocket`）和客户端（`BluetoothSocket`）需指定同样的 UUID，才能建立连接，因为建立连接的方法会阻塞线程，所以服务器端和客户端都应启动新线程连接

1) 服务器端：



```
//UUID 格式一般是"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"可到
//http://www.uuidgenerator.com 申请
BluetoothServerSocket serverSocket = mAdapter.
listenUsingRfcommWithServiceRecord(serverSocketName,UUID);
serverSocket.accept();
```

2)客户端:

//还记得我们刚才在 BroadcastReceiver 获取了 BluetoothDevice 么?

```
BluetoothSocket clienSocket=device. createRfcommSocketToServiceRecord(UUID);

clienSocket.connect();
```

5、数据传递

通过以上操作，就已经建立的 BluetoothSocket 连接了，数据传递无非是通过流的形式

1) 获取流

```
InputStream = socket.getInputStream();
OutputStream = socket.getOutputStream();
```

2) 写出、读入

略

android.bluetooth 下有 8 个类，还有 4 个类没有用到，那 4 个类里定义的都是常量

9.79、一个很好的加密解密字符串

```
package net.sf.andhsl.hotspotlogin;

import java.security.SecureRandom;

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;

/**
 * Usage:
 * <pre>
 * String crypto = SimpleCrypto.encrypt(masterpassword, cleartext)
 * ...
 * String cleartext = SimpleCrypto.decrypt(masterpassword, crypto)
```



```
* </pre>
* @author ferenc.hechler
*/
public class SimpleCrypto {

    public static String encrypt(String seed, String cleartext) throws Exception {
        byte[] rawKey = getRawKey(seed.getBytes());
        byte[] result = encrypt(rawKey, cleartext.getBytes());
        return toHex(result);
    }

    public static String decrypt(String seed, String encrypted) throws Exception {
        byte[] rawKey = getRawKey(seed.getBytes());
        byte[] enc = toByte(encrypted);
        byte[] result = decrypt(rawKey, enc);
        return new String(result);
    }

    private static byte[] getRawKey(byte[] seed) throws Exception {
        KeyGenerator kgen = KeyGenerator.getInstance("AES");
        SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
        sr.setSeed(seed);
        kgen.init(128, sr); // 192 and 256 bits may not be available
        SecretKey skey = kgen.generateKey();
        byte[] raw = skey.getEncoded();
        return raw;
    }

    private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
        SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
        Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
        byte[] encrypted = cipher.doFinal(clear);
        return encrypted;
    }

    private static byte[] decrypt(byte[] raw, byte[] encrypted) throws Exception {
        SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
        Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.DECRYPT_MODE, skeySpec);
        byte[] decrypted = cipher.doFinal(encrypted);
        return decrypted;
    }
}
```



```
public static String toHex(String txt) {
    return toHex(txt.getBytes());
}

public static String fromHex(String hex) {
    return new String(toByte(hex));
}

public static byte[] toByte(String hexString) {
    int len = hexString.length()/2;
    byte[] result = new byte[len];
    for (int i = 0; i < len; i++)
        result[i] = Integer.valueOf(hexString.substring(2*i, 2*i+2), 16).byteValue();
    return result;
}

public static String toHex(byte[] buf) {
    if (buf == null)
        return "";
    StringBuffer result = new StringBuffer(2*buf.length);
    for (int i = 0; i < buf.length; i++) {
        appendHex(result, buf[i]);
    }
    return result.toString();
}

private final static String HEX = "0123456789ABCDEF";
private static void appendHex(StringBuffer sb, byte b) {
    sb.append(HEX.charAt((b>>4)&0x0f)).append(HEX.charAt(b&0x0f));
}
}
```

9.80、Drawable、Bitmap、byte[]之间的转换

1、Drawable → Bitmap 的简单方法

```
((BitmapDrawable)res.getDrawable(R.drawable.youricon)).getBitmap();
```

2、Drawable → Bitmap 的复杂方法

Java 代码

```
public static Bitmap drawableToBitmap(Drawable drawable) {

    Bitmap bitmap = Bitmap
        .createBitmap(
```



```

        drawable.getIntrinsicWidth(),
        drawable.getIntrinsicHeight(),
        drawable.getOpacity() !=
PixelFormat.OPAQUE ? Bitmap.Config.ARGB_8888
                                : Bitmap.Config.RGB_565);

    Canvas canvas = new Canvas(bitmap);
    //canvas.setBitmap(bitmap);
    drawable.setBounds(0, 0, drawable.getIntrinsicWidth(), drawable.getIntrinsicHeight());
    drawable.draw(canvas);
    return bitmap;
}

```

3、Bitmap → Drawable 的简单方法

```

BitmapDrawable bitmapDrawable = (BitmapDrawable)bitmap;
Drawable drawable = (Drawable)bitmapDrawable;

```

```

Bitmap bitmap = new Bitmap (...);
Drawable drawable = new BitmapDrawable(bitmap);

```

4、从资源中获取 Bitmap

Java 代码

```
Resources res=getResources();
```

```
Bitmap bmp=BitmapFactory.decodeResource(res, R.drawable.pic);
```

5、Bitmap → byte[]

Java 代码

```

private byte[] Bitmap2Bytes(Bitmap bm){
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    bm.compress(Bitmap.CompressFormat.PNG, 100, baos);
    return baos.toByteArray();
}

```

6、byte[] → Bitmap

Java 代码

```

private Bitmap Bytes2Bimap(byte[] b){
    if(b.length!=0){
        return BitmapFactory.decodeByteArray(b, 0, b.length);
    }
    else {
        return null;
    }
}

```



9.81、高循环效率的代码

```
/*  
 * How To Write Faster Loops (after Dan Bornstein, Google Engineer)  
 *  
 * - http://www.youtube.com/watch?v=ptjedOZEXPM  
 *  
 */
```

```
/* 1 (fastest) */  
for (int i = initializer; i >= 0; i--) { ... }
```

```
/* 2 */  
int limit = calculateLoopLimit();  
for (int i = 0; i < limit; i++) { ... }
```

```
/* 3 */  
Type[] array = getMyArray();  
for (Type obj : array) { ... }
```

```
/* 4 */  
for (int i = 0; i < array.length; i++) { ... }
```

```
/* 5 */  
for (int i = 0; i < this.var; i++) { ... }
```

```
/* 6 */  
for (int i = 0; i < obj.size(); i++) { ... }
```

```
/* 7 (slowest) */  
Iterable<Type> list = getMyList();  
for (Type obj : list) { ... }
```

```
/*  
 * How To Write Faster Loops (after Dan Bornstein, Google Engineer)  
 *  
 * - http://www.youtube.com/watch?v=ptjedOZEXPM  
 *  
 */
```

```
/* 1 (fastest) */  
for (int i = initializer; i >= 0; i--) { ... }
```

```
/* 2 */
```




```
int limit = calculateLoopLimit();
for (int i = 0; i < limit; i++) { ... }

/* 3 */
Type[] array = getMyArray();
for (Type obj : array) { ... }

/* 4 */
for (int i = 0; i < array.length; i++) { ... }

/* 5 */
for (int i = 0; i < this.var; i++) { ... }

/* 6 */
for (int i = 0; i < obj.size(); i++) { ... }

/* 7 (slowest) */
Iterable<Type> list = getMyList();
for (Type obj : list) { ... }
```

9.82、给模拟器打电话发短信

通过 `gsm call` 命令可以像 Android 模拟器打电话，除了在 EclipseADT 的 DDMS 中通过按钮 Dial 外，还可以通过 DDMS 外壳调用 `gsm call` 命令直接拨打，我们首先需要启动 AndroidEmulator，然后在 cmd 环境下执行 `telnet localhost 5554` 下面就可以向 Android 模拟器 拨号，参数为 `gsmcall < phoneNum>`，比如给 10086 打电话 为 `gsm call +10086`

9.83、加快模拟器速度

其实使用虚拟化的产品在性能上还是有很大的折扣，很多网友发现每次随着 Android 固件的更新、SDK 功能的增加启动速度越来越慢，从早期的 m3 版 SDK 到如今的 1.5 几乎足足慢了 3 倍，其实这是很正常的事情，SDK 的体积也从 55MB 增至 160MB 左右，对于启动时加载的组件也越来越多。

目前给大家以下几个建议：

1. 如果在 Linux 下开发直接使用 VMWare 这样的虚拟机里面跑，每次关闭开发环境时可以直接保存快照，这样在 Linux 下还可以执行 Windows 下的一些程序，比较方便。



2. 使用 RAID 0 或 RAMDISK 的软件来提高系统的 I/O 性能。
3. 如果使用 Windows 操作系统作为开发环境，其实在第一次运行 Android 模拟器的时候系统已经在 C 盘当前登陆帐户下存有缓存。
4. 定期使用 `-wipe-data` 参数启动 `emulator` 来重置模拟器数据。
5. 可以使用 Android 开发网即将提供的一套辅助同步软件做加速，同时使用真机作为调试比较模拟器更有效，运行速度更高。

9.83.1、模拟器“尚未注册网络”

打开 Android 模拟器时，出现无信号，拨打电话或发短信时，提示“尚未注册网络”错误信息的解决方案如下。

- 场景一：你的电脑没有连接上互联网，同时也没有在局域网。

解决办法：右键点击网上邻居,选择"属性",在网络连接窗口中右键点击"本地连接",选择"属性",设置 TCP/IP 属性如下:

IP 地址:192.168.1.100
子网掩码:255.255.255.0
默认网关:192.168.1.100
首选 DNS 服务器:192.168.1.100

- 场景二：你的电脑没有连接上互联网，但在局域网。

解决办法：右键点击网上邻居,选择"属性",在网络连接窗口中右键点击"本地连接",选择"属性",设置 TCP/IP 属性如下:

IP 地址:设置成你所在局域网的 IP，如：192.168.1.100
子网掩码:设置成你所在局域网的掩码，如：255.255.255.0
默认网关:设置成你所在局域网的网关，一般网关的 IP 格式为：*.*.*.1，如：192.168.1.1
首选 DNS 服务器:设置成你所在局域网的路由器 IP，一般路由器的 IP 格式为：*.*.*.1，

如：192.168.1.1

最后一种解决方案是：让你的电脑连接上互联网。

9.84、emulator命令行参数

Android SDK 最重要的仿真器程序 `emulator`，除了前面介绍的 `-avd` 参数(指定 Android 硬件装置与目标平台)、`-sdcard` 参数(在仿真器启动时抓取 SD 存储卡)或 `-skin` 参数(指定仿真器以不同分辨率与显示方向)外，`emulator` 还有相当多的参数可供启动 Android 仿真器时采用。

这个实时追踪的 GPS 卫星定位软件。请到 <http://andappstore.com> 网站下载 GPSTracker.apk，然后运行 `adb` 安装，或者直接使用 AndAppStore Client 直接下载安装 GPS Tracker。

```
emulator -timezone Asia/Shanghai
```

启动 Android 仿真器时，一般情况仿真器会自动去检测当地时间，但是若检测错误时造成时区不对，则可



以使用 `-timezone` 参数指定时区为亚洲上海 Asia/Shanghai, `-timezone` 它的格式为 Area/Location, 例如 Asia/Tokyo、America/Los_Angeles 或 Europe/Paris。

```
emulator -no-boot-anim
```

如果想要快点启动仿真器, 则可以使用 `-no-boot-anim` 省略掉每次开机时的 Android logo 动画画面, 不过实际的仿真器运行速度还是主要取决于计算机硬件配置。

```
emulator -scale auto
```

```
emulator -scale factor (factor: 0.1-3.0)
```

可以使用 `-scale` 参数调整仿真器窗口的大小。如果使用 `-scale auto` 则仿真器会自动依照计算机屏幕的分辨率来调整适当的仿真器窗口大小; 如果不满意的话, 则使用 `scale factor` 给予 0.1~3.0 之间的数值, 来调整仿真器窗口的大小, 例如 `emulator -scale 1.5`。

```
emulator -dpi-device 300
```

`-dpi-device` 可以更改仿真器的显示分辨率, 内定值为 165 dpi, 可以改成 150 dpi、200 dpi、300 dpi 等数字来指定仿真器画面分辨率。

```
emulator -skin <skinID>
```

也可以使用 `-skin` 参数来指定 Android 仿真器内定的 4 种显示模式, 请将 `<skinID>` 改为任何一种模式: HVGA-L (480×320, 水平显示)、HVGA-P (320×480, 垂直显示, 此为内定模式)、QVGA-L (320×240, 水平显示)或 QVGA-P (240×320, 垂直显示)。

```
emulator -help-keys
```

`-help-keys` 会显示操作 Android 仿真器时的键盘快捷键帮助, 例如按键盘上的 Home 键就回到主画面、F2 为仿真器上的 MENU 按键、F3 拨电话、F4 退出通话、ESCAPE 为返回键, F5 搜寻功能、F8 开启或关闭 3G 网络功能、Ctrl-F5/F6 控制音量大小, Ctrl+F11/F12 旋转显示画面为垂直或水平等常用的快捷键。

```
emulator -shell
```

`-shell` 参数相当于 `adb shell` 功能, 在启动 Android 仿真器的同时, 还开启了一个的终端机模式, 可以在 Android 操作系统中使用命令列指令。

```
emulator -data imagefile
```

```
emulator -sdcard imagefile
```

```
emulator -cache imagefile
```

使用 `-data` 或 `-sdcard` 参数之前, 必须先使用 `mksdcard` 指令生成 image file, 例如 `mksdcard 4096M data.img`, 然后运行 `emulator -data data.img` 时, Android 系统的 `/data` 目录就会使用 `data.img` 的文件空间。事实上, Android 系统将使用者的 `/data` 信息默认放在 `userdata-qemu.img` 文件, 如果是 Windows 版本 SDK, 可以在目录 `C:\Documents and Settings\<user>\.android\avd` 查找这个文件, 如果是 Linux/Mac 版本的 SDK, 则可以在 `~/.android/avd` 查找这个 `userdata-qemu.img` 文件。同理, `emulator -sdcard sdcard.img` 则会让 Android 系统的 `/sdcard` 目录使用 `sdcard.img` 的文件空间, 而 `emulator -cache cache.img` 则可以当做浏览器的暂保存保存空间。

```
emulator -wipe-data
```

`-wipe-data` 参数会将 Android 仿真器恢复到出厂设置, 所有 `/data` 的文件与个人信息都会被删除, 所以在运行 `emulator -wipe-data` 前, 确定真的不要所有的信息, 以及先前安装好的 APK 应用程序了, 才进行系统清除动作。

```
emulator -help
```

`emulator` 指令的参数选项还非常的多, 您可以运行 `emulator -help` 获得更多的参数功能。

9.85、如何进行单元测试

第一步: 首先在 `AndroidManifest.xml` 中加入下面红色代码:

作者: craining (曲阜师范大学) 个人主页: <http://craining.blog.163.com/> 邮箱: craining@163.com 323



```

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="cn.android.action" android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <uses-library android:name="android.test.runner" />
        ....
    </application>
    <uses-sdk android:minSdkVersion="6" />
    <instrumentation android:name="android.test.InstrumentationTestRunner"
        android:targetPackage="cn.android.action" android:label="Tests for My App" />
</manifest>

```

上面 targetPackage 指定的包要和应用的 package 相同。

第二步：编写单元测试代码（选择要测试的方法，右键点击“Run As”--“Android Junit Test”）：

```

import android.test.AndroidTestCase;
import android.util.Log;
public class XMLTest extends AndroidTestCase {
    public void testSomething() throws Throwable {
        Assert.assertTrue(1 + 1 == 3);
    }
}

```

9.86、Android自动化测试初探

9.86.1、捕获Activity上的Element

第一部分：前言

Android 系统下应用程序的测试现在应该还算是个新的领域，网上关于这方面的资料很多都是基于白盒测试的，一般都是基于 JUnit 框架和 Android SDK 中 android.test 等命名空间下的内容进行，但是有一个前提，那就是必须要有应用程序的源代码以提供测试接入点，但是这在很多软件公司中是不现实的。很多测试工程师做的工作是完全黑盒，基本接触不到源代码，白盒测试大部分也是由开发自己完成。

回顾一下 Windows 下的黑盒测试自动化，先前使用的是微软提供的基于 .net framework 的 UI Automation 自动化测试框架（要求版本在 .net framework 3.0 以上，即 VS.NET 2008 开发环境），对与擅长 C# 语言的人来说，使用起来确认比较好用。本人也写了基于 UI Automation 的轻量级的自动化框架，将在以后的博文中引出。

那在 Android 操作系统中能不能做类似于 UI Automation 的事情呢？不幸的是，Android 的权限控制分的非常清楚，不同程序之间的数据访问只能通过 Intent，content provider 类似的功能实现。也就是说你开发的运行在

Android 中的自动化程序想要捕获当前运行的 AUT (Application under Test) 界面上的控件等 Element (该术语引自 UI Automation, 觉得翻译成元素有点生硬, 故不作翻译) 基本不可能, 你也拿不到当前 active activity 的引用 (截止本文发帖为止, 个人暂时没有找到办法获得此引用)。

无路可走了? 模拟器里面不能走, 外面能不能走? 或许可以。

第二部分: 捕获 Activity 上的 Element

在 Android 的 SDK 中自带了一个对自动化测试比较有用的工具: hierarchyviewer (位于 SDK 的 tools 目录下)。在模拟器运行的情况下, 使用该工具可以将当前的 Activity 上的 Element 以对象树的形式展现出来, 每个 Element 所含的属性也能一一尽显。这有点像 Windows 上运行的 UI SPY, 唯一遗憾的是不支持事件的触发。不过没有关系, 可以想办法绕, 当务之急是能在自行编写的自动化测试代码里找到 Activity 上的 Element。

第一个想到的办法就是看 hierarchyviewer 源码, 不巧, 网上搜了一下, 没有资源。或许 Google 的官网上有, 但是上不去。看来只能反编译了, 找来 XJad, 暴力之。虽然反编译出来的代码很多地方提示缺少 import, 但代码基本上是正确的。看了一下, 确实也知道了许多。后来在编写代码的过程中, 确实也证明了如果想引用 hierarchyviewer.jar 这个包并调试, 还是需要知道里面的一些设置的。

创建基于 hierarchyviewer.jar 这个包的调用, 需要将它和另外两个包, ddmlib.jar (在 hierarchyviewer.jar 同级目录中有) 和 org-netbeans-api-visual.jar (需要下载并安装 netbeans, 在其安装目录中有) 一并导入到工程项目中, 因为 hierarchyviewer 的实现依附于这两个包。

想在代码中获取 Activity 上的 Element 需要进行如下几个步骤 (如果使用过 hierarchyviewer 这个工具后会发现, 自动化代码所要写的就是该工具上的使用步骤):

1. **Ensure adb running**
2. **Set adb location** (因为 hierarchyviewer 和模拟器的沟通完全是依靠 adb 做的, 所以设定正确的 adb 程序的位置至关重要, 本人就曾在这个问题上栽了半天多)
3. **Get Active Device** (这个等同动作发生在启动 hierarchyviewer 工具时)
4. **Start View Server** (等同于工具上 Start Server 菜单触发事件)
5. **Load Scene** (等同于工具上 Load View Hierarchy 菜单触发事件)
6. **Get Root View Node** (获得对象树的根节点, 这个动作在工具上点击 Load View Hierarchy 菜单后会自动加载)

7. Get Sub View Node (获得想要查找的 View Node, 这个动作在工具上点击 Load View Hierarchy 菜单后会自动加载)

说明: 上述步骤中一些名称实际上就是 hierarchyviewer 中所提供的可访问方法名称, 如 startViewServer、loadScene、rootNode 等。另外 View Node 实际上 hierarchyviewer 中的一个类, 表示的对象树上的一个 Element。

现将部分核心代码粘贴如下:

```
1. Set adb location System.setProperty("hierarchyviewer.adb", "E:\\  
\\Android\\android-sdk-windows\\tools");
```

其中 "hierarchyviewer.adb" 这个 key 是 hierarchyviewer.jar 中指定的, 后面的 value 是存放 Android SDK 的路径。这个目录必须是当前运行的模拟器所对应的 adb 的目录, 不能自行使用其他目录下 adb, 否则会发生 adb 进程异常退出的错误。

2. Get Active Device

```
IDevice[] devices = null;  
  
DeviceBridge.terminate();  
  
while(null==devices || 0==devices.length){  
  
DeviceBridge.initDebugBridge() ;  
  
//it must wait for some time, otherwise will throw exception  
  
try {  
  
        Thread.sleep(1000);  
  
} catch (InterruptedException e) {  
  
e.printStackTrace();  
  
}  
  
devices = DeviceBridge.getDevices() ;  
  
}  
  
return devices;
```

以上方法返回的是所有当前运行的 Device 列表



3. Start View Server

```
DeviceBridge.startViewServer(device);
```

4. Load Scene

```
ViewHierarchyLoader.loadScene(device, Window.FOCUSED_WINDOW) ;
```

5. Get Root View Node

```
vhs.getRoot() ;
```

其中 vhs 是 ViewHierarchyScene 的实例对象

6. Get Sub View Node

```
public ViewNode findFirstChildrenElement(IDevice device, ViewNode entryViewNode, String
elementID) {

    ViewNode node=null;

    if(0!=entryViewNode.children.size()){

        for(int i=0;i<entryViewNode.children.size();i++){

            node=entryViewNode.children.get(i);

            if(node.id==elementID)

                return node;

            else

                continue;

        }

    }

    return node;}


```

虽然上述步骤所涉及的代码量不多，但是花了一天多的时间才终究研究出来，写下此文希望对正在研究 Android 自动化测试的同学们有些帮助。



到目前为止, Element 已经得到了, 接下去就是实现怎么去触发这些 Element, 如 click button, enter text 等等, 尚需再慢慢研究, 等有结果再贴出来分享!

9.86.2、Hierarchyviewer 捕获Element的

Android SDK tools 下的工具 hierarchyviewer 可以展现 Device 上的 Element 的层次分布和自身属性, 其核心函数之一就是 LoadScene, 研究后发现其实现方法是向 Device 的 4939 端口通过 socket 的方式发送了一个 DUMP 的命令, Device 会自动处理该命令并将所有 Screen 上的 Element 层次结构和属性一并发回, 实现代码如下:

```
public static void listElement(IDevice device){

    Socket socket;

    BufferedReader in;

    BufferedWriter out;

    socket = null;

    in = null;

    out = null;

    do{

        DeviceBridge.setupDeviceForward(device);

    }while(4939!=DeviceBridge.getDeviceLocalPort(device));

    socket = new Socket();

    try {

        socket.connect(new InetSocketAddress("127.0.0.1", DeviceBridge.getDeviceLocalPort(device)));

        out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

        in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
```




```
System.out.println("==> DUMP");

out.write((new StringBuilder()).append("DUMP -1").toString());

out.newLine();

        out.flush();

        do

        {

                String line;

if ((line = in.readLine()) == null || "DONE.".equalsIgnoreCase(line))

                break;

                line = line.trim();

                System.out.println(line);

        } while (true);

    } catch (IOException e) {

        e.printStackTrace();

    }

}
```

运行后的结果摘录其中一部分 (button5)，列举如下。注：当前 device 中运行的是 2.1 SDK 中自带的 Calculator 程序：

```
com.android.calculator2.ColorButton@43b8bee8 mText=1, 5
getEllipsize()=4, null mMinWidth=1, 0 mMinHeight=1, 0 mMeasuredWidth=2, 79
mPaddingBottom=1, 0 mPaddingLeft=1, 0 mPaddingRight=1, 0 mPaddingTop=1, 0
mMeasuredHeight=2, 78 mLeft=2, 81
mPrivateFlags_DRAWING_CACHE_INVALID=3, 0x0 mPrivateFlags_DRAWN=4, 0x20
mPrivateFlags=8, 16779312 mID=9, id/digit5 mRight=3, 160 mScrollX=1, 0
mScrollY=1, 0 mTop=1, 0 mBottom=2, 78 mUserPaddingBottom=1, 0
mUserPaddingRight=1, 0 mViewFlags=9, 402669569 getBaseline()=2, 54
getHeight()=2, 78 layout_gravity=4, NONE layout_weight=3, 1. 0
```

```

layout_bottomMargin=1,0 layout_leftMargin=1,1 layout_rightMargin=1,0
layout_topMargin=1,0 layout_height=11,FILL_PARENT
layout_width=11,FILL_PARENT getTag()=4,null getVisibility()=7,VISIBLE
getWidth()=2,79 hasFocus()=5,false isClickable()=4,true
isDrawingCacheEnabled()=5,false isEnabled()=4,true
isFocusable()=4,true isFocusableInTouchMode()=5,false
isFocused()=5,false isHapticFeedbackEnabled()=4,true
isInTouchMode()=4,true isOpaque()=5,false isSelected()=5,false
isSoundEffectsEnabled()=4,true willNotCacheDrawing()=5,false
willNotDraw()=5,false

```

另外还支持如下命令：

- LIST will show the list of windows:

LIST

43514758 com.android.launcher/com.android.launcher.Launcher

4359e4d0 TrackingView

435b00a0 StatusBarExpanded

43463710 StatusBar

43484c58 Keyguard

DONE.

9.86.3、架构实现

前两节讲了用 Android SDK 自带的 tool-hierarchyviewer 来捕获 Activity 上 Element，并分析了其中的原理。对于要实现 GUI 自动化，还有哪些工作没有完成呢？

- n Invoke 界面上的 Element，如点击按钮，在文本框中输入内容等
- n Press 手机自身所有的按键，如 HOME 键，Menu 键，左右上下方向键，通话键，挂机键等
- n 判断测试结果

前面说过，直接从 Emulator 内部获取当前 Activity 上的 Element 这条路已经断了，同理，探索像 UI Automation 上一样 Invoke Element 的操作估计是行不通了，因为你拿不到 Element 的对象实例，所以实例所支持的方法当然也没有办法拿到。

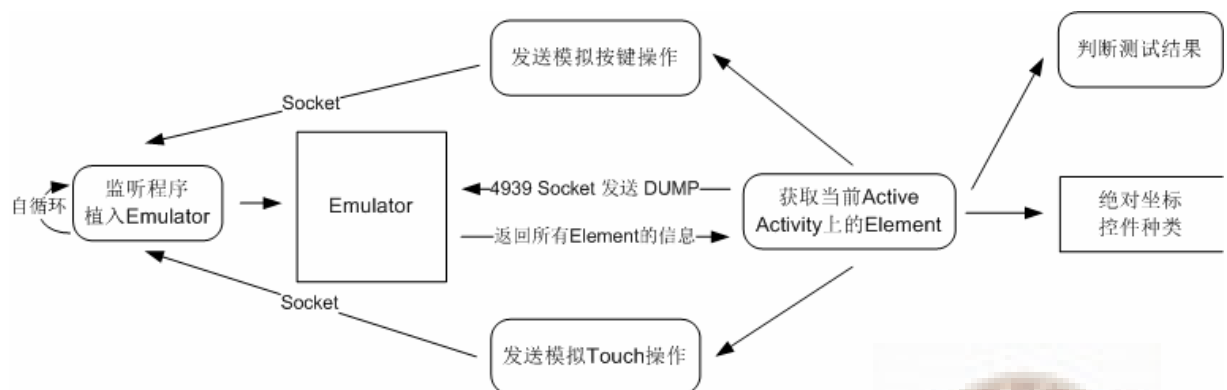
怎么办？实在不行，基于坐标来对 Element 进行触发总可以吧。在 Windows 中发送基于坐标发送键盘和鼠标事件一般是在无法识别 Element 的情况下，想的最后一招，这使我想起起了 Android 中的 monkey 测试，对着屏幕就是一通乱点，压根就不管点的是什么。所幸的是，当前 Android 系统中我们得到了 Element 的属性信息，其中就包括坐标信息，而且这种信息是具有弹性的，也就是说即使 Element 的坐标随着开发的改变而有所变化，也不用担心，因为当前的坐标是实时获得的。

那么怎样才能给 Element 发送模拟按键等操作呢？总不能用 Windows 当前的键盘和鼠标事件吧，那样一旦模拟器的位置改变或失去焦点，啥都白搭，风险太大了。看来给 Emulator 内部发送模拟按键等操作比较靠谱。查了一下 SDK，其中确实有这样的方法存在，但是我们当前的测试基础架构程序位于 Emulator 外部，怎么办？突然想起了 hierarchyviewer 的实现机制，通过 Socket 来发送信息。Hierarchyviewer 有系统自带的进程给予答复响应（具体是哪个进程进行的响应不清楚，没有研究过）。那么我们也来模拟做一个 Listener 总可以吧。

其实对于模拟按键发送，网上的帖子很多，但大部分是基于一种方式实现的，IWindowManager 接口。不巧的是，SDK 并没有将该接口提供为 public，所以需要用到 android 源码替代 android.jar 包进行编译的方式进行绕行，感觉方法有点复杂。在后面另一篇系列文章中我会列出我在网上看到的另一种基于 Instrumentation 和 MessageQueue 的机制实现方法。

最后就剩下判断测试结果了。判断测试结果一般分为如下两种：外部条件是否满足，如文件是否产生，数据是否生成等；内部条件是否满足，如对应的 Element 是否出现或消失，Element 上内容如字符串是否有变化。对于前一种本文不予讨论，后一种情况下，Element 出现或消失可以通过 hierarchyviewer 来获取。仔细研究过 hierarchyviewer 会发现，它并没有提供 Element 界面上内容（Text）的属性。这可有点晕了，好像又要回到实现捕获 Activity 实例的老路上来了。考虑图像识别？这好像不靠谱。突然想到，4939 端口上发送 DUMP 命令后的返回结果中会不会有此类 hierarchyviewer 没有显示出来的信息呢，万幸，还真有。在我上一篇博文（Hierarchyviewer 捕获 Element 的实现原理）中查询 mText 字段，会发现 mText=1, 5 这样的信息，其实就是代表了计算器 Button5 上显示的内容 5，逗号前的 1 表示后跟一位信息。

至此，问题似乎都解决掉了。画个基础架构图做个总结：



9.86.4、模拟键盘鼠标事件（Socket+Instrumentation实现）

通过 Socket + Instrumentation 实现模拟键盘鼠标事件主要通过以下三个部分组成：

1 Socket 编程：实现 PC 和 Emulator 通讯，并进行循环监听

1 Service 服务：将 Socket 的监听程序放在 Service 中，从而达到后台运行的目的。这里要说明的是启动服务有两种方式，bindService 和 startService，两者的区别是，前者会使启动的 Service 随着启动 Service 的 Activity 的消亡而消亡，而 startService 则不会这样，除非显式调用 stopService，否则一直会在后台运行因为 Service 需要通过一个 Activity 来进行启动，所以采用 startService 更适合当前的情形

1 Instrumentation 发送键盘鼠标事件：Instrumentation 提供了丰富的以 send 开头的函数接口来实现模拟键盘鼠标，如下所述：

```
sendCharacterSync(int keyCode)           //用于发送指定 KeyCode 的按键

sendKeyDownUpSync(int key)              //用于发送指定 KeyCode 的按键

sendPointerSync(MotionEvent event)      //用于模拟 Touch

sendStringSync(String text)             //用于发送字符串
```

注意：以上函数必须通过 Message 的形式抛到 Message 队列中。如果直接进行调用加会导致程序崩溃。

对于 Socket 编程和 Service 网上有很多成功的范例，此文不再累述，下面着重介绍一下发送键盘鼠标模拟事件的代码：

1. 发送键盘 KeyCode:

步骤 1. 声明类 handler 变量

```
private static Handler handler;
```

步骤 2. 循环处理 Message

//在 Activity 的 onCreate 方法中对下列函数进行调用

```
private void createMessageHandleThread() {
```

```
    //need start a thread to raise looper, otherwise it will be blocked
```

```
    Thread t = new Thread() {
```



```
public void run() {

    Log.i( TAG, "Creating handler ..." );

    Looper.prepare();

    handler = new Handler() {

        public void handleMessage(Message msg) {

            //process incoming messages here

        }

    };

    Looper.loop();

    Log.i( TAG, "Looper thread ends" );

}

};

t.start();

}
```

步骤 3. 在接收到 Socket 中的传递信息后抛出 Message

```
handler.post( new Runnable() {

    public void run() {

        Instrumentation inst=new Instrumentation();

        inst.sendKeyDownUpSync(keyCode);

    }

} );
```

1. Touch 指定坐标，如下例子即

touch point (240,400)

```
Instrumentation inst=new Instrumentation();
```



```
inst.sendPointerSync(MotionEvent.obtain(SystemClock.uptimeMillis(), SystemClock.uptimeMillis(
), MotionEvent.ACTION_DOWN, 240, 400, 0));
```

```
inst.sendPointerSync(MotionEvent.obtain(SystemClock.uptimeMillis(), SystemClock.uptimeMillis(
), MotionEvent.ACTION_UP, 240, 400, 0));
```

3. 模拟滑动轨迹

将上述方法中间添加 MotionEvent.ACTION_MOVE

9.86.5、再述模拟键盘鼠标事件（adb shell 实现）

上一篇博文中讲述了通过 Socket 编程从外部向 Emulator 发送键盘鼠标模拟事件，貌似实现细节有点复杂。其实 Android 还有一种更简单的模拟键盘鼠标事件的方法，那就是通过使用 adb shell 命令。

1. 发送键盘事件：

命令格式 1: adb shell input keyevent “value”

其中 value 以及对应的 key code 如下表所列：

KeyEvent Value	KEYCODE	Comment
0	KEYCODE_UNKNOWN	
1	KEYCODE_MENU	
2	KEYCODE_SOFT_RIGHT	
3	KEYCODE_HOME	
4	KEYCODE_BACK	

在 SDK2.1 的模拟器中命令失效，sendevent 命令可行



5

KEYCODE_CALL

6

KEYCODE_ENDCALL

7

KEYCODE_0

8

KEYCODE_1

9

KEYCODE_2

10

KEYCODE_3

11

KEYCODE_4

12

KEYCODE_5

13

KEYCODE_6

14

KEYCODE_7

15

KEYCODE_8



16

KEYCODE_9

17

KEYCODE_STAR

18

KEYCODE_POUND

19

KEYCODE_DPAD_UP

20

KEYCODE_DPAD_DOWN

21

KEYCODE_DPAD_LEFT

22

KEYCODE_DPAD_RIGHT

23

KEYCODE_DPAD_CENTER

24

KEYCODE_VOLUME_UP

25

KEYCODE_VOLUME_DOWN

26

KEYCODE_POWER



27

KEYCODE_CAMERA

28

KEYCODE_CLEAR

29

KEYCODE_A

30

KEYCODE_B

31

KEYCODE_C

32

KEYCODE_D

33

KEYCODE_E

34

KEYCODE_F

35

KEYCODE_G

36

KEYCODE_H

37

KEYCODE_I



38

KEYCODE_J

39

KEYCODE_K

40

KEYCODE_L

41

KEYCODE_M

42

KEYCODE_N

43

KEYCODE_O

44

KEYCODE_P

45

KEYCODE_Q

46

KEYCODE_R

47

KEYCODE_S

48

KEYCODE_T



49

KEYCODE_U

50

KEYCODE_V

51

KEYCODE_W

52

KEYCODE_X

53

KEYCODE_Y

54

KEYCODE_Z

55

KEYCODE_COMMA

56

KEYCODE_PERIOD

57

KEYCODE_ALT_LEFT

58

KEYCODE_ALT_RIGHT

59

KEYCODE_SHIFT_LEFT



60

KEYCODE_SHIFT_RIGHT

61

KEYCODE_TAB

62

KEYCODE_SPACE

63

KEYCODE_SYM

64

KEYCODE_EXPLORER

65

KEYCODE_ENVELOPE

66

KEYCODE_ENTER

67

KEYCODE_DEL

68

KEYCODE_GRAVE

69

KEYCODE_MINUS

70

KEYCODE_EQUALS



71

KEYCODE_LEFT_BRACKET

72

KEYCODE_RIGHT_BRACKET

73

KEYCODE_BACKSLASH

74

KEYCODE_SEMICOLON

75

KEYCODE_APOSTROPHE

76

KEYCODE_SLASH

77

KEYCODE_AT

78

KEYCODE_NUM

79

KEYCODE_HEADSETHOOK

80

KEYCODE_FOCUS

81

KEYCODE_PLUS



82

KEYCODE_MENU

83

KEYCODE_NOTIFICATION

84

KEYCODE_SEARCH

85

TAG_LAST_KEYCODE

命令格式 2: adb shell sendevent [device] [type] [code] [value]

如:

adb shell sendevent /dev/input/event0 1 229 1 代表按下按下 menu 键

adb shell sendevent /dev/input/event0 1 229 0 代表按下松开 menu 键

说明: 上述的命令需组合使用

另外所知道的命令如下:

Key Name	CODE
MENU	229
HOME	102
BACK (back button)	158
CALL (call button)	231
END (end call button)	107

2. 发送鼠标事件(Touch):

命令格式: adb shell sendevent [device] [type] [code] [value]

情况 1: 在某坐标点上 touch



如在屏幕的 x 坐标为 40, y 坐标为 210 的点上 touch 一下, 命令如下

```
adb shell sendevent /dev/input/event0 3 0 40

adb shell sendevent /dev/input/event0 3 1 210

adb shell sendevent /dev/input/event0 1 330 1 //touch

adb shell sendevent /dev/input/event0 0 0 0           //it must have

adb shell sendevent /dev/input/event0 1 330 0 //untouch

adb shell sendevent /dev/input/event0 0 0 0 //it must have
```

注: 以上六组命令必须配合使用, 缺一不可

情况 2: 模拟滑动轨迹 (可下载并采用 aPaint 软件进行试验)

如下例是在 aPaint 软件上画出一条开始于 (100, 200), 止于 (108, 200) 的水平直线

```
adb shell sendevent /dev/input/event0 3 0 100 //start from point (100,200)

adb shell sendevent /dev/input/event0 3 1 200

adb shell sendevent /dev/input/event0 1 330 1 //touch

adb shell sendevent /dev/input/event0 0 0 0

adb shell sendevent /dev/input/event0 3 0 101 //step to point (101,200)

adb shell sendevent /dev/input/event0 0 0 0

.....

                        //must list each step, here just skip

adb shell sendevent /dev/input/event0 3 0 108 //end point (108,200)

adb shell sendevent /dev/input/event0 0 0 0

adb shell sendevent /dev/input/event0 1 330 0 //untouch

adb shell sendevent /dev/input/event0 0 0 0
```

键盘响应



我们继续讨论下 Android 游戏开发的一些前置知识，平时设计自己的显示类 View 时需要捕获按键事件，比如 KeyEvent、首先引入 android.view.KeyEvent 类，直接重写 onKeyDown 方法，同样在键盘上每个按钮都对应一个 Scancode 扫描码，详细的定义在 KeyEvent 类中有，直接查看 Android SDK 中的定义，实现的方法如下：

```
public boolean onKeyDown(int keyCode, KeyEvent msg) {
    if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER) {
        // 按下中键时触发的事件，这里 android123.com.cn 提醒网友 G1 或 ADP1 使用的是轨迹球，这个 Trackball 仍然可以按下的，不仅仅是方向的导航。
        return (true);
    }
    if (keyCode == KeyEvent.KEYCODE_DPAD_LEFT) {
        //向左
        return (true);
    }
    if (keyCode == KeyEvent.KEYCODE_DPAD_RIGHT) {
        //向右
        return (true);
    }
    if (keyCode == KeyEvent.KEYCODE_DPAD_UP) {
        //向上
        return (true);
    }
    if (keyCode == KeyEvent.KEYCODE_DPAD_DOWN) {
        //向下
        return (true);
    }
    return super.onKeyDown(keyCode, msg);
}
```

9.87、反编译APK

方法 0。(推荐)

使用附带的工具包-----APK 反编译工具.rar，解压后有使用说明

方法一：

工具下载：需用到 dex2jar 和 JD-GUI 这 2 个工具

dex2jar 下载地址：<http://laichao.googlecode.com/files/dex2jar-0.0.7-SNAPSHOT.zip>

作者：craining（曲阜师范大学）个人主页：<http://craining.blog.163.com/> 邮箱：craining@163.com 344



JD-GUI 下载地址:

windows 版 JD-GUI: <http://laichao.googlecode.com/files/jdgui.zip>

Linux 版 JD-GUI: <http://laichao.googlecode.com/files/jd-gui-0.3.2.linux.i686.tar.gz>

步骤:

1. 首先找到 Android 软件安装包中的 classes.dex:

把.apk 文件改名为.zip, 然后解压缩, 得到其中的 classes.dex 文件, 它就是 java 文件编译再通过 dx 工具打包成的, 所以我们就用上述提到的 2 个工具来逆方向导出 java 源文件

2. 把 classes.dex 拷贝到 dex2jar.bat 所在目录。

在命令行模式下定位到 dex2jar.bat 所在目录, 运行 `dex2jar.bat classes.dex`, 生成 classes.dex.dex2jar.jar

3. 运行 JD-GUI 工具 (它是绿色无须安装的)

打开上面的 jar 文件, 即可看到源代码

方法二: (推荐)

反编译 apk 生成程序的源代码和图片、XML 配置、语言资源等文件。

转载自: http://blog.sina.com.cn/s/blog_5752764e0100kv34.html

工具下载:

在 <http://code.google.com/p/android-apktool/> 下载获得, apktool-1.0.0.tar.bz2 和 apktool-install-windows-2.1_r01-1.zip 两个包都要下。

下载后解开, 为了方便使用, 按作者的推荐把得到的 4 个文件复制到 C:\Windows 文件夹里。

这个工具是 Java 写的, 需要你的电脑安装了 JRE 或者 JDK, 并在系统环境变量 Path 里加入 java.exe 所在路径。

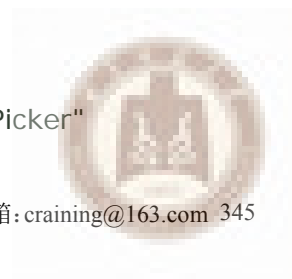
以上准备工作做好后, 就可以用它来反编译 APK 了。这里我用动态壁纸的 APK 来做示范。

如果用过动态壁纸, 你也许会发现在设置壁纸时, 界面的显示有点问题: “动态壁纸”, 在手机上中文显示为 “当前壁纸”。这是 “动态壁纸选择器” LiveWallpapersPicker.apk 的翻译错误造成的。

(假设 LiveWallpapersPicker.apk 放在 C 盘根目录)

开始 > 运行, 输入 cmd, 回车。

`apktool d "C:\LiveWallpapersPicker.apk" "C:\LiveWallpapersPicker"`



（命令行解释：apktool d 要反编译的文件 输出文件夹）

这样，LiveWallpapersPicker.apk 就被反编译了，输出内容在 C:\LiveWallpapersPicker 文件夹里。

打开 C:\LiveWallpapersPicker 文件夹，我们发现里面有一些 XML 文件和一些文件夹。绝大部分情况下，语言和图片资源都在 res 文件夹里，我们这个例子当然不例外。

打开 res 文件夹，可以其中又是很多文件夹。（又是）绝大部分情况下，语言资源都放在 values* 文件夹里。比如说 values 放默认语言（英语居多），values-de 放德语，values-fr 放法语等等。一般我们关心的是 values-zh-rCN（简体中文）和 values-zh-rTW（繁体中文）。

打开 values-zh-rCN 文件夹，其中有个 strings.xml。打开看看：

```
<?xml version="1.0" encoding="UTF-8"?>
<resources>
  <string name="application_name">动态壁纸选择器</string>
  <string name="live_wallpaper_picker_title">当前壁纸</string>
  <string name="live_wallpaper_preview_title">当前壁纸预览</string>
  <string name="configure_wallpaper">设置...</string>
  <string name="wallpaper_instructions">设置壁纸</string>
  <string name="live_wallpaper_empty">无当前壁纸。</string>
  <string name="set_live_wallpaper">设置壁纸</string>
  <string name="wallpaper_title_and_author">%1$s 提供者：%2$s</string>
  <string name="live_wallpaper_loading">正在载入当前壁纸...</string>
</resources>
```

很好，就是它了。把“当前壁纸”都改为“动态壁纸”，再检查和修正标点——中文内容用中文标点（强烈呼吁大家注意标点问题，目前马大哈太多了）后，保存。

本例改这么多就够了。其它复杂的 APK 建议把 res 文件夹里的内容都检查下，至少你关心的语言文件夹里的内容都检查下。

改完后，就可以重打包了。还是在 cmd 命令行里，输入：

```
apktool b "C:\LiveWallpapersPicker"
```

（命令行解释：apktool b 要打包内容所在文件夹）

就可以了。生成的 APK 在 C:\LiveWallpapersPicker\dist 文件夹里，叫 out.apk。

这个out.apk是没有签名的，所以不能直接装到手机里。签名工具和方法见 [43 条](#)，这里不说了。

签名后得到的 APK，就是可以装到手机里的了。

三. 软件去广告实例



1.解压 apktool.jar 到 C:\Windows 解压 apktool-install-windows.zip 到任意文件夹(例如 E 盘根目录)

2.Win+R 运行 CMD,用 cd 命令转到 apktool-install-windows 所在文件夹,输入 apktool 看看。会列出一些帮助的话就成功了。Apktool 命令

```
apktool d XXX.apk ABC 反编译 XXX.apk 到文件夹 ABC
apktool b ABC          从文件夹 ABC 重建 APK, 输出到 ABC\dist\out.apk
```

然后我们反编译一枚软件玩玩...

AutoMemoryManager 的免费版底部有一条广告, 去掉它吧。

把 com.lim.android.automemman.apk 放到同文件夹(我的就是 E 盘根目录)

Win+R 运行 CMD

E:<回车>

E:\>apktool d com.lim.android.automemman.apk AMM <回车>

I: Baksmaling...

I: Decoding resource table...

I: Decoding resources...

I: Copying assets and libs...

现在文件被 decode 到 E:\AMM 了, 打开 E:\AMM\res\layout\main.xml 看, 所有都可见了吧~
编辑第 59 行

```
<com.admob.android.ads.AdView android:id="@id/ad" android:layout_width="fill_parent"
android:layout_height="wrap_content" admobSdk:backgroundColor="#ff000000"
admobSdk:textColor="#ffffff" admobSdk:keywords="Android application" />
```

改为

```
<com.admob.android.ads.AdView android:id="@id/ad" android:layout_width="0.0dip"
android:layout_height="0.0dip" admobSdk:backgroundColor="#ff000000"
admobSdk:textColor="#ffffff" admobSdk:keywords="Android application" />
```

然后 CMD 输入

E:\>apktool b AMM

I: Checking whether sources has changed...

I: Smaling...

I: Checking whether resources has changed...



I: Building resources...

I: Building apk file...

用 Auto-sign 签名 E:\AMM\dist\out.apk 安装
这样广告就不见了

再看,嘿嘿~很帅吧...

这其实就是改了 AndroidManifest.xml 里的 ADmob 广告 ID 罢了

9.88、更换APK图标(签名打包)

方法一.

前几天见到坛里有人问怎样美化 APK 软件图标,其实已经有高手以前发过了,可能帖子太久沉了吧,很多机友没找到,在下不才,再次发个(原帖我忘地址了,谢谢原作者!他原帖有些地方没说明清楚的,我补充上了!)。

要个性化软件图标且不丢失,必须在 PC 端完成。

首先保证你的 PC 能运行 JAVA 环境和下载好 APK 签名软件 APK-sign 并放置在 C 盘根目录(就是这两点原帖作

者没说明清楚的,可上 JAVA 官网下载 JAVA 安装,签名软件 APK-sign 解压后放置在 C 盘根目录!!)

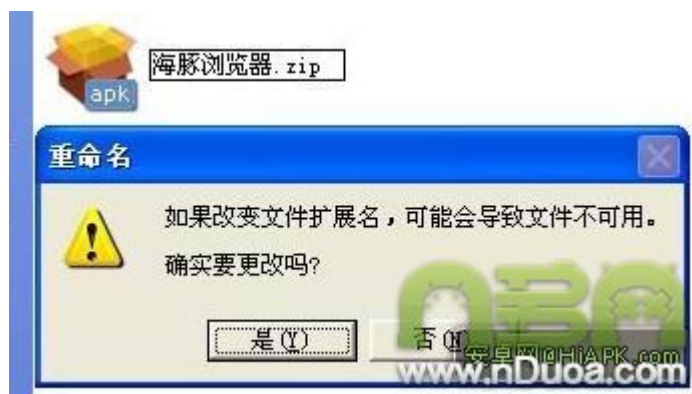


美化图标前要找好图标素材 PNG 格式图片,并转换好大小(大部分软件图标大小都是 72x72\48x48\36x36 的 PNG 图片)。准备好了后就可以工作了。。

选定要美化图标的 APK 软件(我用海豚浏览器做个例),先修改 APK 软件



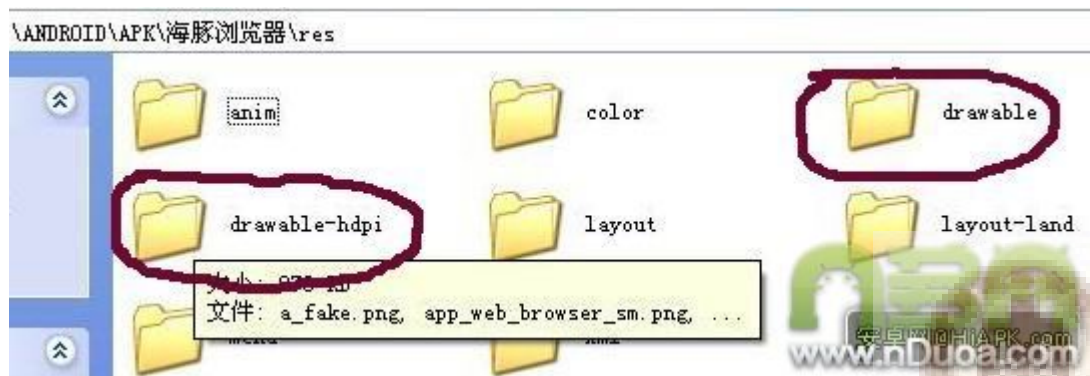
后缀为 ZIP 如图红框选是



修改好后直接把它解压, 解压后软件的所有信息就是下面的了。。



打开 RES 文件。。drawable-hdpi 子文件夹, 就有要换的程序图标..





把转换好的个性化图标名称修改成和原图标名称一样。。并替换它（有些软件不止一个图标，

其他文件夹也打开看看有无一样但如前文提到大小的原图标，也要一起更换!!）

替换好后把整个文件夹剪切到 APK-sign 文件夹里面，准备给它重新签名。。图



拖拉至 sign_pack.bat。。出现签名画面。。按任意键操作。。即完成软件签名操作。。



软件签名后就在 APK-sign 文件夹\海豚浏览器_cn_安卓版_www.hiapk.com 里面。。即完成美化软

件图标。。复制出来就可以安装啦！

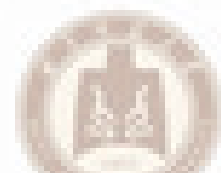
PS: RES 文件夹里其他文件夹也打开, 看看有无大小都是 72x72\48x48\36x36 的 PNG 图片。。和原图标一

样但大小不一, 也要更换!!!

下面说下个性化你的运营商名称的操作（已经有机友发过, 再罗唆一下, 手机需先 ROOT）。

下载 AndroidResEdit 汉化软件（貌似用它得装 91 助手）待用。

首先把 system/framework/framework-res.apk 复制出来。。改后缀为 ZIP。。用 WinRAR 右键打开（



别解压)...



找到里面\res\xml\plmn.xml 文件复制出来，WinRAR 别关闭。。

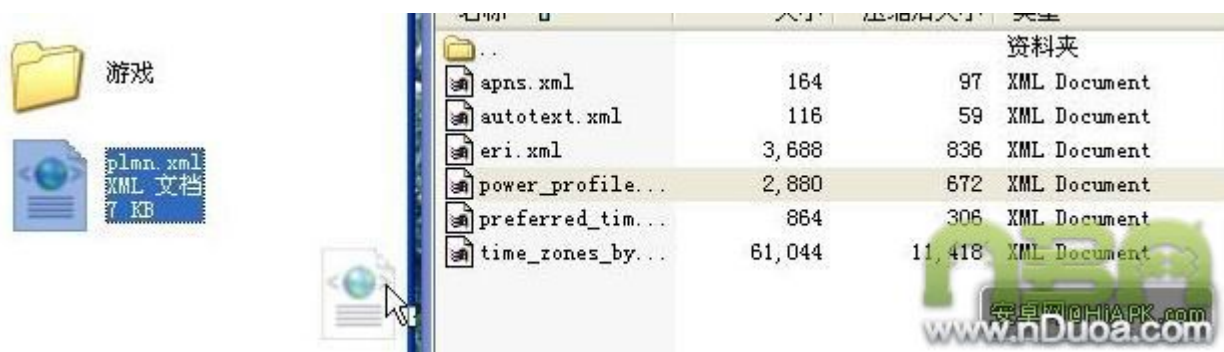


使用 AndroidResEdit 打开 plmn.xml，找到“中国移动”或“中国联通”运营商名称，修改为你想要的名称（双击运营商名称对应的替换资源空白处，跳出的提示框输入你的运营商名称，用记事本先写好再复

制，只能是小于等于四个字)...



保存 plmn 后用新的 plmn 代替原来的 plmn



关闭 WinRAR。。改 framework-res.ZIP 为 framework-res.apk，拷回卡里，用 RE 管理器把它复制到 system

按图设置 framework-res.apk 的权限。。



设置权限后移动到/framework 覆盖原来的 framework-res.apk。。重启手机，桌面上就是你的个性运营商

名称了！



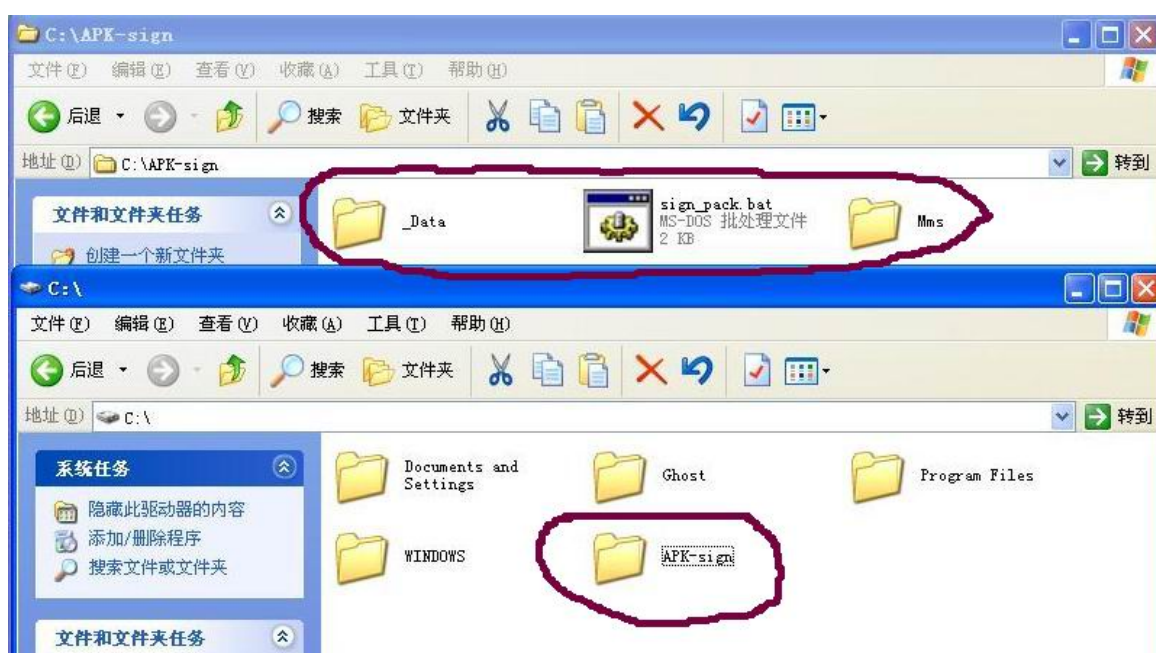
方法二.

下面就以这款名叫 Mms 的软件为例。

先要下载这个 APK 文件打包签名文件：

解压后会得到一个 sign_pack.bat 的批处理文件与一个_Data 的文件夹：





找好图标素材 PNG 格式图片并转换好大小（大部分软件图标大小都是 72x72\48x48\36x36 的 PNG 图片）。工具嘛，就是 PC 自带的 WINRAR 足以！

选定要美化图标的 APK 软件（我用短信 MMS 做个例），修改 APK 后缀为 ZIP（如图 2 红框选是），

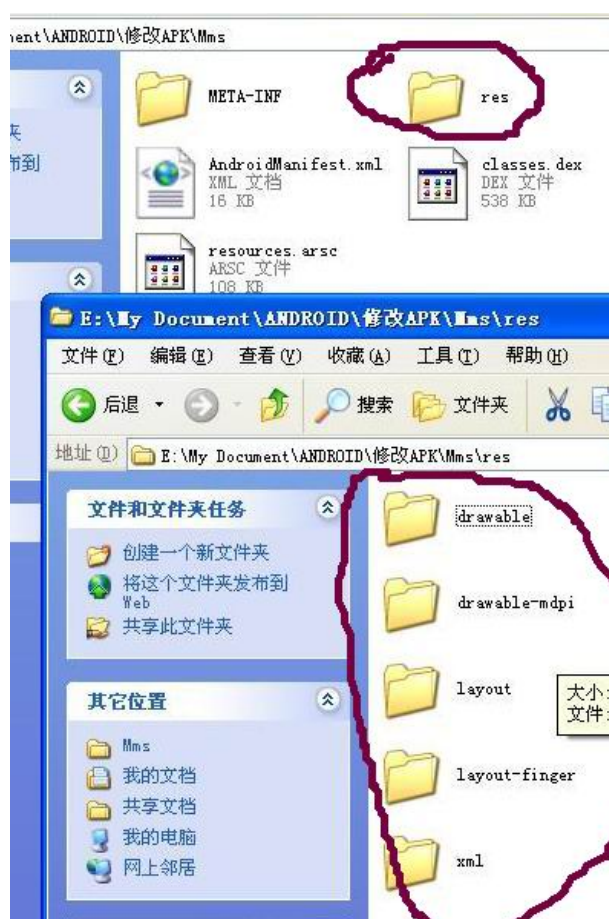


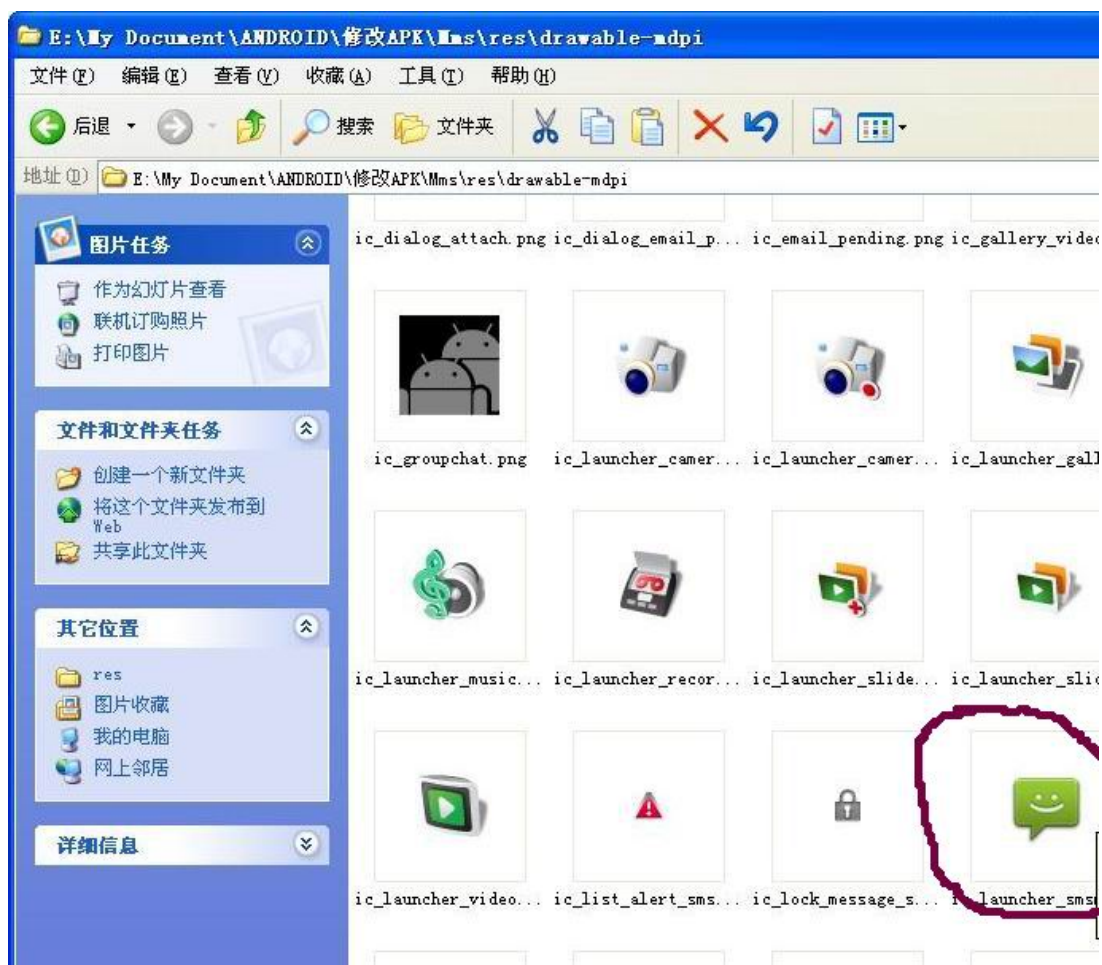
修改好后选择 ZIP 解压（如图 3），





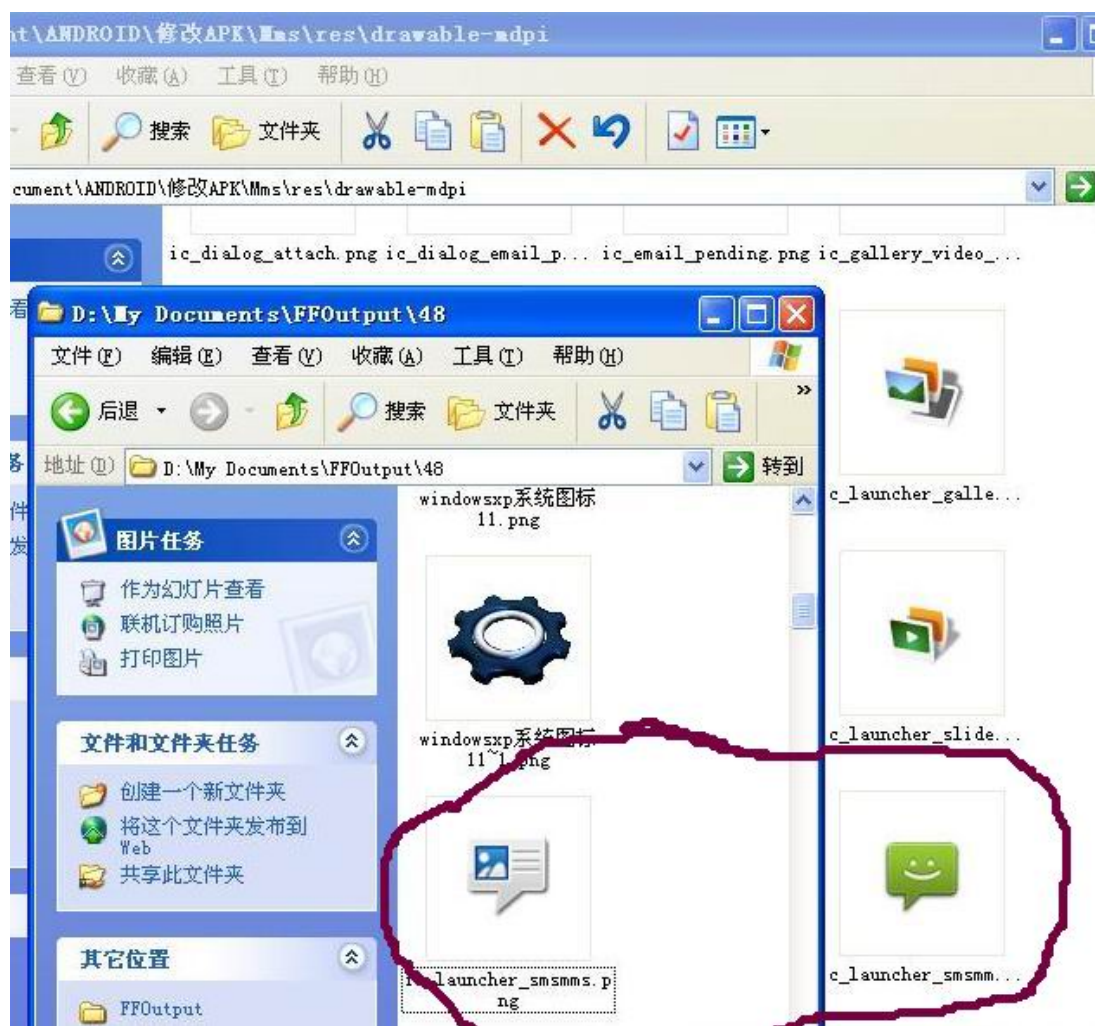
解压后 APK 的所有信息就是下面的了。。我们要用的就是里面的 RES 文件夹。。打开。。drawable-mdpi 子文件夹, 就有要换的程序图标 (如图 4、5)。





把转换好的个性化图标名称修改成和原图标名称一样。。并替换它（如图 6）

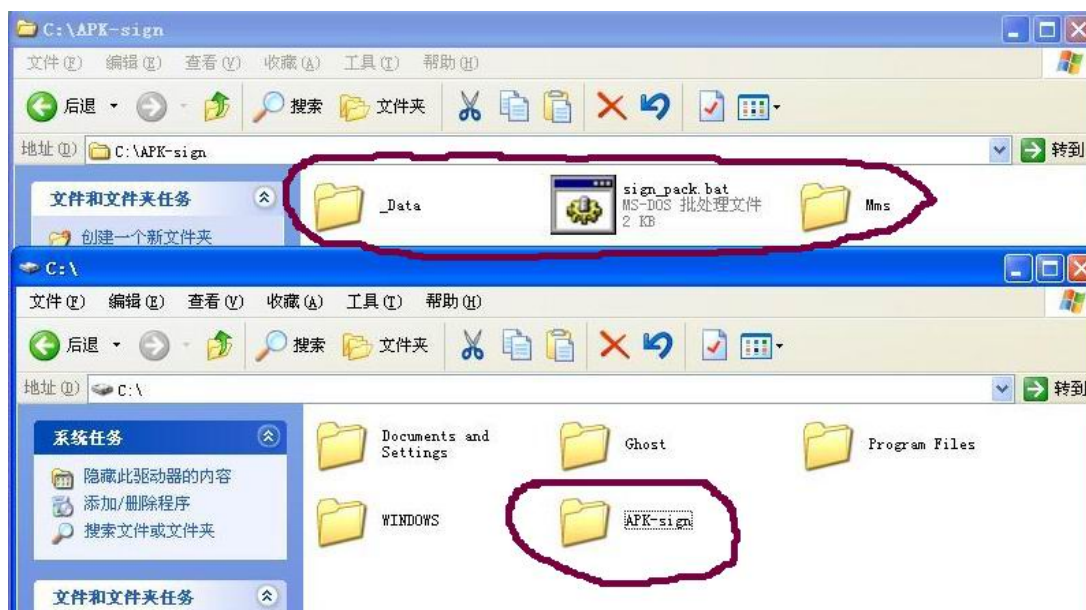




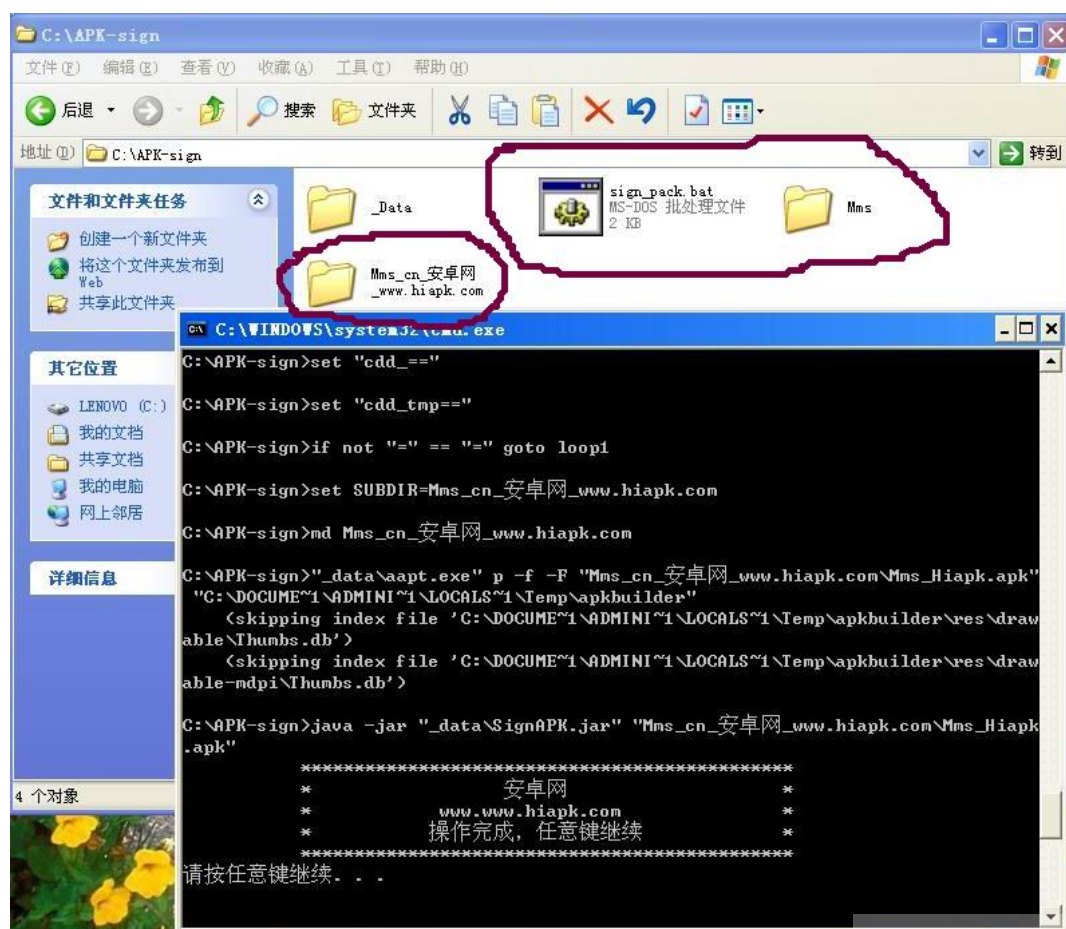
[有些软件不止一个，其他文件夹也打开看看有无一样但大小不一的图片，也要更换，一般都是如前文提到的大小！！]

替换好后把整个文件夹剪到 APK-sign 文件夹里面。。图 7





拖拉至 sign_pack.bat。。出现签名画面。。按任意键操作。。即完成软件签名操作。。图 8



软件签名后就在 APK-sign 文件夹\Mms_cn_安卓网_www.hiapk.com 里面。。图 9



即完成个性化软件图标啦！安装后看看是否你的软件图标与众不同呢？

下面的是我的所有个性化软件图标。。看有几个和你现在的不一样的呢！！



PS: RES 文件夹里其他文件夹也打开，看看有无大小都是 72x72\48x48\36x36 的 PNG 图片。。和原图标一样但大小不一，也要更换!!!



9.89、利用Android Market赚钱

随着我的第一个付费app的成功交易，我想有必要写下Android app如何认证，发布，到赚钱的几个流程，与大家共享，也希望大家有条件的可以试试开发一些免费或者付费的Android小 [软件](#)，以后移动开发的机会会很多，现在先练练手

关于如何开发 Android app，这里就不详细叙述了，外面相关的文章大把。

发布 app 前，你需要一个 google id，然后要注册成 android developer. (链接：<http://www.android.com/market/>)。

缴付 25 美刀就可以成功申请了。申请后你还可以购买 G1 的开发机子，但这种机子不能上 Market 买付费 app，因为带有 root 权限，所以一些版权限制可以很容易的被破解

如果你要发布付费软件，你需要注册 Google checkout，一个信用卡就足够了。

如果你没有美国合法报税身份的话，每个月的收入上限是\$500，多余的部分 Google 先扣住，是为了防止偷税漏税的问题。而且你需要一个银行帐号，Google 要经过认证的，才可以把收入打款到你的帐号。

目前美国的 G1 只能查看发布到美国市场的 app，UK 的只能看 UK 的，所以你一定需要一台对应你目标地区的 G1，免得到时候看不到自己的 app。

上面这些步骤设置完毕，你就可以开始发布你的 app 了。

发布的步骤很简单，先把你的 app 在 eclipse 里面生成 apk 文件。

然后通过 jarsigner 认证。认证的具体方法在这里：

<http://code.google.com/android/devel/sign-publish.html>

简单的讲就是你要生成一个 keystore 文件，然后用这个文件来 sign 你的 app，然后就可以上传到 Market 了。

中间当然还要填些表格，比如你的 app 的一些信息，归类，价格，等等。

上传后 app 会立刻发布出来，Google 没有什么 2 次认证的，所以... 灰常快

我的第一个付费app叫ShopFusion Pro。这是个去一些打折网站找你需要的折扣和促销产品的小软件。 [服务器](#)端处理不同网站的RSS数据(PHP+MySQL)。然后用JSON传输到 android app上面。这个app有个免费版本的(带广告)，名字叫ShopFusion。这个软件的前身是DealsDroid。DealsDroid是去年12月份的某一天我突发奇想做的，总共也只用了一天，目前为止有 3000 多个使用者。DealsDroid没有 [服务器](#)端，所有的RSS处理都在客户端，所以速度比较慢，特别是在GSM EDGE下面。有兴趣的朋友可以下过来试试，要用美国版的G1才可以看到的。当然你有多余的米的话，买个吧~~~ 打了这么多字也是辛苦的~~~

如果你的 app 用户量潜力上来说是巨大的，而且用户会时常使用你的 app 的，比如 twitter 客户端，SNS 插件等等，你可以考虑用广告的收入方式而发布免费的 app。

说起广告 banners，大家一定不会陌生了。最流行的就是 google adsense。但本人对 adsense 的印象很不好。自己有个 adsense 的帐号，但时不时广告都会变成公益广告，而且收入奇低，——至今未收到他们的支票...

所以，按道理上来说，Android app 和 google adsense 的结合会很紧密，但到目前为止还没有一个很好的方式。

这里介绍另外一个新兴的专攻移动广告的公司——AdMob (<http://www.admob.com>)。

AdMob 提供了专门的 Android Ad SDK，可以很方便的集成 ad 到你的 app 里面。（他们也提供 iPhone SDK, PHP, Ruby, JSP, 等等 snippets）。他们也有很详细的用户数据分析，用户习惯，等等。他们差不多每 1500 多印象(impression)我可以赚到 1 美刀。目前为止 ShopFusion 还只有 2 美刀 ——#...

我为一个波特兰的设计师开发的 Wallpaperoid(壁纸下载)和 Tunes for Android(铃声下载)的 app，每天的 PV 竟然突破到 225K... 日收入 200 多美刀... >_< 我只赚到几百块的苦力费... IT 民工真不是人当的...

不过废话短说，我们是在网页上放 AdMob 的广告。要下载壁纸或者铃声都是先要通过 G1 的内置浏览器去下载网页，然后下载。

下面简单说说在 app 里面如何放 AdMob 的广告：

首先当然是申请 AdMob 帐号，然后设置你的 app 的信息（这里就略过去了，想赚钱自己要下点功夫 ^_^）。

一切设置好后，就可以下载 AdMob 的 Android 的库文件，你把这个 jar 加入到自己的 android 项目中。

在 AndroidManifest.xml 里配置：

Xml 代码

```
<meta-data android:value="你的 ID" android:name="ADMOB_PUBLISHER_ID" /> 在你的 layout 文件
里，指定 AdMob 的 ad view:      Xml 代码  <com.admob.android.ads.AdView
android:id="@+id/ad"              android:visibility="gone"
android:layout_width="fill_parent" android:layout_height="wrap_content"
android:layout_alignParentBottom="true" app:backgroundColor="#000000"
app:textColor="#FFFFFF"           app:keywords="Android game" />
<com.admob.android.ads.AdView    android:id="@+id/ad"
android:visibility="gone"         android:layout_width="fill_parent"
```



```

android:layout_height="wrap_content"                android:layout_alignParentBottom="true"
app:backgroundColor="#000000"                      app:textColor="#FFFFFF"
app:keywords="Android game"                        />

```

compile... 就可以了。集成是相当的方便

最后说一句：放 banner 要遵守规则，不要恶意点击，切记切记...

9.90、Android-Market 使用

1、账号注册

步骤很简单，到 Android Market 发布网站建你的帐号，要收取 25 美元的注册费，地址在 <http://market.android.com/publish>。

Android Market 支持的国家和地区包括巴西、印度、墨西哥、俄罗斯、韩国、中国香港以及中国台湾等，届时支持 Android 应用商店支持的国家和地区会达到 32 个

既然没有支持中国大陆，那就填写中国香港吧，我是使用广发信用卡成功支付的（试过招商信用卡不成功支付 25\$，但成功贡献了好几次当地币 8 元），消费美元 25.00 元，当地币 8.00 元，关于 app 的收入分成比例发布人员大概能得到 70% 的钱，Google 收取 30% 的管理费。

2、应用开发、打包、签名

应用软件开发例程：我们以 www.souapp.com 最新发布的“手机条码购物软件-for dangdang 网”(market://search?q=pname:com.souapp.dangdangbook)为例，具体细节看我去年的文章“手机条码购物软件（For 当当网）--无线互联网需要创新！”

软件名称：手机条码购物软件

软件开发背景：

中国互联网网络信息中心（CNNIC）在京发布了《中国手机上网行为研究报告》，报告显示，截至 2008 年年底，中国手机用户已经超过 6.4 亿，而通过手机上网的用户数量已超过 1.176 亿。

根据市场调查越来越多的手机用户在外出时参加不同商家举办的线下购物场所营销活动，于此同时广大消费者越来越青睐于具有目录销售 B2C 网站的购物服务，为了顺应 3G 手机购物时代的趋势各大实体商家和 B2C 购物网站如何抢夺消费者，此手机条码购物软件会成为商家和购物网站用来战胜竞争对手的法宝。

先了解一下目前消费者手机购物需求：

- 随地都有可能被“忽悠”出购物欲望，例如：路过不能错过，看到购物促销活动就被拉进去了。
- 虽说都有固定的购物场所和购物网站，但是要看具体情况来选择商家了，例如：在书店挑了 20 本数，首先书的价格肯定没有当当网便宜，其次送货服务也是消费者所不得不考虑的现实问题；
- 消费者在看到物美的同时，最关心的就是商品是否价廉了，众多 B2C 购物网站都可能卖统一类产品，如何选择？恐怕只有一个办法，那就是比比价格了。
- 在 N 家 B2C 购物网站和实体商家就有 N 个会员卡，积分不能统一有效兑现，所以腾讯为啥急于推出“返利网”了。

软件功能简介：手机条码购物软件是手机上安装并使用的手机客户端软件，其主要功能它

利用扫描并识别商品的唯一标示：条形码，来作为用户操作的必要查询条件进行使用，用户的操作可以是通过条形码查询此商品的基本信息，消费者的口碑，价格信息，提供购买服务的商家等。

编码过程我就赘述了，相信每个人都会开发这种功能软件 $O(n \times n)O\sim$ 。

打包、签名，就需要你，右击工程，使用 Android Tools 的 Export Signed Application Package... 的功能

3、应用上传发布

登录:<http://market.android.com/publish>

上图，你可以看到你所有的应用都在这个清单中显示，你可以将应用 publish,也可以 unpublish, 所以为什么官方 android market 公布的应用数量只有 7 万多，而第三方的数据分析网站 <http://www.androlib.com> 公布有 14 万之多，其中重要的原因就是，androlib 在监听 androlid market 的 publish app,只要有他就加到自己的数据库里，但是如果已发布的 app 已经被作者 unpublish, androlib 依然会算到他的 app 数据中，所以你会发现在 androlib 上下载 app 时，有的 app 找不到而无法下载，简而言之 androlib 的 app 统计只做加法，不做减法。

下面我们点击具体一个 app 的详细管理页面看看，

如果您已经开发好一个 Android 应用程序，并想发布到 Android Market 上销售，仅要简单的三步骤就可以发布您的程序。

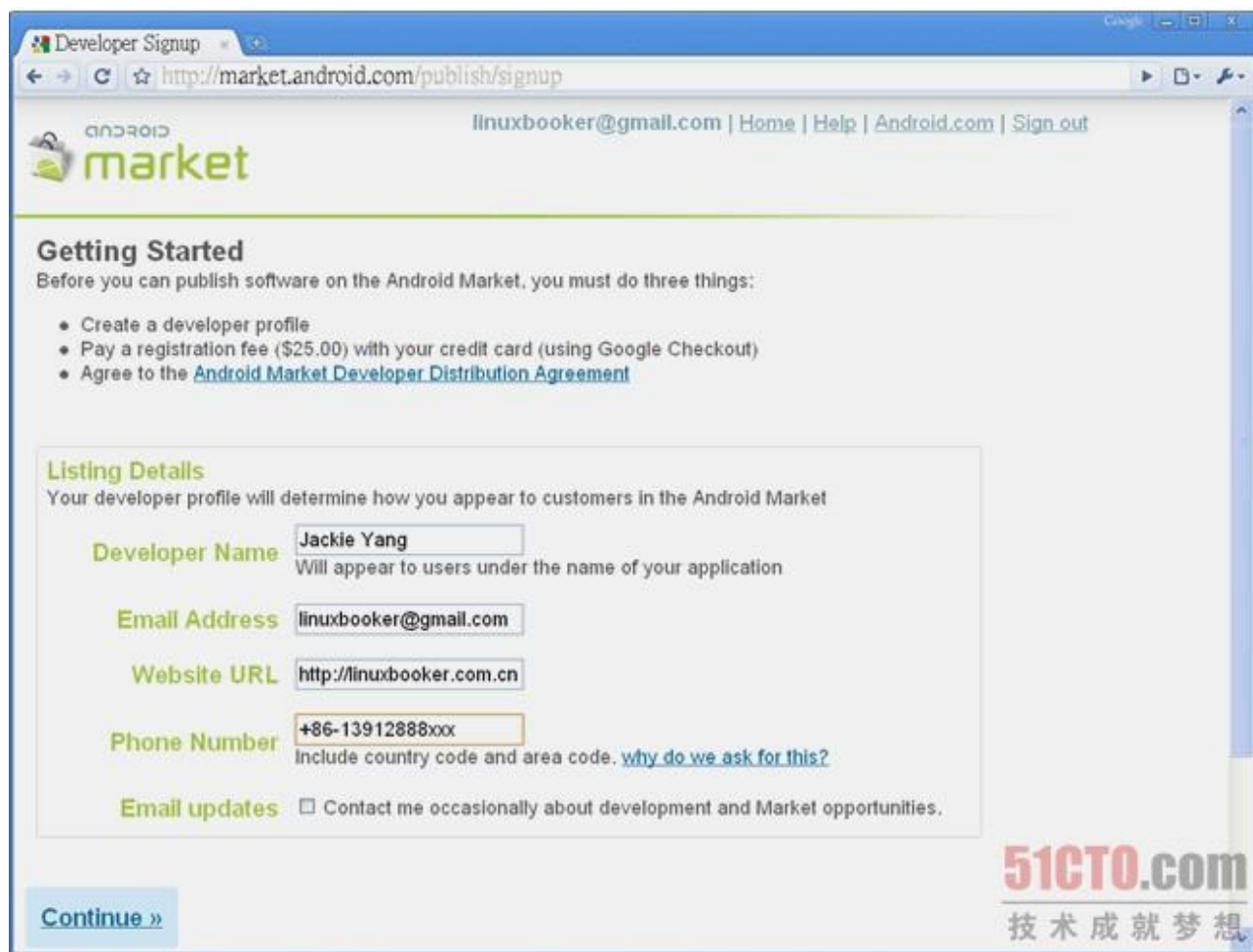
注册为商人 (注册费为 25 美元，采用信用卡支付)；

上载并描述您的应用程序功能；

发布您的应用程序。

首先要注册为商人，您只要点选 Android Market 网页中左边的"Learn more"链接，然后输入您的 Gmail 账号口令，登录后会让您填写开发程序名称、Email、网址与电话，如图 1-10 所示。





Developer Signup

http://market.android.com/publish/signup

linuxbooker@gmail.com | Home | Help | Android.com | Sign out

Getting Started

Before you can publish software on the Android Market, you must do three things:

- Create a developer profile
- Pay a registration fee (\$25.00) with your credit card (using Google Checkout)
- Agree to the [Android Market Developer Distribution Agreement](#)

Listing Details

Your developer profile will determine how you appear to customers in the Android Market

Developer Name Jackie Yang
Will appear to users under the name of your application

Email Address linuxbooker@gmail.com

Website URL http://linuxbooker.com.cn

Phone Number +86-13912888xxx
Include country code and area code. [why do we ask for this?](#)

Email updates ☐ Contact me occasionally about development and Market opportunities.

[Continue »](#)

51CTO.com
技术成就梦想

接下来 Google 会跟您索取 25 元美元的 Android Market 程序开发账号注册费用，并要求您使用信用卡在 Google Checkout 支付，如图 1-11 所示。Google 跟开发者收取 25 元美元的目的是除了控管上载免费程序的质量外，也希望藉由这次的收费督促开发者努力开发一些优秀的付费程序，让使用者下载，而将来您通过销售程序所得的 70%费用会归您所有，另外 30%的费用必须用来支付电信商跟电子收费商 Google Checkout 的手续费，而这就是 Android Market 的营利运作方式。另外，开发者可以直接在自己的 Android Market 开发网页上看到所开发的应用程序被下载的次数、使用者评分以及使用意见。

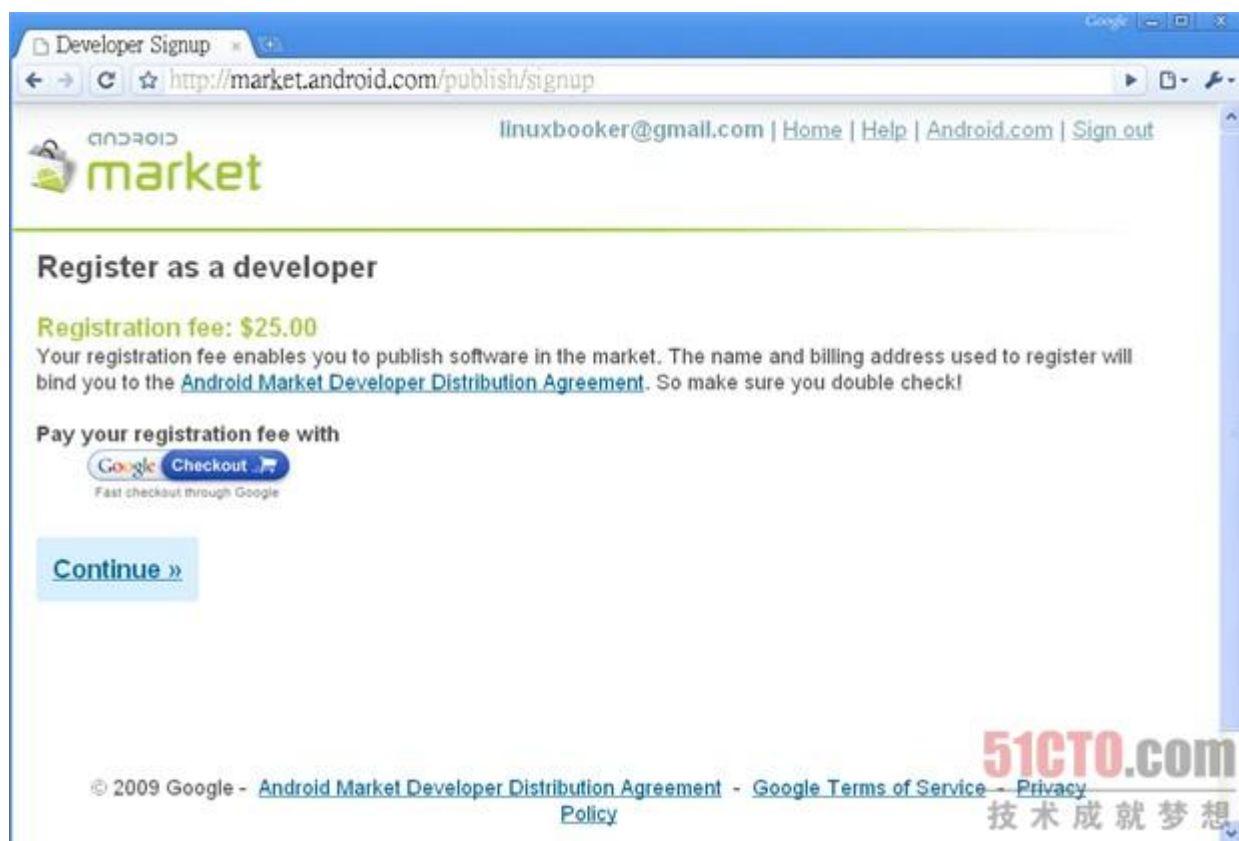


图 1-11 Android Market 注册费用 25 元美元

4、应用下载和安装

android 手机应用的文件名后缀名为".apk", 把它安装到你的手机上可以通过 2 种方式: PC 软件安装和手机内置的 market 商店程序安装。

第一种方式:PC 软件安装,这里我不截图给大家了,你可以下个"91 助手手机助手 for android" (点击可以直接下载,我这服务做到家了),或者"豌豆荚手机精灵",个人觉得到目前为止这两个软件比较适合大众人群使用(不表示其它的软件不好用呀,只是有的太过于适合开发人员使用了)。

手机条码购物软件的 apk 文件(点击可以直接下载),注意我没有使用真机做测试,前不久我的 G1 刷成板砖了~~~~(>_<)~~~~,希望下载并用过这个软件朋友提供反馈信息,谢谢。

第二种方式:手机内置的 market 商店程序安装,先在你手机里找到 market 软件商店吧,每个手机名字会有差异的

下面就看图,操作可以啦:

- 1、启动 Android Market
- 2、调用 Search 功能,输入 dangdang 或者 com.souapp.dangdangbook 都可以,
- 3、找到应用了,就点击它下载。

9.91、传感器

9.91.1、获取手机上的传感器

布局文件: **main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:id="@+id/text_show"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />
</LinearLayout>
```

Java 代码:

```
package com.ruin.book.SensorTest;
```

```
import java.util.List;
```

```
import android.app.Activity;
```

```
import android.content.Context;
```

```
import android.hardware.Sensor;
```

```
import android.hardware.SensorManager;
```

```
import android.os.Bundle;
```

```
import android.widget.TextView;
```

```
public class SensroTest extends Activity {
```

```
    /** Called when the activity is first created. */
```

```
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.main);
```

```
        // 准备显示信息的 UI 组建
```

```
        final TextView tx1 = (TextView) findViewById(R.id.text_show);
```

```
        // 从系统服务中获得传感器管理器
```

```
        SensorManager sm = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
```

```
        // 从传感器管理器中获得全部的传感器列表
```

```
        List<Sensor> allSensors = sm.getSensorList(Sensor.TYPE_ALL);
```



```
// 显示有多少个传感器
tx1.setText("经检测该手机有" + allSensors.size() + "个传感器，他们分别是：\n");

// 显示每个传感器的具体信息
for (Sensor s : allSensors) {

    String tempString = "\n" + " 设备名称：" + s.getName() + "\n"
        + " 设备版本：" + s.getVersion() + "\n" + " 供应商："
        + s.getVendor() + "\n";

    switch (s.getType()) {
    case Sensor.TYPE_ACCELEROMETER:
        tx1.setText(tx1.getText().toString() + s.getType()
            + " 加速度传感器 accelerometer" + tempString);

        break;

    case Sensor.TYPE_GYROSCOPE:
        tx1.setText(tx1.getText().toString() + s.getType()
            + " 陀螺仪传感器 gyroscope" + tempString);

        break;

    case Sensor.TYPE_LIGHT:
        tx1.setText(tx1.getText().toString() + s.getType()
            + " 环境光线传感器 light" + tempString);

        break;

    case Sensor.TYPE_MAGNETIC_FIELD:
        tx1.setText(tx1.getText().toString() + s.getType()
            + " 电磁场传感器 magnetic field" + tempString);

        break;

    case Sensor.TYPE_ORIENTATION:
        tx1.setText(tx1.getText().toString() + s.getType()
            + " 方向传感器 orientation" + tempString);

        break;

    case Sensor.TYPE_PRESSURE:
        tx1.setText(tx1.getText().toString() + s.getType()
            + " 压力传感器 pressure" + tempString);

        break;
    }
```



```

        + " 压力传感器 pressure" + tempString);

        break;

    case Sensor.TYPE_PROXIMITY:
        tx1.setText(tx1.getText().toString() + s.getType()
            + " 距离传感器 proximity" + tempString);

        break;

    case Sensor.TYPE_TEMPERATURE:
        tx1.setText(tx1.getText().toString() + s.getType()
            + " 温度传感器 temperature" + tempString);

        break;

    default:
        tx1.setText(tx1.getText().toString() + s.getType() + " 未知传感器"
            + tempString);

        break;

    }

}

}

}

```

9.91.2、

1、Android 所有的传感器都归传感器管理器 `SensorManager` 管理，获取传感器管理器的方法很简单：

```
String service_name = Context.SENSOR_SERVICE;
SensorManager sensorManager = (SensorManager) getSystemService(service_name);
```

2、现阶段 Android 支持的传感器有 8 种,它们分别是：

传感器类型常量	内部整数值	中文名称
<code>Sensor.TYPE_ACCELEROMETER</code>	1	加速度传感器
<code>Sensor.TYPE_MAGNETIC_FIELD</code>	2	磁力传感器
<code>Sensor.TYPE_ORIENTATION</code>	3	方向传感器
<code>Sensor.TYPE_GYROSCOPE</code>	4	陀螺仪传感器
<code>Sensor.TYPE_LIGHT</code>	5	环境光照传感器

Sensor.TYPE_PRESSURE	6	压力传感器
Sensor.TYPE_TEMPERATURE	7	温度传感器
Sensor.TYPE_PROXIMITY	8	距离传感器

3、从传感器管理器中获取其中某个或者某些传感器的方法有如下三种：

第一种：获取某种传感器的默认传感器

```
Sensor defaultGyroscope = sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
```

第二种：获取某种传感器的列表

```
List<Sensor> pressureSensors = sensorManager.getSensorList(Sensor.TYPE_PRESSURE);
```

第三种：获取所有传感器的列表，我们这个例子就用的第三种

```
List<Sensor> allSensors = sensorManager.getSensorList(Sensor.TYPE_ALL);
```

4、对于某一个传感器，它的一些具体信息的获取方法可以见下表：

方法	描述
<code>getMaximumRange()</code>	最大取值范围
<code>getName()</code>	设备名称
<code>getPower()</code>	功率
<code>getResolution()</code>	精度
<code>getType()</code>	传感器类型
<code>getVendor()</code>	设备供应商
<code>getVersion()</code>	设备版本号

9.92、时间类

```
import java.util.*;
import java.text.SimpleDateFormat;

/**
 * Get date string; Get time string;
 *
 */

public class GetTime {
    /**
```

* 获得日期或时间字符串

```
*
```



```

* @param dateOrTime
* @return
*/
public static String returnDateOrTime(int dateOrTime) {
    if (dateOrTime == 0) { // 返回日期
        SimpleDateFormat sDateFormat = new SimpleDateFormat("yyyy-MM-dd");
        return sDateFormat.format(new java.util.Date());
    } else { // 返回时刻
        Calendar ca = Calendar.getInstance();
        int minute = ca.get(Calendar.MINUTE);
        int hour = ca.get(Calendar.HOUR_OF_DAY);

        return getString(hour) + ":" + getString(minute);
    }
}

/**

```

* num天前的日期

```

*
* @param date
* @param num
* @return
*/
public static String stringToPreDateString(String date, int num) {
    String[] strdate = date.split("-");
    GregorianCalendar dateTime = new GregorianCalendar();
    dateTime.set(Integer.parseInt(strdate[0]),
        Integer.parseInt(strdate[1]) - 1, Integer.parseInt(strdate[2])
        - num);
    Date d = dateTime.getTime();
    SimpleDateFormat sDateFormat = new SimpleDateFormat("yyyy-MM-dd");

    return sDateFormat.format(d);
}

/**

```

* num天后的日期

```

*
* @param date

```



```

* @param num
* @return
*/
public static String stringToNextDateString(String date, int num) {
    String[] strdate = date.split("-");
    GregorianCalendar dateTime = new GregorianCalendar();
    dateTime.set(Integer.parseInt(strdate[0]),
        Integer.parseInt(strdate[1]) - 1, Integer.parseInt(strdate[2])
            + num);
    Date d = dateTime.getTime();
    SimpleDateFormat sDateFormat = new SimpleDateFormat("yyyy-MM-dd");

    return sDateFormat.format(d);
}

/**

```

* 判断 thingdate 的 dotime 天后是否在今天之后

```

*
* @param thingdate
* @param thingTime
* @param dotime
* @return
*/
public static boolean thingInTime(String thingdate, String thingTime,
    String dotime) {
    Calendar ca = Calendar.getInstance();
    int year = ca.get(Calendar.YEAR);
    int month = ca.get(Calendar.MONTH) + 1;
    int day = ca.get(Calendar.DATE);
    int minute = ca.get(Calendar.MINUTE);
    int hour = ca.get(Calendar.HOUR_OF_DAY);

    String thisDateTime = year + "-" + getString(month) + "-"
        + getString(day) + " " + getString(hour) + ":"
        + getString(minute);

    String[] strEndDate = thingdate.split("-");
    GregorianCalendar end_dateTime = new GregorianCalendar();
    end_dateTime.set(Integer.parseInt(strEndDate[0]),
        Integer.parseInt(strEndDate[1]) - 1,
        Integer.parseInt(strEndDate[2]) + Integer.parseInt(dotime));
    Date endDate = end_dateTime.getTime();

```



日期

```

SimpleDateFormat DateFormat = new SimpleDateFormat("yyyy-MM-dd");
String endDateTime2 = DateFormat.format(endDate); // 获得dotime天之后的
日期

String[] strEndDate1 = endDateTime2.split("-");
String[] strEndTime1 = thingTime.split(":");

String endDateTime = strEndDate1[0] + "-"
    + getString(Integer.parseInt(strEndDate1[1])) + "-"
    + getString(Integer.parseInt(strEndDate1[2])) + " "
    + getString(Integer.parseInt(strEndTime1[0])) + ":"
    + getString(Integer.parseInt(strEndTime1[1]));

boolean flag = false;
if (covertTimeToLong(thisDateTime) < covertTimeToLong(endDateTime)) {
    flag = true;
} else if (covertTimeToLong(thisDateTime) >=
covertTimeToLong(endDateTime)) {
    flag = false;
}

return flag;
}

/**

```

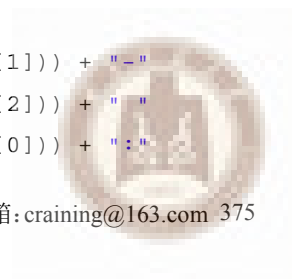
* 判断 testDate+testTime 是否在两个时间之内

```

*
* @param first
* @param testDate
* @param testTime
* @param last
* @return 是否
*/
public static boolean betweenTwoDate(String first, String testDate,
    String testTime, String last) {
    String[] testDateArray = testDate.split("-");
    String[] testTimeArray = testTime.split(":");

    String testDateTime = testDateArray[0] + "-"
        + getString(Integer.parseInt(testDateArray[1])) + "-"
        + getString(Integer.parseInt(testDateArray[2])) + " "
        + getString(Integer.parseInt(testTimeArray[0])) + ":"

```



```

        + getString(Integer.parseInt(testTimeArray[1]));

String[] firstDateArray = first.split("-");
String[] lastDateArray = last.split("-");

first = firstDateArray[0] + "-"
        + getString(Integer.parseInt(firstDateArray[1])) + "-"
        + getString(Integer.parseInt(firstDateArray[2])) + " "
        + returnDateOrTime(1);
last = lastDateArray[0] + "-"
        + getString(Integer.parseInt(lastDateArray[1])) + "-"
        + getString(Integer.parseInt(lastDateArray[2])) + " "
        + "23:59";

if (covertTimeToLong(first) <= covertTimeToLong(testDateTime)
    && covertTimeToLong(testDateTime) <= covertTimeToLong(last)) {
    return true;
} else {
    return false;
}
}

/**
 * 日期字符串返回年月日
 *
 * @param date
 * @param one
 * @return
 */
public static String stringToOneString(String date, String one) {
    String[] strdate = date.split("-");
    if (one.equals("year")) {
        return strdate[0];
    } else if (one.equals("month")) {
        return strdate[1];
    } else {
        return strdate[2];
    }
}

/**
 * 时间字符串返回时分
 *
 * @param time
 * @param one

```




```
* @return
*/
public static String stringToTimeString(String time, String one) {
    String[] strdate = time.split(":");
    if (one.equals("hour")) {
        return strdate[0];
    } else {
        return strdate[1];
    }
}

/**
 * 将日期时间字符串转化为 long 型, 用于比较大小
 *
 * @param time
 * @return
 */
public static long covertTimeToLong(String time) {
    long ll = 0L;
    int yy = 0;
    int mm = 0;
    int dd = 0;
    int hh = 0;
    int mi = 0;
    if (time != null && !"".equals(time)) {
        yy = Integer.parseInt(time.substring(0, 4));
        mm = Integer.parseInt(time.substring(5, time.lastIndexOf("-")));
        dd = Integer.parseInt(time.substring(time.lastIndexOf("-") + 1,
            time.length() - 6));
        hh = Integer.parseInt(time.substring(time.length() - 5,
            time.indexOf(":")));
        mi = Integer.parseInt(time.substring(time.length() - 2));
        GregorianCalendar gc = new GregorianCalendar(yy, mm, dd, hh, mi);
        Date d = gc.getTime();
        ll = d.getTime();
    } else {
        ll = Long.MAX_VALUE;
    }

    return ll;
}

/**
 * 将时间转为特定的格式 如: 1 转为 01
```



```

*
* @param mmm
* @return
*/
public static String getString(int mmm) {
    if (mmm < 10) {
        if (mmm == 0) {
            return "00";
        } else {
            return "0" + mmm;
        }
    } else {
        return "" + mmm;
    }
}
}

```

附录:

附录 1、Xml布局中的常用属性

android:background="@android:color/transparent" 控件透明和半透明

1. 通用属性

首先按照程序的目录结构大致分析:

res/layout/ 这个目录存放的就是布局用的 xml 文件, 一般默认为 main.xml

res/values/ 这个目录存放的是一堆常量的 xml 文件

res/drawable/ 存放的是一些图片什么的, 当然图标也在这里

下面主要对 layout 下的 xml 文件做个介绍, 顺便也把布局的方法总结一下:

- 文件的开头

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

这是在说明 xml 版本及字符编码

- 紧接着到了关键的部分:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/ android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
</LinearLayout>
```

其中开头的“LinearLayout”是布局的方式, 可以有很多种, 最常用的应该就是 Linear



了，其他的布局方法等下在后面总结。

接着 `android:layout_width(height)="wrap_content"` 是在设置这部分的宽高，也可以是绝对值，当然设置为绝对值时要标上单位。

- 在 `<LinearLayout ...>` 和 `</LinearLayout>` 之间可以添加控件了，比如要添加一个名字为 `btn` 的 `Button` 控件，并且 `Button` 上显示的文字是 `"Test!"`，可以这样写：

```
<Button id="@+id/btn"
    android:text="Test!"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

开头 `id` 后面的就是控件名称，在用于添加事件 `Listener` 时会用到，而下几行的 `android:xxx` 就是设置控件的属性了，这些属性在 `Android` 的文档中都有，不需要特别去记，一般现查就可以了。

- 有一点要说明的是，布局方法可以嵌套，有点像 `java` 中的 `Container`，可以非常方便的把界面“堆”出来。

布局方式的简单说明：

查了 `Android` 文档发现布局确实很多，只列出两个我自己认为较常用的：

- `LinearLayout` 线性的布局方式，要么上下，要么左右的添加控件，很常用；
- `GridView` 中文翻译过来是网格布局，控件按照顺序依次填到每个格子里就好了，出来的界面会很整齐，较常用；

具体的几个布局如下：

@<1> `LinearLayout` (线性布局) 提供了控件水平垂直排列的模型，同时可以通过设置子控件的 `weight` 布局参数控制各个控件在布局中的相对大小。水平 (`vertical`) 垂直 (`horizontal`)

`fill-parent`: 占满整个屏幕, `wrap-content`: 刚好适合控件内容的大小

对齐方式 `gravity` 取值:

`top`: 不改变大小, 位置置于容器的顶部

`bottom`: 不改变大小, 位置置于容器的底部

`left`: 不改变大小, 位置置于容器的左边

`right`: 不改变大小, 位置置于容器的右边

`center_vertical`: 不改变大小, 位置置于容器的纵向中央部分

`center_horizontal`: 不改变大小, 位置置于容器的横向中央部分

`center`: 不改变大小, 位置置于容器的横向和纵向的中央部分

`fill_vertical`: 可能的话, 纵向延伸可以填满容器

`fiil_horizontal`: 可能的话, 横向延伸可以填满容器

`fiil`: 可能的话, 纵向和横向延伸填满容器

@<2> `AbsoluteLayout` (坐标布局) 可以让子元素指定准确的 `x/y` 坐标值, 并显示在屏幕上。(0, 0) 为左上角, 当向下或向右移动时, 坐标值将变大。`AbsoluteLayout` 没有页边框, 允许元素之间互相重叠 (尽管不推荐)。我们通常不推荐使用 `AbsoluteLayout`, 除非你有正当理由要使用它, 因为它使界面代码太过刚性, 以至于在不同的设备上可能不能很好地工作。

`Android: layout_x/layout_y=" 56px"` 确定控件位置

@<3> `RelativeLayout` (相对布局) 允许子元素指定他们相对于其它元素或父元素的位置 (通过 `ID` 指定)。因此, 你可以以右对齐, 或上下, 或置于屏幕中央的形式来排列两个元素。元素按顺序排列, 因此如果第一个元素在屏幕的中央, 那么相对于这个元素的其它元

素将以屏幕中央的相对位置来排列。如果使用 XML 来指定这个 layout，在你定义它之前，被关联的元素必须定义。

Android: layout_centerInparent, 将当前控件放置于起父控件的横向和纵向的中央部分

Android: layout_centerHorizontal, 使当前控件置于父控件横向的中央部分

Android: layout_centerVertical, 使当前控件置于父控件纵向的中央部分

Android: layout_alignParentLeft, 使当前控件的左端和父控件左端对齐

Android: layout_alignParentRight, 使当前控件的右端和父控件右端对齐

Android: layout_alignParentTop, 使当前控件的顶端和父控件顶端对齐

Android: layout_alignParentBottom, 使当前控件的底端和父控件底端对齐

上述属性只能设置 Bool 类型的值, “true” 或 “false”

Android: layout_below/layout_above/ layout_toLeftOf/ layout_toRightOf
= “@id/” 使当前控件置于给出 id 的空间的下方/上方/左边/右边

Android: layout_marginBottom/layout_marginLeft/layout_marginRight/layout_marginTop=” 30px” 使当前控件底部/左边/右边/顶部空出相应像素空间

④<4> FrameLayout(单帧布局)是最简单的一个布局对象。它被定制为你屏幕上的一个空白备用区域，之后你可以在其中填充一个单一对象——比如，一张你要发布的图片。所有的子元素将会固定在屏幕的左上角；你不能为FrameLayout中的一个子元素指定一个位置。后一个子元素将会直接在前一个子元素之上进行覆盖填充，把它们部份或全部挡住（除非后一个子元素是透明的）。

Android:src=http://blog.soso.com/qz.q/” @drawable/” 属性指定所需图片的文件位置，用 ImageView 显示图片时，也应当用 android: src 指定要显示的图片

④<5> TableLayout（表格布局）以行列的形式管理子控件，每一行为一个 TableRow 的对象，TableRow 也可以添加子控件

android: collapseColumns= “n” 隐藏 TableLayout 里面的 TableRow 的列 n

android: stretchColumns= “n” 设置列 n 为可延伸的列

android: shrinkColumns= “n” 设置列 n 为可收缩的列

Android:src=http://blog.soso.com/qz.q/” @drawable/” 属性指定所需图片的文件位置，用 ImageView 显示图片时，也应当用 android: src 指定要显示的图片。



2. Edit Text部分属性

比较乱, 由于从不同地方搜集的, 有重复的, 慢慢看~~~

EditText继承关系: View-->TextView-->EditText

EditText的属性很多, 这里介绍几个:

android:hint="请输入数字! "//设置显示在空间上的提示信息

android:numeric="integer"//设置只能输入整数, 如果是小数则是: decimal

android:singleLine="true"//设置单行输入, 一旦设置为true, 则文字不会自动换行。

android:password="true"//设置只能输入密码

android:textColor = "#ff8c00"//字体颜色

android:textStyle="bold"//字体, bold, italic, bolditalic

android:textSize="20dip"//大小

android:capitalize = "characters"//以大写字母写

android:textAlign="center"//EditText没有这个属性, 但TextView有, 居中

android:textColorHighlight="#cccccc"//被选中文字的底色, 默认为蓝色

android:textColorHint="#ffff00"//设置提示信息文字的颜色, 默认为灰色

android:textScaleX="1.5"//控制字与字之间的间距

android:typeface="monospace"//字型, normal, sans, serif, monospace

android:background="@null"//背景, 这里没有, 指透明

android:layout_weight="1"//权重, 控制控件之间的地位, 在控制控件显示的大小时蛮有用的。

android:textAppearance="?android:attr/textAppearanceLargeInverse"//文字外观

android:layout_gravity="center_vertical"//设置控件显示的位置: 默认top, 这里居中显示, 还有bottom

android:gravity="top" //多行中指针在第一行第一位置

et.setSelection(et.length());//调整光标到最后一行

android:autoText //自动拼写帮助

android:capitalize //首字母大写

android:digits //设置只接受某些数字

Android: singleLine//是否单行或者多行, 回车是离开文本框还是文本框增加新行

android: numeric //只接受数字

android: phoneNumber //输入电话号码

android: editable //是否可编辑

android:autoLink="all" //设置文本超链接样式当点击网址时, 跳向该网址

android:password="true"//设置只能输入密码

android:textColor = "#ff8c00"//字体颜色

android:textStyle="bold"//字体, bold, italic, bolditalic

android:textSize="20dip"//大小

android:capitalize = "characters"//以大写字母写

android:textAlign="center"//EditText 没有这个属性, 但 TextView 有

android:textColorHighlight="#cccccc"//被选中文字的底色, 默认为蓝色

android:textColorHint="#ffff00"//设置提示信息文字的颜色, 默认为灰色

android:textScaleX="1.5"//控制字与字之间的间距

android:lineSpacingExtra="3dip " //设置行间距

android:typeface="monospace"//字型, normal, sans, serif, monospace



`android:background="@null"` // 空间背景，这里没有，指透明

`android:layout_weight="1"` // 权重 在控制控件显示的大小时蛮有用的。

`android:textAppearance="?android:attr/textAppearanceLargeInverse"` // 文字外观，这里引用的是系统自带的一个外观，? 表示系统是否有这种外观，否则使用默认的外观。不知道这样理解对不对？

属性名称描述

`android:autoLink` 设置是否当文本为 URL 链接/email/电话号码/map 时，文本显示为可点击的链接。可选值(none/web/email/phone/map/all)

`android:autoText` 如果设置，将自动执行输入值的拼写纠正。此处无效果，在显示输入法并输入的时候起作用。

`android:bufferType` 指定 `getText()` 方式取得的文本类别。选项 `editable` 类似于 `StringBuilder` 可追加字符，

也就是说 `getText` 后可调用 `append` 方法设置文本内容。`spannable` 则可在给定的字符区域使用样式，参见这里 1、这里 2。

`android:capitalize` 设置英文字母大写类型。此处无效果，需要弹出输入法才能看得到，参见 `EditText` 此属性说明。

`android:cursorVisible` 设定光标为显示/隐藏，默认显示。

`android:digits` 设置允许输入哪些字符。如“1234567890.+-% ()”

`android:drawableBottom` 在 `text` 的下方输出一个 `drawable`，如图片。如果指定一个颜色的话会把 `text` 的背景设为该颜色，并且同时和 `background` 使用时覆盖后者。

`android:drawableLeft` 在 `text` 的左边输出一个 `drawable`，如图片。

`android:drawablePadding` 设置 `text` 与 `drawable`(图片)的间隔，与 `drawableLeft`、`drawableRight`、`drawableTop`、`drawableBottom` 一起使用，可设置为负数，单独使用没有效果。

`android:drawableRight` 在 `text` 的右边输出一个 `drawable`，如图片。

`android:drawableTop` 在 `text` 的正上方输出一个 `drawable`，如图片。

`android:editable` 设置是否可编辑。这里无效果，参见 `EditText`。

`android:editorExtras` 设置文本的额外的输入数据。在 `EditText` 再讨论。

`android:ellipsize` 设置当文字过长时,该控件该如何显示。有如下值设置:”start”——省略号显示在开头;”end”——省略号显示在结尾;”middle”——省略号显示在中间;”marquee”——以跑马灯的方式显示(动画横向移动)

`android:freezesText` 设置保存文本的内容以及光标的位置。参见：这里。

`android:gravity` 设置文本位置，如设置成“center”，文本将居中显示。

`android:hintText` 为空时显示的文字提示信息，可通过 `textColorHint` 设置提示信息的颜色。此属性在 `EditText` 中使用，但是这里也可以用。

`android:imeOptions` 附加功能，设置右下角 IME 动作与编辑框相关的动作，如 `actionDone` 右下角将显示一个“完成”，而不设置默认是一个回车符号。这个在 `EditText` 中再详细说明，此处无用。

`android:imeActionId` 设置 IME 动作 ID。在 `EditText` 再做说明，可以先看这篇帖子：这里。

`android:imeActionLabel` 设置 IME 动作标签。在 `EditText` 再做说明。

`android:includeFontPadding` 设置文本是否包含顶部和底部额外空白，默认为 `true`。

`android:inputMethod` 为文本指定输入法，需要完全限定名(完整的包名)。例如：`com.google.android.inputmethod.pinyin`，但是这里报错找不到。

`android:inputType` 设置文本的类型，用于帮助输入法显示合适的键盘类型。在 `EditText`

中再详细说明，这里无效果。

`android:linksClickable` 设置链接是否点击连接，即使设置了 `autoLink`。

`android:marqueeRepeatLimit` 在 `ellipsize` 指定 `marquee` 的情况下，设置重复滚动的次数，当设置为 `marquee_forever` 时表示无限次。

`android:ems` 设置 `TextView` 的宽度为 N 个字符的宽度。这里测试为一个汉字字符宽度，如图：

`android:maxEms` 设置 `TextView` 的宽度为最长为 N 个字符的宽度。与 `ems` 同时使用时覆盖 `ems` 选项。

`android:minEms` 设置 `TextView` 的宽度为最短为 N 个字符的宽度。与 `ems` 同时使用时覆盖 `ems` 选项。

`android:maxLength` 限制显示的文本长度，超出部分不显示。

`android:lines` 设置文本的行数，设置两行就显示两行，即使第二行没有数据。

`android:maxLines` 设置文本的最大显示行数，与 `width` 或者 `layout_width` 结合使用，超出部分自动换行，超出行数将不显示。

`android:minLines` 设置文本的最小行数，与 `lines` 类似。

`android:lineSpacingExtra` 设置行间距。

`android:lineSpacingMultiplier` 设置行间距的倍数。如“1.2”

`android:numeric` 如果被设置，该 `TextView` 有一个数字输入法。此处无用，设置后唯一效果是 `TextView` 有点击效果，此属性在 `EditText` 将详细说明。

`android:password` 以小点“.”显示文本

`android:phoneNumber` 设置为电话号码的输入方式。

`android:privateImeOptions` 设置输入法选项，此处无用，在 `EditText` 将进一步讨论。

`android:scrollHorizontally` 设置文本超出 `TextView` 的宽度的情况下，是否出现横拉条。

`android:selectAllOnFocus` 如果文本是可选择的，让他获取焦点而不是将光标移动为文本的开始位置或者末尾位置。`TextView` 中设置后无效果。

`android:shadowColor` 指定文本阴影的颜色，需要与 `shadowRadius` 一起使用。效果：

`android:shadowDx` 设置阴影横向坐标开始位置。

`android:shadowDy` 设置阴影纵向坐标开始位置。

`android:shadowRadius` 设置阴影的半径。设置为 0.1 就变成字体的颜色了，一般设置为 3.0 的效果比较好。

`android:singleLine` 设置单行显示。如果和 `layout_width` 一起使用，当文本不能全部显示时，后面用“...”来表示。如 `android:text="test_ singleLine "` `android:singleLine="true"` `android:layout_width="20dp"` 将只显示“t...”。如果不设置 `singleLine` 或者设置为 `false`，文本将自动换行

`android:shadowDx` 设置阴影横向坐标开始位置。

`android:shadowDy` 设置阴影纵向坐标开始位置。

`android:shadowRadius` 设置阴影的半径。设置为 0.1 就变成字体的颜色了，一般设置为 3.0 的效果比较好。

`android:singleLine` 设置单行显示。如果和 `layout_width` 一起使用，当文本不能全部显示时，后面用“...”来表示。如 `android:text="test_ singleLine "` `android:singleLine="true"` `android:layout_width="20dp"` 将只显示“t...”。如果不设置 `singleLine` 或者设置为 `false`，文本将自动换行

`android:text` 设置显示文本。

`android:textSize` 设置文字大小，推荐度量单位“sp”，如“15sp”



`android:textStyle` 设置字形[**bold**(粗体) 0, **italic**(斜体) 1, **bolditalic**(又粗又斜) 2] 可以设置一个或多个, 用“|”隔开

`android:typeface` 设置文本字体, 必须是以下常量值之一: `normal` 0, `sans` 1, `serif` 2, `monospace`(等宽字体) 3]

`android:height` 设置文本区域的高度, 支持度量单位: `px`(像素)/`dp`/`sp`/`in`/`mm`(毫米)

`android:maxHeight` 设置文本区域的最大高度

`android:minHeight` 设置文本区域的最小高度

`android:width` 设置文本区域的宽度, 支持度量单位: `px`(像素)/`dp`/`sp`/`in`/`mm`(毫米), 与 `layout_width` 的区别看这里。

`android:maxLength` 设置文本区域的最大宽度

`android:minWidth` 设置文本区域的最小宽度

`android:textAppearance` 设置文字外观。如“`?android:attr/textAppearanceLargeInverse`

”这里引用的是系统自带的一个外观, ?表示系统是否有这种外观, 否则使用默认的外观。

可 设 置 的 值 如 下 :
`textAppearanceButton/textAppearanceInverse/textAppearanceLarge/textAppearanceLargeInverse/textAppearanceMedium/textAppearanceMediumInverse/textAppearanceSmall/textAppearanceSmallInverse`

`android:textAppearance` 设置文字外观。如“`?android:attr/textAppearanceLargeInverse`

”这里引用的是系统自带的一个外观, ?表示系统是否有这种外观, 否则使用默认的外观。可 设 置 的 值 如 下 :

`textAppearanceButton/textAppearanceInverse/textAppearanceLarge/textAppearanceLargeInverse/textAppearanceMedium/textAppearanceMediumInverse`

3. `layout_alignParentRight` `android:paddingRight`

`android:layout_alignParentRight="true"`

使当前控件的右端和父控件的右端对齐。这里属性值只能为 `true` 或 `false`, 默认 `false`。

`android:layout_marginLeft="10dp"`

使当前控件左边空出相应的空间。

`android:layout_toLeftOf="@id/ok"`

使当前控件置于 `id` 为 `ok` 的控件的左边。

`android:layout_alignTop="@id/ok"`

使当前控件与 `id` 控件的上端对齐。

`padding` 表示填充, `margin` 表示边距

可通过 `android:padding` 属性进行设置, 4 个方向的边距属性为 `android:paddingLeft`, `android:paddingRight`, `android:paddingTop`, and `android:paddingBottom`.

结论:

*`android:layout_marginBottom`

*`android:layout_marginLeft`




```

*android:layout_marginRight
*android:layout_marginTop
上面几个属性的值是根据下面的相对位置的对象的值来做计算的,如果没有相对的对象就以
总体布局来计算
*android:layout_below
*android:layout_above
*android:layout_toLeftOf
*android:layout_toRightOf
*android:layout_alignTop

*android:layout_centerHorizontal //是否支持横屏或竖屏
*android:layout_centerVertical //这个根据单词的意思: 中
心垂直
*android:layout_centerInparent //
android:layout_centerInParent="true"//居中在父对象
android:layout_centerInParent="false"... 浏览器不支持多窗口显示,意思就是说所有
页面在单一窗口打开,这样避免了页面布局控制显示问题
下面的相对于父的相对位置
*android:layout_alignParentBottom
*android:layout_alignParentLeft
*android:layout_alignParentRight
*android:layout_alignParentTop
*android:layout_alignWithParentIfMissing

```

附录 2、intent action

```

android.intent.action.ALL_APPS
android.intent.action.ANSWER
android.intent.action.ATTACH_DATA
android.intent.action.BUG_REPORT
android.intent.action.CALL
android.intent.action.CALL_BUTTON
android.intent.action.CHOOSER
android.intent.action.CREATE_LIVE_FOLDER
android.intent.action.CREATE_SHORTCUT
android.intent.action.DELETE
android.intent.action.DIAL
android.intent.action.EDIT
android.intent.action.GET_CONTENT
android.intent.action.INSERT
android.intent.action.INSERT_OR_EDIT

```



android.intent.action.MAIN
android.intent.action.MEDIA_SEARCH
android.intent.action.PICK
android.intent.action.PICK_ACTIVITY
android.intent.action.RINGTONE_PICKER
android.intent.action.RUN
android.intent.action.SEARCH
android.intent.action.SEARCH_LONG_PRESS
android.intent.action.SEND
android.intent.action.SENDTO
android.intent.action.SET_WALLPAPER
android.intent.action.SYNC
android.intent.action.SYSTEM_TUTORIAL
android.intent.action.VIEW
android.intent.action.VOICE_COMMAND
android.intent.action.WEB_SEARCH
android.net.wifi.PICK_WIFI_NETWORK
android.settings.AIRPLANE_MODE_SETTINGS
android.settings.APN_SETTINGS
android.settings.APPLICATION_DEVELOPMENT_SETTINGS
android.settings.APPLICATION_SETTINGS
android.settings.BLUETOOTH_SETTINGS
android.settings.DATA_ROAMING_SETTINGS
android.settings.DATE_SETTINGS
android.settings.DISPLAY_SETTINGS
android.settings.INPUT_METHOD_SETTINGS
android.settings.INTERNAL_STORAGE_SETTINGS
android.settings.LOCALE_SETTINGS
android.settings.LOCATION_SOURCE_SETTINGS
android.settings.MANAGE_APPLICATIONS_SETTINGS
android.settings.MEMORY_CARD_SETTINGS
android.settings.NETWORK_OPERATOR_SETTINGS
android.settings.QUICK_LAUNCH_SETTINGS
android.settings.SECURITY_SETTINGS
android.settings.SETTINGS
android.settings.SOUND_SETTINGS
android.settings.SYNC_SETTINGS
android.settings.USER_DICTIONARY_SETTINGS
android.settings.WIFI_IP_SETTINGS
android.settings.WIFI_SETTINGS
android.settings.WIRELESS_SETTINGS



附录 3、Android 的动作、广播、类别等标志

String BATTERY_CHANGED_ACTION 广播：充电状态，或者电池的电量发生变化
"android.intent.action.BATTERY_CHANGED"

String BOOT_COMPLETED_ACTION 广播：在系统启动后，这个动作被广播一次(只有一次)
"android.intent.action.BOOT_COMPLETED"

String CALL_FORWARDING_STATE_CHANGED_ACTION 广播：语音电话的呼叫转移状态已经改变 "android.intent.action.CFF"

String CONFIGURATION_CHANGED_ACTION 广播：设备的配置信息已经改变，参见 Resources.Configuration. "android.intent.action.CONFIGURATION_CHANGED" Creator CREATOR 无 无

String DATA_ACTIVITY_STATE_CHANGED_ACTION 广播：电话的数据活动(data activity)状态(即收发数据的状态)已经改变。"android.intent.action.DATA_ACTIVITY"

String DATA_CONNECTION_STATE_CHANGED_ACTION 广播：电话的数据连接状态已经改变 "android.intent.action.DATA_STATE"

String DATE_CHANGED_ACTION 广播：日期被改变
"android.intent.action.DATE_CHANGED"

String FOTA_CANCEL_ACTION 广播：取消所有被挂起的 (pending) 更新下载
"android.server.checkin.FOTA_CANCEL"

String FOTA_INSTALL_ACTION 广播：更新已经被确认，马上就要开始安装
"android.server.checkin.FOTA_INSTALL"

String FOTA_READY_ACTION 广播：更新已经被下载，可以开始安装
"android.server.checkin.FOTA_READY"

String FOTA_RESTART_ACTION 广播：恢复已经停止的更新下载
"android.server.checkin.FOTA_RESTART"

String FOTA_UPDATE_ACTION 广播：通过 OTA 下载并安装操作系统更新
"android.server.checkin.FOTA_UPDATE"

String MEDIABUTTON_ACTION 广播：用户按下了“Media Button”
"android.intent.action.MEDIABUTTON"

String MEDIA_BAD_REMOVAL_ACTION 广播：扩展介质(扩展卡)已经从 SD 卡插槽拔



出，但是挂载点 (mount point) 还没解除 (unmount)
"android.intent.action.MEDIA_BAD_REMOVAL"

String MEDIA_EJECT_ACTION 广播：用户想要移除扩展介质(拔掉扩展卡)
"android.intent.action.MEDIA_EJECT"

String MEDIA_MOUNTED_ACTION 广播：扩展介质被插入，而且已经被挂载
"android.intent.action.MEDIA_MOUNTED"

String MEDIA_REMOVED_ACTION 广播：扩展介质被移除。
"android.intent.action.MEDIA_REMOVED"

String MEDIA_SCANNER_FINISHED_ACTION 广播：已经扫描完介质的一个目录
"android.intent.action.MEDIA_SCANNER_FINISHED"

String MEDIA_SCANNER_STARTED_ACTION 广播：开始扫描介质的一个目录
"android.intent.action.MEDIA_SCANNER_STARTED"

String MEDIA_SHARED_ACTION 广播：扩展介质的挂载被解除 (unmount)，因为它已经作为 USB 大容量存储被共享 "android.intent.action.MEDIA_SHARED"

String MEDIA_UNMOUNTED_ACTION 广播：扩展介质存在，但是还没有被挂载 (mount)
"android.intent.action.MEDIA_UNMOUNTED"

String MESSAGE_WAITING_STATE_CHANGED_ACTION 广播：电话的消息等待(语音邮件)状态已经改变 "android.intent.action.MWI"

String TIMEZONE_CHANGED_ACTION 广播：时区已经改变
"android.intent.action.TIMEZONE_CHANGED"

String TIME_CHANGED_ACTION 广播：时间已经改变(重新设置)
"android.intent.action.TIME_SET"

String TIME_TICK_ACTION 广播：当前时间已经变化(正常的时间流逝)
"android.intent.action.TIME_TICK"

String UMS_CONNECTED_ACTION 广播：设备进入 USB 大容量存储模式
"android.intent.action.UMS_CONNECTED"

String UMS_DISCONNECTED_ACTION 广播：设备从 USB 大容量存储模式退出
"android.intent.action.UMS_DISCONNECTED"

String WALLPAPER_CHANGED_ACTION 广播：系统的墙纸已经改变
"android.intent.action.WALLPAPER_CHANGED"



String XMPP_CONNECTED_ACTION 广播：XMPP 连接已经被建立
"android.intent.action.XMPP_CONNECTED"

String XMPP_DISCONNECTED_ACTION 广播：XMPP 连接已经被断开
"android.intent.action.XMPP_DI"

String NETWORK_TICKLE_RECEIVED_ACTION 广播：设备收到了新的网络 "tickle" 通知
"android.intent.action.NETWORK_TICKLE_RECEIVED"

String PACKAGE_ADDED_ACTION 广播：设备上新安装了一个应用程序包
"android.intent.action.PACKAGE_ADDED"

String PACKAGE_REMOVED_ACTION 广播：设备上删除了一个应用程序包
"android.intent.action.PACKAGE_REMOVED"

String PHONE_STATE_CHANGED_ACTION 广播：电话状态已经改变
"android.intent.action.PHONE_STATE"

String PROVIDER_CHANGED_ACTION 广播：更新将要（真正）被安装
"android.intent.action.PROVIDER_CHANGED"

String PROVISIONING_CHECK_ACTION 广播：要求 polling of provisioning service 下载最新的设置
"android.intent.action.PROVISIONING_CHECK"

String SCREEN_OFF_ACTION 广播：屏幕被关闭 "android.intent.action.SCREEN_OFF"

String SCREEN_ON_ACTION 广播：屏幕已经被打开 "android.intent.action.SCREEN_ON"

String SERVICE_STATE_CHANGED_ACTION 广播：电话服务的状态已经改变
"android.intent.action.SERVICE_STATE"

String SIGNAL_STRENGTH_CHANGED_ACTION 广播：电话的信号强度已经改变
"android.intent.action.SIG_STR"

String STATISTICS_REPORT_ACTION 广播：要求 receivers 报告自己的统计信息
"android.intent.action.STATISTICS_REPORT"

String STATISTICS_STATE_CHANGED_ACTION 广播：统计信息服务的状态已经改变
"android.intent.action.STATISTICS_STATE_CHANGED"

String BROWSABLE_CATEGORY 类别：能够被浏览器安全使用的 activities 必须支持这个类别
"android.intent.category.BROWSABLE"



String ALTERNATIVE_CATEGORY 类别: 说明 activity 是用户正在浏览的数据的一个可选操作 "android.intent.category.ALTERNATIVE"

String DEFAULT_CATEGORY 类别: 如果 activity 是对数据执行确省动作(点击, center press)的一个选项, 需要设置这个类别。 "android.intent.category.DEFAULT"

String FRAMEWORK_INSTRUMENTATION_TEST_CATEGORY 类别: To be used as code under test for framework instrumentation tests. "android.intent.category.FRAMEWORK_INSTRUMENTATION_TEST"

String GADGET_CATEGORY 类别: 这个 activity 可以被嵌入宿主 activity (activity that is hosting gadgets)。 "android.intent.category.GADGET"

String HOME_CATEGORY 类别: 主屏幕 (activity), 设备启动后显示的第一个 activity "android.intent.category.HOME"

String LAUNCHER_CATEGORY 类别: Activity 应该被显示在顶级的 launcher 中 "android.intent.category.LAUNCHER"

String PREFERENCE_CATEGORY 类别: activity 是一个设置面板 (preference panel) "android.intent.category.PREFERENCE"

String SAMPLE_CODE_CATEGORY 类别: To be used as an sample code example (not part of the normal user experience). "android.intent.category.SAMPLE_CODE"

String SELECTED_ALTERNATIVE_CATEGORY 类别: 对于被用户选中的数据, activity 是它的一个可选操作 "android.intent.category.SELECTED_ALTERNATIVE"

String TAB_CATEGORY 类别: 这个 activity 应该在 TabActivity 中作为一个 tab 使用 "android.intent.category.TAB"

String TEST_CATEGORY 类别: 作为测试目的使用, 不是正常的用户体验的一部分 "android.intent.category.TEST"

String UNIT_TEST_CATEGORY 类别: 应该被用作单元测试(通过 test harness 运行) "android.intent.category.UNIT_TEST"

String WALLPAPER_CATEGORY 类别: 这个 activity 能过为设备设置墙纸 "android.intent.category.WALLPAPER"

String DEVELOPMENT_PREFERENCE_CATEGORY 类别: 说明 activity 是一个设置面板 (development preference panel). "android.intent.category.DEVELOPMENT_PREFERENCE"

String EMBED_CATEGORY 类别: 能够在上级(父)activity 中运行

"android.intent.category.EMBED"

String DELETE_ACTION 动作：从容器中删除给定的数据 "android.intent.action.DELETE"

String DEFAULT_ACTION 动作：和 VIEW_ACTION 相同，是在数据上执行的标准动作
"android.intent.action.VIEW"

String DIAL_ACTION 动作：拨打数据中指定的电话号码 "android.intent.action.DIAL"

String EDIT_ACTION 动作：为制定的数据显示可编辑界面 "android.intent.action.EDIT"

String EMERGENCY_DIAL_ACTION 动作：拨打紧急电话号码
"android.intent.action.EMERGENCY_DIAL"

String GET_CONTENT_ACTION 动作：让用户选择数据并返回。
"android.intent.action.GET_CONTENT"

String INSERT_ACTION 动作：在容器中插入一个空项 (item)。
"android.intent.action.INSERT"

String LOGIN_ACTION 动作：获取登录凭证。 "android.intent.action.LOGIN"

String MAIN_ACTION 动作：作为主入口点启动，不需要数据 "android.intent.action.MAIN"

String SETTINGS_ACTION 动作：显示系统设置。输入：无 "android.intent.action.SETTINGS"

String SYNC_ACTION 动作：执行数据同步 "android.intent.action.SYNC"

String WALLPAPER_SETTINGS_ACTION 动作：显示选择墙纸的设置界面。输入：无
"android.intent.action.WALLPAPER_SETTINGS"

String WEB_SEARCH_ACTION 动作：执行 web 搜索
"android.intent.action.WEB_SEARCH"

String VIEW_ACTION 动作：向用户显示数据 "android.intent.action.VIEW"

String SENDTO_ACTION 动作：向 data 指定的接收者发送一个消息
"android.intent.action.SENDTO"

String RUN_ACTION 动作：运行数据(指定的应用)，无论它(应用)是什么
"android.intent.action.RUN"

String PICK_ACTION 动作：从数据中选择一个项目 (item)，将被选中的项目返回
"android.intent.action.PICK"



String PICK_ACTIVITY_ACTION 动作：选择一个 activity，返回被选择的 activity 的类(名)
"android.intent.action.PICK_ACTIVITY"

String ADD_SHORTCUT_ACTION 动作：在系统中添加一个快捷方式
"android.intent.action.ADD_SHORTCUT"

String ALL_APPS_ACTION 动作：列举所有可用的应用 "android.intent.action.ALL_APPS"

String ANSWER_ACTION 动作：处理拨入的电话 "android.intent.action.ANSWER"

String BUG_REPORT_ACTION 动作：显示 activity 报告错误
"android.intent.action.BUG_REPORT"

String CALL_ACTION 动作：拨打电话，被呼叫的联系人在数据中指定
"android.intent.action.CALL"

String CLEAR_CREDENTIALS_ACTION 动作：清除登陆凭证 (credential)
"android.intent.action.CLEAR_CREDENTIALS"

String TEMPLATE_EXTRA 附加数据：新记录的初始化模板
"android.intent.extra.TEMPLATE"

String INTENT_EXTRA 附加数据：和 PICK_ACTIVITY_ACTION 一起使用时，说明用户选择的用来显示的 activity 和 ADD_SHORTCUT_ACTION 一起使用的时候，描述要添加的快捷方式 2010-08-19

"android.intent.extra.INTENT"

String LABEL_EXTRA 附加数据：大写字母开头的字符标签，和 ADD_SHORTCUT_ACTION 一起使用
"android.intent.extra.LABEL"

int MULTIPLE_TASK_LAUNCH 启动标记：和 NEW_TASK_LAUNCH 联合使用，禁止将已有的任务改变为前景任务 (foreground)。 8 0x00000008

int NEW_TASK_LAUNCH 启动标记：设置以后，activity 将成为历史堆栈中的第一个新任务(栈顶) 4 0x00000004

int NO_HISTORY_LAUNCH 启动标记：设置以后，新的 activity 不会被保存在历史堆栈中 1 0x00000001

int SINGLE_TOP_LAUNCH 启动标记：设置以后，如果 activity 已经启动，而且位于历史堆栈的顶端，将不再启动(不重新启动) activity 2 0x00000002



int FORWARD_RESULT_LAUNCH 启动标记：如果这个标记被设置，而且被一个已经存在的 activity 用来启动新的 activity，已有 activity 的回复目标 (reply target) 会被转移给新的 activity。 16
0x00000010

★★★附带工具包说明

1. APK反编译工具.rar

-----对 apk 进行反编译，得比较精确的 java 源码

2. APK安装工具.rar

-----实现双击 apk 就能安装到开启的模拟器中

两个工具都是 Windows 下的工具！

注：

如果你没有在附件中找到它们，但又需要的话可以到此处下载：

<http://www.cmd100.com/bbs/forum-viewthread-tid-9195-fromuid-1714.html>

（可能需要先注册成为 cmd100 论坛 的用户才能下载）

至此本文档基本结束，更多的功能就只能靠您自己去发现啦！Android 开发很给力，坚持下去，您一定会成为真正的 android 开发高手！有机会的话尝试底层开发哦！将来做出我们国人自己的智能手机（平板）系统，走向国际！

