OPHONE平台

开发入门指南

北京宽连十方技术有限公司

2009-10

修改记录

文件编号	版本号	拟制人/修改人	拟制/修改日期	更改理由	主要更改内容（写要点即可）
	V1.0	XXX	yyyy.mm.dd	创建	

目录

1 概述

1.1 什么是Ophone

OPhone是中国移动今年主导的智能手机系列，采用了中国移动自主研发OMS手机操作系统。该平台采用开源的Linux作为系统内核，借鉴并兼容Android平台，集成灵活高效的Java应用框架，充分借鉴当下主流手机操作系统所具有的良好用户体验，提供了一套完整的电话解决方案和各类移动数据业务解决方案。

OPhone SDK是专为OPhone平台设计的软件开发套件，它包括OPhone API，OPhone模拟器，开发工具，示例代码和帮助文档。***OPhone SDK兼容Android SDK，因此开发者在开发OPhone应用的时候可以同时使用OPhone API和Android API。***

1.2 开发环境搭建

1.2.1 操作系统支持

Windows XP or Vista
Linux (tested on Linux Ubuntu 8.04)

1.2.2 开发环境要求

Eclipse IDE
Eclipse 3.4.2 (Ganymede)
○ Eclipse JDT plugin (included in most Eclipse IDE packages) ○ WST 3.0.4 ○ EMF 2.4.2 ○ GEF 3.4.2
JDK 5 or JDK 6 (JRE alone is not sufficient) ADT plugin WDT plugin

注：Eclipse要求的版本是3.4.2，其他的版本可能会在插件安装上有问题。

1.2.3 插件安装要求

- 安装过程参考：

  http://www.ophonesdn.com/documentation/ophone/gettingstarted/installing_sdk.html

1) 下载安装OPhone SDK

安装完ophoneSDK后，运行OPhone模拟器：现在你可以在命令行里面运行OPhone模拟器：

sdk_folder/tools/emulator -skin HVGA

2) 安装Eclipse插件 - ADT

ADT是为在Eclipse IDE下进行OPhone应用开发而提供的Eclipse插件。如果要使用Eclipse作为调试和编译的集成开发环境，则需要首先安装ADT。

注：你可以在SDK目录中找到ADT安装包：sdk_folder/tools/ophone/ADT-0.8.0.zip

3) 安装Eclipse 插件 (WDT)

如果你使用Eclipse作为Widget应用的开发环境，你还需要安装Eclipse插件WDT。WDT是Widget Develpment Tools的缩写，它集成了Wdiget工程开发需要的工具。这些工具可以实现Widget工程的创建、源代码的编辑、运行和调试等功能。WDT功能强大，可扩展性好，让Widget的开发变得更加简单和快捷。

注：你可以在OPhone SDK中找到WDT安装包：sdk_folder/tools/ophone/jil-wdt-site.zip

注:下载地址

1) Eclipse下载地址：
http://download.actuatechina.com/eclipse/eclipse/downloads/drops/R-3.4.2-200902111700/eclipse-SDK-3.4.2-win32.zip
2) OPhone SDK下载地址（需要注册为会员）：
http://www.ophonesdn.com/resource/sdk

1.3 OPHONE开发的知识要求

- 开发程序：有过JAVA程序开发经验
- 开发环境：Eclipse的熟练使用。
- 熟悉ophone SDK
- 熟悉android SDK.

1.4 官方开发技术网站

- Ophone的官方网站：http://www.ophonesdn.com/
- android的官方网站：http://code.google.com/android/index.html
  http://developer.android.com/sdk/index.html

2 Ophone开发入门

2.1 OPHONE初体验

2.1.1 Helloworld

相关过程请参考：

http://www.ophonesdn.com/documentation/ophone/gettingstarted/hello_ophone.html

这是OMSDN上的helloworld例程，接下来我们对该例程进行分析，初步了解一下如何编写OPhone代码，如果你具备面向对象基础，应该很容易理解。为了篇幅起见，暂时忽略localSearch函数的内容

```
1.  package oms.hello;
2.
3.  import ...
4.  import oms.servo.search.SearchProvider;
5.  import android.database.Cursor;
6.  import android.net.Uri;
7.  import android.os.Bundle;
8.
9.  public class HelloOPhone extends Activity {
10.     /** Called when the activity is first created. */
11.     @Override
12.     public void onCreate(Bundle icicle) {
13.         super.onCreate(savedInstanceState);
14.         setContentView(R.layout.main);
15.
16.        // step 2: call OPhone API(LocalSearch)
17.        String searchSelection = "type:" + SearchProvider.TYPE_CALL;
18.        String searchResult = localSearch(searchSelection);
19.     // step 3: output the result to the TextView
20.        if (searchResult != null) {
21.            TextView tv = (TextView) findViewById(R.id.textview);
22.            tv.setText(searchResult);
23.        }
24.
25.    }
26.
27.    public String localSearch(String searchSelection) {
1.              // search for SMS
2.              Uri uri = Uri.parse(SearchProvider.CONTENT_URI);
3.              Cursor cursor = getContentResolver().query(uri, null, searchSelection,
4.                      null, null);
5.
6.              StringBuffer result = new StringBuffer();
7.              result.append("#id   #calltype   #title   #time(#duration)\n");
8.              // print result out
9.              while (cursor.moveToNext()) {
```

```
10.             // Use cursor.respond function to get the data.
11.             Bundle extras = new Bundle();
12.             extras = cursor.respond(extras);
13.             // Extract the data from search result
14.             String id = extras.getString(SearchProvider.FIELD_ID);
15.             String calltype = extras.getString(SearchProvider.FIELD_CALL_TYPE);
16.             String title = extras.getString(SearchProvider.FIELD_TITLE);
17.             long time = Long.parseLong(extras.getString(SearchProvider.FIELD_TIM
    E));
18.             int duration = Integer.parseInt(extras.getString(SearchProvider.FIELD_C
    ALL_DURATION));
19.             result.append("\n").append(id)
20.                 .append("\n[").append(calltype).append("]")
21.                 .append("\t").append(title)
22.                 .append("\t").append(new Date(time).toString())
23.                 .append("(").append(duration).append(")")
24.                 .append("\n");
25.         }
26.
27.         cursor.close();
28.     return result.toString();
29.   }
30. }
```

程序分成三个步骤，1.初始化，2.调用 localSearch 返回结果，3.输出结果。

1. 在 create 之后，指定这个 Activity 的界面布局，需要从 R 中获得引用。
2. 接下来定义两个 String，这里用到了 SearchProvider.TYPE_CALL。通过定义呼叫类型，采用 localSearch 函数将其读出并修改格式，显示在 TextView 上。SearchProvider 为 OPhone 平台上的一个服务（“服务”前文介绍），该 class 提供了一个程序内部的搜索框架，文档说：Local search API consists of two parts, one is ContentProvider, which can parse given query string and return Cursor for hitting results, application using search ContentProvider need to handle the search result itself; another is Intent, which launch local search Activity with given arguments.该服务的 TYPE_CALL 是一个 query string，即搜索对象的类型，这是呼叫的类型，还有如下类型：

```
TYPE_APPLICATION
TYPE_WIDGET
TYPE_EMAIL
TYPE_MESSAGE
TYPE_FETION
TYPE_SMS
TYPE_MMS
TYPE_CONTACTS
TYPE_CALENDAR
TYPE_ALL_FILE
TYPE_FILE
TYPE_AUDIO
TYPE_VIDEO
TYPE_CALL
TYPE_WEB
```

3. 创建一个 TextView，这里我们注意到，创建时用到了资源的引用。
接下来，我们简要了解一下localsearch函数的功能。

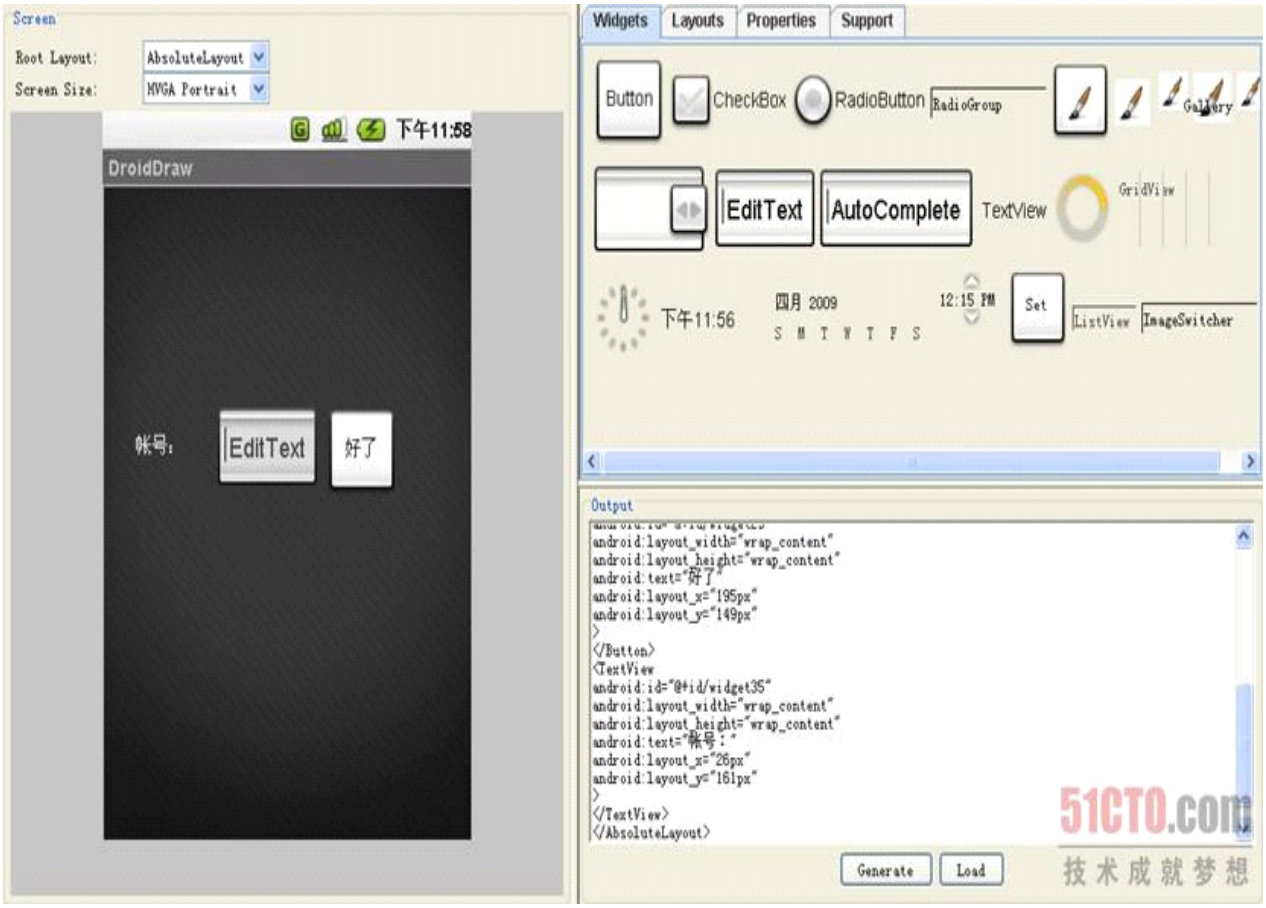
一开始，出现了URI这个概念，URI是通过前面提到的 ContentProvider来想搜索行为发送 Intent。 SearchProvider.CONTENT_URI实际上就是"content://search"。cursor 类似于查询结果的一个指针，通过getContentResolver函数来获得查询结果。

接下来的代码比较容易理解，通过下移查询指针，添加查询结果字符串，最后将指针关闭，返回查询结果的字符形式。

2.1.2 可视化开发工具

Android手机有着华丽的机身、流畅的执行速度，唯一欠缺的就是"具有视觉美感的UI设计员",所幸，在Google还未完整推出GUI的拖拉工具之前，已有网友以Java写出了好用的可视化GUI布局拖拉工具程序：DroidDraw。

DroidDraw目前是一个公开的Google Code，除了可以在线免费下载（http://code.google.com/p/droiddraw/）使用到计算机端执行之外，也提供在线直接使用的版本（http://www.droiddraw.org/）；DroidDraw同时还提供了源代码（Source Code），可供程序员自行参考或修改。



期待未来Google能将拖拉布局的功能纳入Android Editor当中，如此一来更能整合Android SDK以及开发环境的功能，且让我们拭目以待吧！

2.1.3 Android的调试工具—adb

Android手机操作系统作为一款开源的系统深受广大开发人员的喜爱。我们可以在这一系统的模拟器中对此进行相关修改，以达到自己的目的。比如在界面图形的操作上就可以通过Android adb这样的一款调试工具来进行自行编译。

Android 的主要调试工具 adb(Android debuging bridge)，ddms 是一个在 adb 基础上的一个图形化工具。

这里主要讲解 Android adb，它是一个命令行工具。而 ddms 功能与 adb 相同，只是它有一个图形化界面。对不喜欢命令操作方式的人来说是一个不错的选择。

这些命令在 Android sdk 下的 tools 目录下。这些命令在 linux 和 window 中都可运行

首先确定本机上有一个模拟器已启动。确定是否有模拟器已启动可以使用命令：

```
1.  adb devices
```



```
2.  List of devices attached
3.  emulator-5554 device
4.  emulator-5556 device
```

返回一个 5554 的模拟器。

给模拟器安装一个应用程序，使用命令 adb install <app_apk>

```
5.  adb install /home/myname/test.apk
```

t 在 Android adb 中，test.apk 是一个打包好的应用程序。

安装好的程序可以在用 adb shell 命令在模拟器目录/system/app 中找到，文件名字都和安装的一样。

文件传输：pc 机与模拟机之间的文件传输可以使用 adb pull 和 adb push

adb pull 是把文件从模拟机上复制到 pc 机上，使用方法如下

```
6.  adb pull < remote> < local>
```

其中 remote 代表模拟机文件路径，local 为 pc 机文件路径。

如：adb pull /system/app/test.apk /home/myname/test.apk

adb push 则相返，是把文件从 pc 机上复制到模拟机上，push 可以把任务文件都复制到模拟机上，如果是把一个 apk 文件上传到/system/app/下则和 adb install 作用一样。使用方法

```
7.  adb push < local> < remote>
```

大家都明白 Android 是一个操作系统平台，启动一个模拟器就启动了一个操作系统。可以使用 Android adb 连接到这个操作系统，并运行一些系统命令，就像平时大家用终端访问一台远程 linux/unix 服务器。可以使用 adb shell。如下进行 shell 后运行 ls 命令，在该 shell 下可运行 linux 下一些常用的命令,注意：adb shell 是只启动一个模拟器（emulator）的情况，如果启动了多个模拟器，如刚才使用的 adb devices 命令返回了两个模拟器，如果要连接其中一个则加参数-s:使用如：adb -s emulator-5554 shell

```
8.  adb shell
9.  # ls
10. sqlite_stmt_journals
11. cache
12. sdcard
13. etc
```

```
14.  init
15.  init.goldfish.rc
16.  init.rc
17.  data
18.  sys
19.  system
20.  proc
21.  default.prop
22.  sbin
23.  root
24.  dev
25.  #
```

Android adb 删除文件,如果使用 adb shell 直接进入用 rm 命令删除文件是删不掉的。在 adb shell 命令前运行 adb remount

```
26.  $adb remount;
27.  $adb shell
28.  #
```

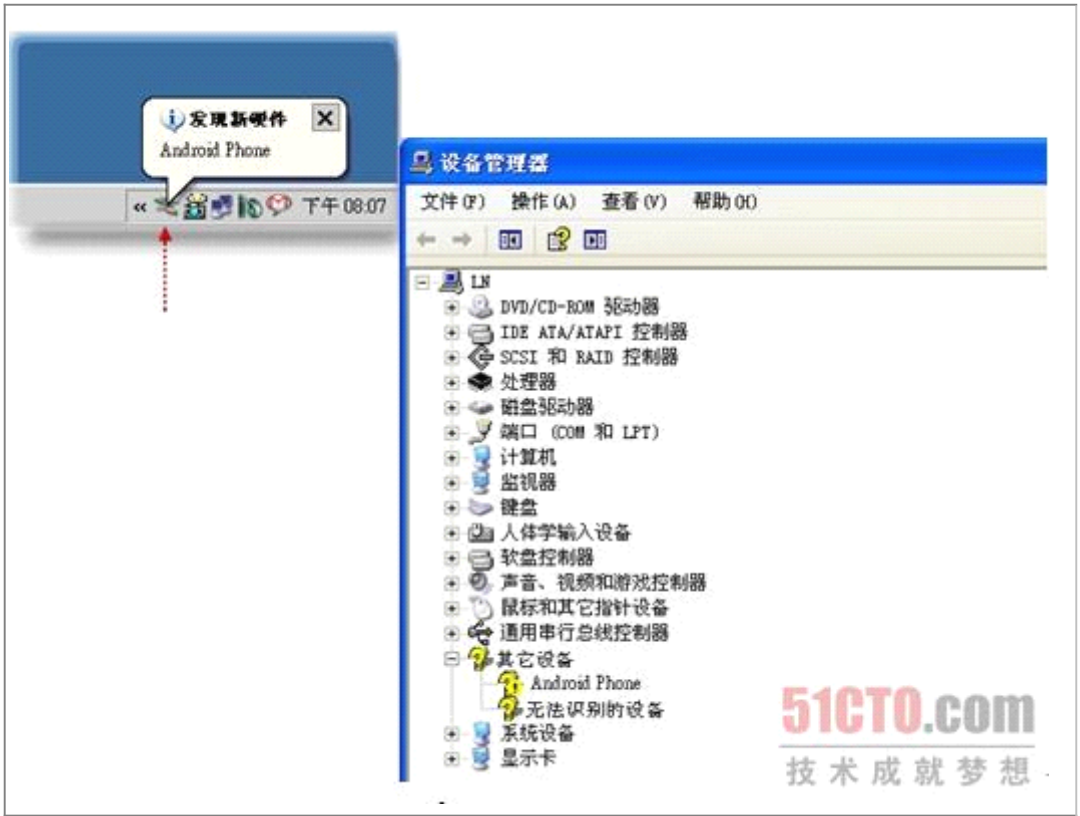
2.1.4 部署应用程序到Android手机

要部署程序在模拟器上运行，在先前 Hello World 的程序已经看过了，在项目名称上单击右键执行 Android 应用程序即可，但事实上，要将 Android 程序，部署在手机环境中进行测试，方法也是相同的,同样调用"Run As-Android Application"的方式执行,不同的是,需要事先安装好 Android 的 USB Driver,并且通过 USB 联机至手机,在与手机联机的状况下,就可以让 Eclipse 在运行 Android 程序时，直接将程序部署于实机环境中执行。

Android USB 驱动程序是随着 Android SDK 所提供的，每一个版本的 SDK 都可能有不同版本的 USB Driver Version，其存放在以下 Android SDK 解开后的参考位置，如：

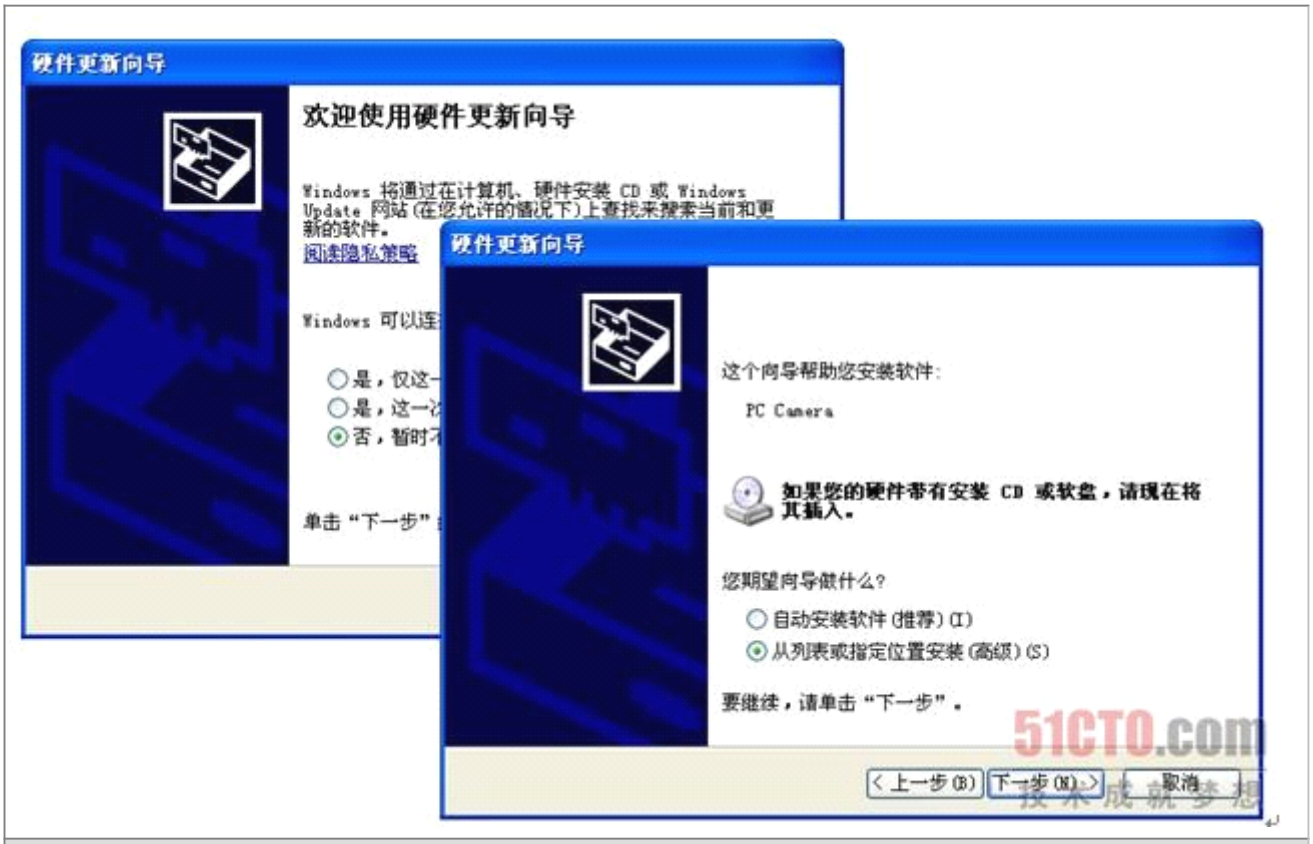
D:\SDK\android\usb_driver\

安装的步骤是先将手机以 USB 与计算机连接，操作系统会找到名为 Android Phone 的设备，但是却在装置管理员当中无法正确被识别，如下所示。



（点击查看大图）图 2-21 操作系统找到名为 Android Phone 的装置，但无法正确被识别

接着画面会跳出添加硬件向导，选择"从列表或指定位置安装（高级）"来自行挑选驱动程序位置。

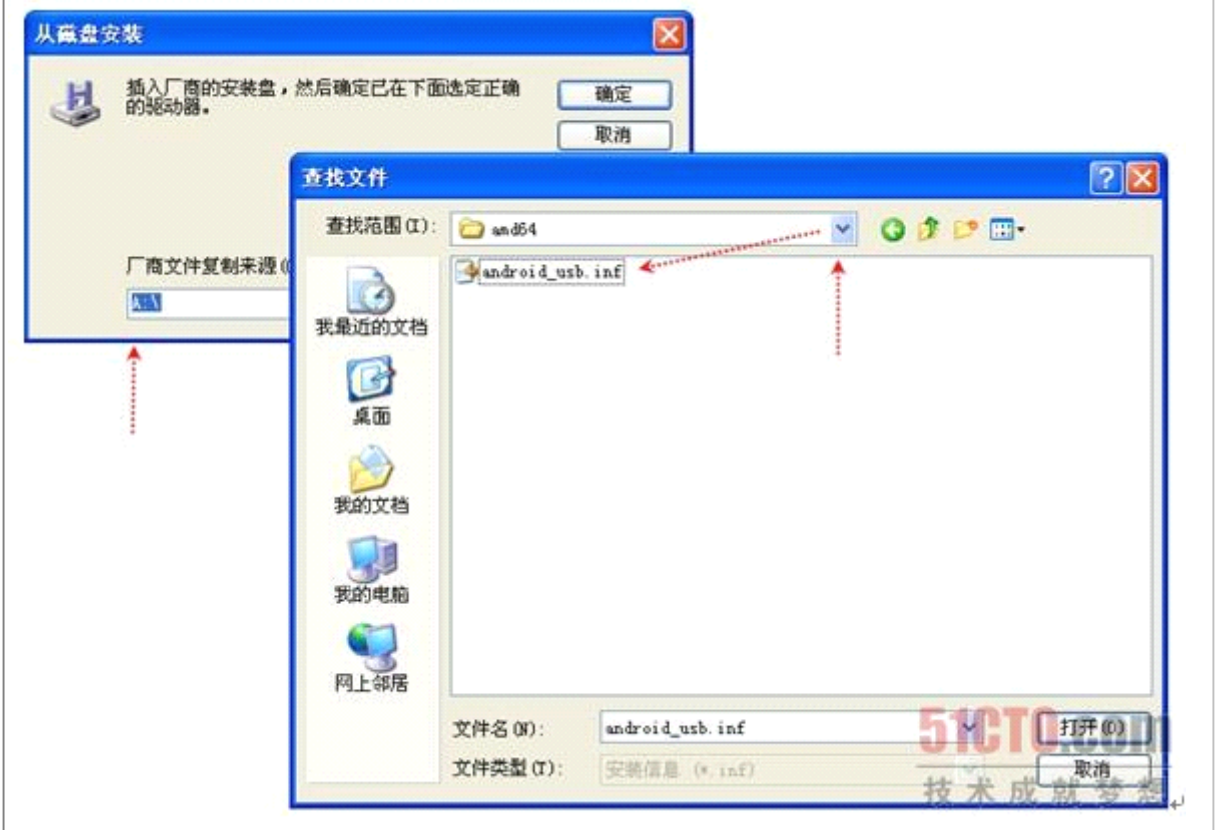


（点击查看大图）图 2-22 选择"从列表或指定位置安装"自己安装驱动程序

在"搜索和安装选项"的画面中，选择"不要搜索，我要自己选择要安装的驱动程序"选项，选择"显示所有设备"后，按下"下一步"。

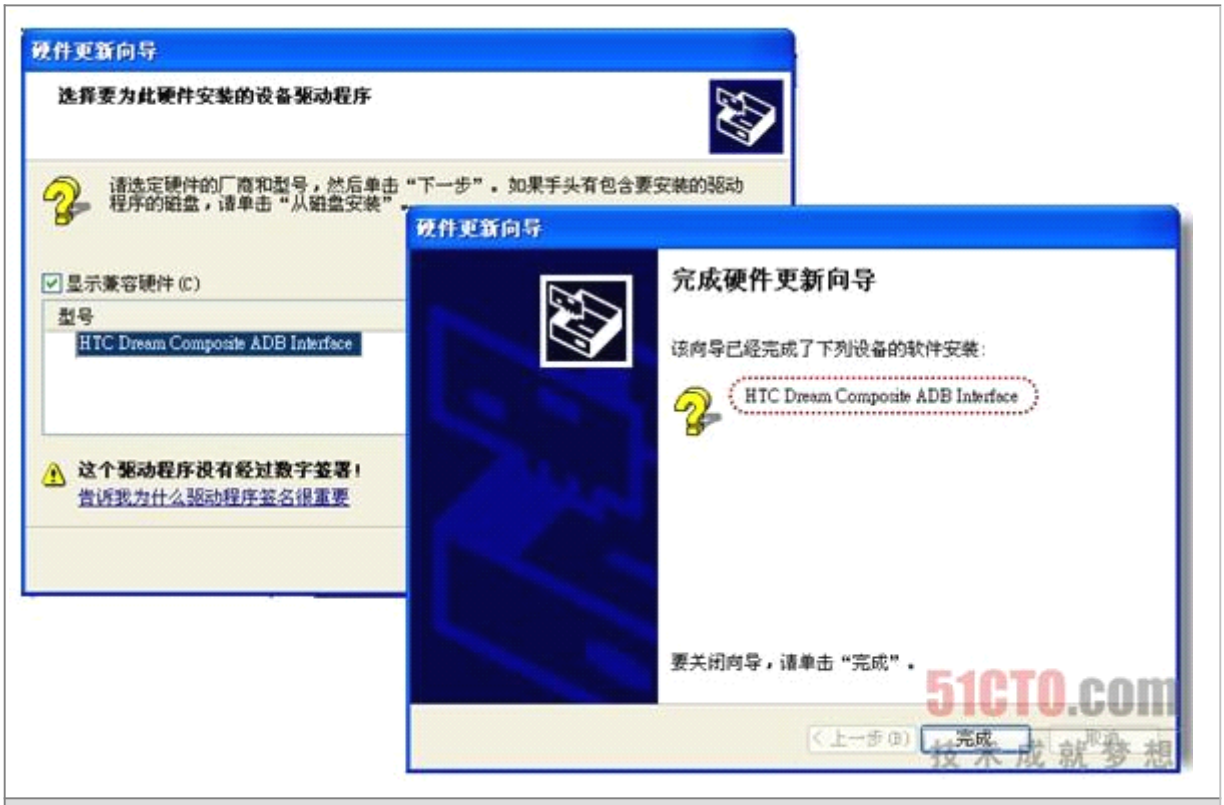
（点击查看大图）图 2-23 选择不要搜索系统数据，改以自行挑选硬件的方式

利用浏览按钮选择复制源为 Android USB Driver 程序路径：



（点击查看大图）图 2-24 选择 Android SDK 里所附的 USB Driver

选择驱动程序后，于显示兼容硬件列表中选择"HTC Dream Composite ADB Interface"，程序将 Android 手机的 USB ADB Interface 安装完成。



（点击查看大图）图 2-25 安装 Android USB ADB Interface 完成

设备管理器会自动新增一项 ADB Interface 的项目，表示已经顺利安装了 Android 手机与计算机的联机。

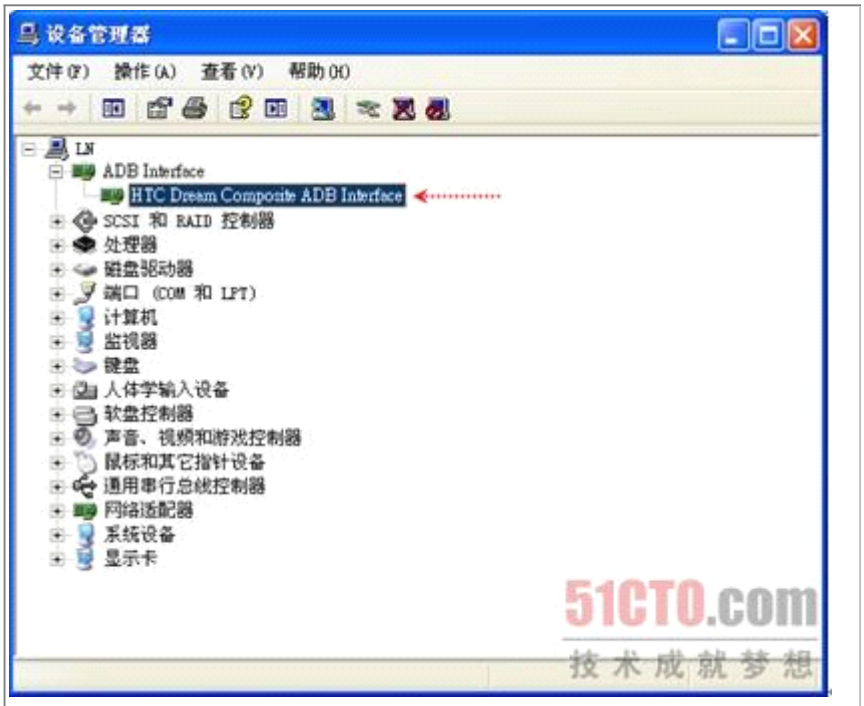
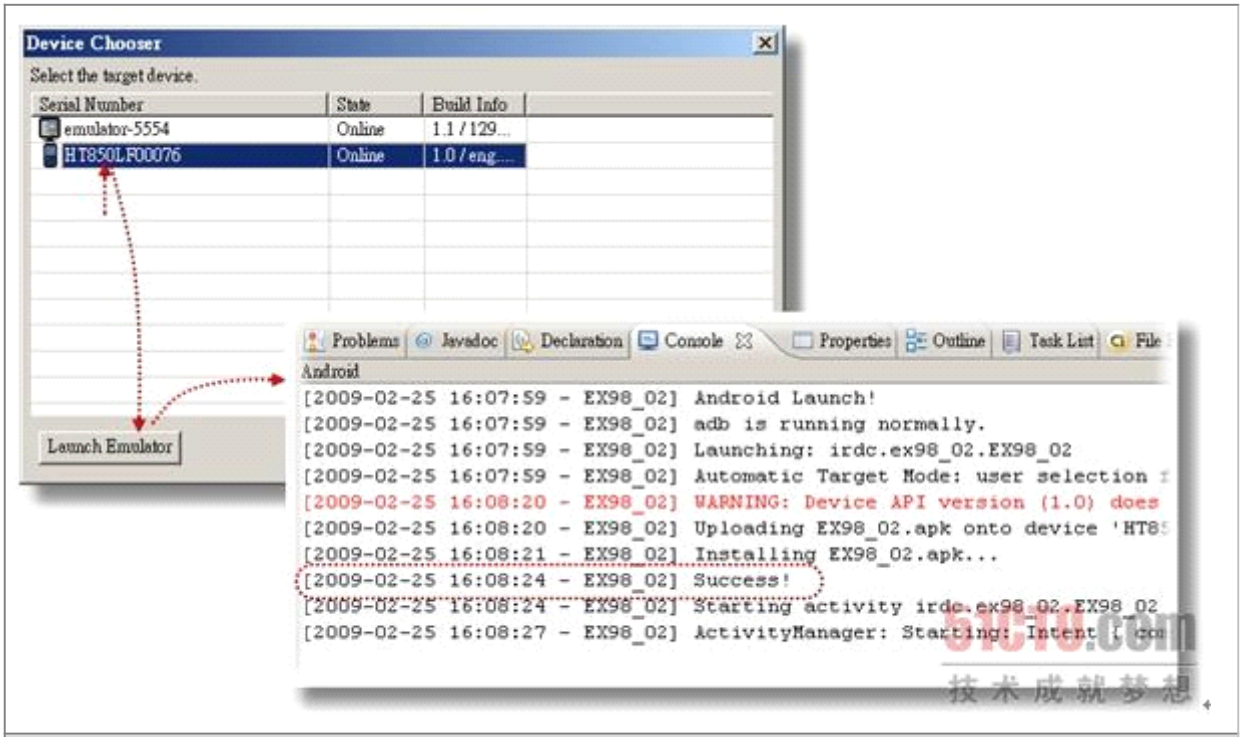


图 2-26 顺利安装了 Android 手机与计算机的联机

安装完 ADB Interface 之后，暂时还无法通过 Eclipse 将 Android 项目程序部署至手机上，必须先将手机上的 USB 调试（Debug）模式打开，在手机上执行"应用程序设置-开发>>USB 调试"。

（点击查看大图）图 2-27 将手机的 USB 调式模式打开

在 Eclipse 执行项目时，若程序发现先前已打开的模拟器与手机同时并存，那么将会跳出 Device Chooser 的窗口让开发者选择要部署的设备，下图为选择 Android G1 手机之后，于 Console 里显示正确执行的 Log 纪录。



（点击查看大图）图 2-28 上图中的警告为开发使用的是 SDK 1.1 但手机是 SDK 1.0 的警告

部署程序到手机上测试是最适合的方法，因为许多功能皆需要手机才能进行测试，如 WiFi 驱动程序、平衡感应器、电池剩余计量等等。

2.2 OPHONE API介绍

OPhone SDK兼容Android SDK，因此开发者在开发OPhone应用的时候可以同时使用OPhone API和Android API，所以这里介绍OPHONE特有的API功能和调用方法，OPhoneSDK主要提供的特有API主要有如下

- *本地搜索（Local Search）*
- *主屏（Home）*
- *邮件（Mail）*

2.2.1 本地搜索API的使用

2.2.2 主屏（Home）API 的使用

2.2.3 邮件（Mail）API 的使用

3 Android开发指南

3.1 Android概述

Android是一个针对移动设备的程序集，其中包括一个操作系统，一个中间件和一些关键性应用．本文首先概览了Android SDK提供的工具和APIs，当您在使用Java语言来开发Android平台的应用时您会用到它们。

3.1.1 Android术语列表概览

Android 手机操作系统中有一些常用的术语需要我们在学习的过程中熟练掌握，才能更好的运用这一系统来帮助我们完成一些需求。那么大家可以一起来看看我们在这里总结的几个常见 Android 术语。

1，apk 扩展名：apk 是 Android 包的扩展名，一个 Android 包包含了与某个 Android 应用程序相关的所有文件，apk 文件将 AndroidManifest.xml 文件、应用程序代码(dex 文件)、资源文件和其他文件组成一个压缩包，一个项目只能打包压缩成一个 apk 文件。

2， dex 扩展名：Android 的程序被编译成.dex(Dalvik Executable)格式文件，然后再进行打包生成可直接安装的 apk 文件。

3， 应用程序(APP) ：一个或多个 Activity、服务、监听和 Intent 接收器的集合，一个应用程序有一个文件清单，并且打包成一个 apk 文件。 4， Action：对 Intent 发送器意图的描述，一个活动是一个指派给 Intent 的字符串值。

5， ADB（ Android Debug Bridge ）：SDK 自带的一个基于命令行的调试程序。它提供了设备浏览工具、设备上的拷贝工具和为调试转寄端口的功能。

6， 内容源：Android 术语中的内容源是建立在类 ContentProvider 之上的用于处理指定格式的内容请求字符串，并返回指定格式的数据的类。

7， Dalvik Android：虚拟机的名字,Dalvik 虚拟机是一个只能解释执行 dex 文件的虚拟机，dex 文件针对存储性能和内存管理进行了优化。 Dalvik 虚拟机是基于寄存器的虚拟机，并且能够运行经过 Dalvik 自带的“dx”工具转换过的 Java 类。 虚拟机运行在兼容 Posix 的操作系统上，依赖于底层的功能(如线程和低级内存管理)。Dalvik 的核心类库有意做得与 Java 标准版非常类似，但它明显更适合小型移动设备。

8， 意图(Intent) ：意图是一个 Intent 类，它包含很多描述调用者意图做什么的字段。调用者发送意图到 Android 意图处理器，意图处理器会遍历所有应用程序的意图过滤器来查找与该意图最匹配的 Activity。意图字段包括渴望的动作、种类、数据、数据的 MIME 类型、一个处理类和其他约束。

9， 意图过滤器(intent-filter)：Activity 和意图接收器(Receiver)在它们的文件清单中包含一个或多个过滤器，用来描述什么类型的意图或者信息是它们能处理或想接收的。一个意图过滤器列出了一系列要求，例如，意图或信息必须满足的数据类型、被请求的动作和 URI 的格式。 对于 Activity，Android 搜索意图和 Activity 过滤器匹配程度最高的 Activity；对于消息，Android 会将消息转发给所有匹配意图过滤器的接收器。

10，Intent 接收器(Receiver)：一个监听是由 Context.broadcastIntent()发出的信息广播的类。

11，布局资源：一个描述 Activity 屏幕布局的 XML 文件。

12，文件清单：应用程序中的一个 XML 文件，用于描述包中多个 Activity、Intent 过滤器、服务和其他内容。可以打开 AndroidManifest.xml 查看其包含的内容。

13，资源：整个 Android 术语可以为用户提供的 XML、位图或其他文件，构建程序时会导入进来，稍后会被代码加载，Android 支持多种类型的资源，请参考 Resources 中的详细描述，程序定义的资源文件应当保存在 res/ 子目录下。

14，服务(Service) ：运行在后台执行多种固定任务的类，如播放音乐或检测网络活动。

15，URIs：Android 使用 URI 字符串请求数据(如通信录列表)和动作(如在浏览器中打开网页)。字符串可以具有不同的格式。所有请求数据的 URI 必须以“content://”开头。有效的动作 URI 字符串会被设备上的适当的程序处理，例如，以“ http://”开头的 URI 字符串会被浏览器处理。

16，AIDL(AndRoid 接口描述语言)：是一种接口描述语言；编译器可以通过 aidl 文件生成一段代码，通过预先定义的接口达到两个进程内部通信进程的目的.

17，JNI：java 本地编程接口，是 Java Native Interface 的英文缩写。他能够使 java 代码与用其他编程语言编写的应用程序和库进行互操作。（其他编程语言大多是 c, c++和汇编语言。）

3.1.2 Android的特征

程序程序框架可重用及可复写组件组成

针对移动设备优化过的Dalvik虚拟机

整合浏览器，该浏览器基于开源的WebKit引擎开发

提供了优化过得图形系统，该系统由一个自定义的2D图形库；一个遵循OpenGL ES 1.0标准(硬件加速)的3D图形库组成

使用SQLite来实现结构化数据的存储

媒体方面对一些通用的audio, video, 和图片格式提供支持(MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)

GSM技术(依赖硬件)

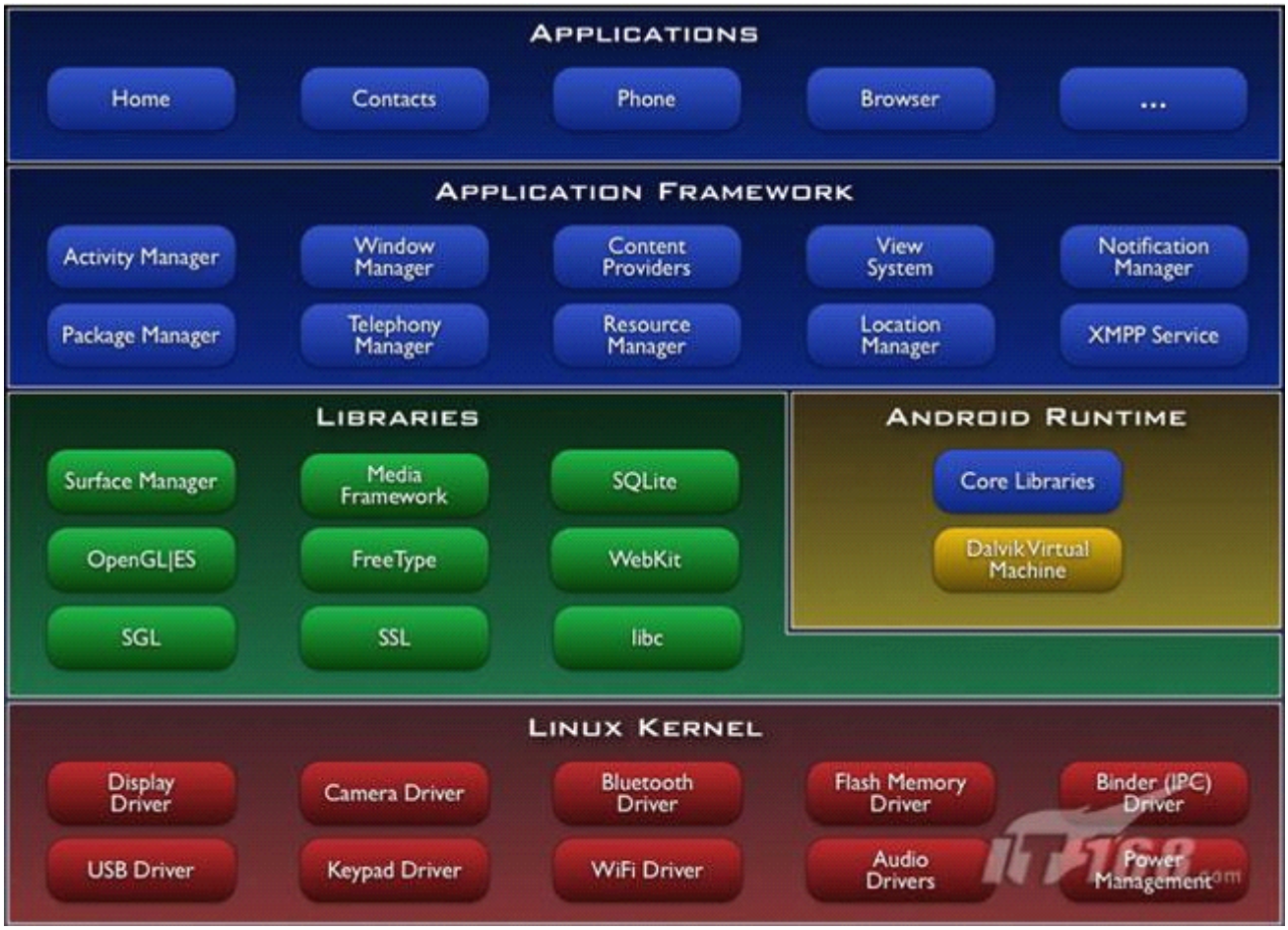
蓝牙, EDGE, 3G和WiFi(依赖硬件)

Camera, GPS, 指南针, 和加速计 (依赖硬件)

非常丰富的开发环境，包括一个设备模拟器，调适工具，内存和效率调优工具和一个Eclipse的插件

3.2 Android架构

3.2.1 系统架构说明



android 的系统架构和其操作系统一样，采用了分层的架构。从架构图看，android 分为四个层，从高层到低层分别是**应用程序层、应用程序框架层、系统运行库层**和 **linux 核心层**。

1） **应用程序**

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

2） **应用程序框架**

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

隐藏在每个应用后面的是一系列的服务和系统, 其中包括:

a) 丰富而又可扩展的视图(Views)，可以用来构建应用程序， 它包括列表(lists)，网格(grids)，文本框(text boxes)，按钮(buttons)， 甚至可嵌入的 web 浏览器。

b) 内容提供器(Content Providers)使得应用程序可以访问另一个应用程序的数据(如联系人数据库)， 或者共享它们自己的数据

c) 资源管理器(Resource Manager)提供 非代码资源的访问，如本地字符串，图形，和布局文件(layout files)。

d) 通知管理器 (Notification Manager) 使得应用程序可以在状态栏中显示自定义的提示信息。

e) 活动管理器(Activity Manager) 用来管理应用程序生命周期并提供常用的导航回退功能。

3) 系统运行库

a) 程序库

Android 包含一些 C/C++库，这些库能被 Android 系统中不同的组件使用。它们通过 Android 应用程序框架为开发者提供服务。以下是一些核心库：

* 系统 C 库 - 一个从 BSD 继承来的标准 C 系统函数库(libc)， 它是专门为基于 embedded linux 的设备定制的。

* 媒体库 - 基于 PacketVideo OpenCORE;该库支持多种常用的音频、视频格式回放和录制，同时支持静态图像文件。编码格式包括 MPEG4, H.264, MP3, AAC, AMR, JPG, PNG 。

* Surface Manager - 对显示子系统的管理，并且为多个应用程序提 供了 2D 和 3D 图层的无缝融合。

* LibWebCore - 一个最新的 web 浏览器引擎用，支持 Android 浏览器和一个可嵌入的 web 视图。

* SGL - 底层的 2D 图形引擎

* 3D libraries - 基于 OpenGL ES 1.0 APIs 实现;该库可以使用硬件 3D 加速(如果可用)或者使用高度优化的 3D 软加速。

* FreeType -位图(bitmap)和矢量(vector)字体显示。

* SQLite - 一个对于所有应用程序可用，功能强劲的轻型关系型数据库引擎。

b) Android 运行库

Android 包括了一个核心库，该核心库提供了 JAVA 编程语言核心库的大多数功能。

每一个 Android 应用程序都在它自己的进程中运行，都拥有一个独立的 Dalvik 虚拟机实例。Dalvik 被设计成一个设备可以同时高效地运行多个虚拟系统。 Dalvik 虚拟机执行(.dex)的 Dalvik 可执行文件，该格式文件针对小内存使用做了优化。同时虚拟机是基于寄存器的，所有的类都经由 JAVA 编译器编译，然后通过 SDK 中 的 "dx" 工具转化成.dex 格式由虚拟机执行。

Dalvik 虚拟机依赖于 linux 内核的一些功能，比如线程机制和底层内存管理机制。

4) Linux 内核

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

3.2.2 Apk文件结构简介

做过OPhone/Android应用开发的人，对apk文件应该不会陌生。apk文件，即Android application package文件。每个要安装到OPhone平台的应用都要被编译打包为一个单独的文件，后缀名为.apk，其中包含了应用的二进制代码、资源、配置文件等,其目录结构如下：

3.1 Manifest 文件

AndroidManifest.xml 是每个应用都必须定义和包含的,它描述了应用的名字、版本、权限、引用的库文件等等信息[,]，如要把 apk 上传到 Google Market 上，也要对这个 xml 做一些配置。网上已有很多资料，在此就不多做介绍了。

在 apk 中的 AndroidManifest.xml 是经过压缩的,可以通过 AXMLPrinter2 工具[,]解开，具体命令为：

view plaincopy to clipboardprint?

```
java -jar AXMLPrinter2.jar AndroidManifest.xml
```

3.2 META-INF 目录

META-INF 目录下存放的是签名信息，用来保证 apk 包的完整性和系统的安全。在 eclipse 编译生成一个 api 包时，会对所有要打包的文件做一个校验计算，并把计算结果放在 META-INF 目录下。而在 OPhone 平台上安装 apk 包时，应用管理器会按照同样的算法对包里的文件做校验，如果校验结果与 META-INF 下的内容不一致，系统就不会安装这个 apk。这就保证了 apk 包里的文件不能被随意替换。比如拿到一个 apk 包后，如果想要替换里面的一幅图片，一段代码， 或一段版权信息，想直接解压缩、替换再重新打包，基本是不可能的。如此一来就给病毒感染和恶意修改增加了难度，有助于保护系 统的安全。

3.3 classes.dex 文件

classes.dex 是 java 源码编译后生成的 java 字节码文件。由于 Android 使用的 dalvik 虚拟机与标准的 java 虚拟机是不兼容的，dex 文件与 class 文件相比，不论是文件结构还是 opcode 都不一样。目前常见的 java 反编译工具都不能处理 dex 文件。

Android 模拟器中提供了一个 dex 文件的反编译工具，dexdump。用法为首先启动 Android 模拟器，把要查看的 dex 文件用 adb push 上传的模拟器中，然后通过 adb shell 登录，找到要查看的 dex 文件，执行 dexdump xxx.dex。

总的来说 dexdump 反编的结果可读性很差。

目前在网上能找到的另一个 dex 文件的反编译工具是 Dedexer。Dedexer 可以读取 dex 格式的文件，生成一种类似于汇编语言的输出。这种输出与 jasmin[ ]的输出相似，但包含的是 Dalvik 的字节码。

dedexer 与 dexdump 相比至少有 3 个优点:

一、不需要在 android 模拟器中运行。

二、把 dex 文件按照 java 源代码 package 的目录结构建好了目录，每个 class 文件对应一个 ddx 文件。不像 dexdump 那样把所有的结果都放在一起。

三、按照 Dedexer 作者的说法，可以把 Dedexer 作为一个像 jasmin 那样的反编译引擎，目前好多强大的 java 反编译工具都是以 jasmin 作为反编译引擎的

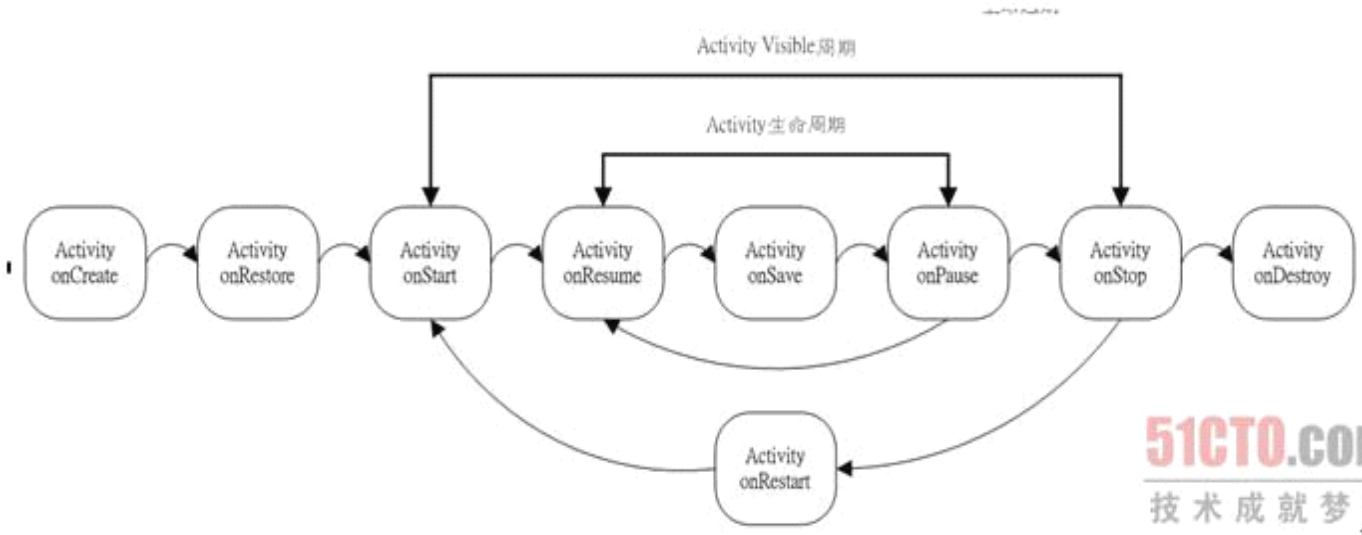
3.4 res 目录

res 目录存放资源文件。关于 apk 文件中的资源管理，OPhone SDN 网站上已经有文章做过详细介绍[ ]，就不在此敷述。

3.5 resources.arsc

编译后的二进制资源文件。

3.2.3 Application的生命周期



注：Activity 类的应用程序有其默认运行的方式，为了确保应用程序运行的优先级，理解 Activity 在手机运行时的生命周期，及其可视性（Visible）周期。

以 Hello World 程序里继承自 Activitry 类开始，一旦程序被执行，即会照以上流程顺序进行，若需要在 Activity 程序里编写程序，默认常见的进入点为重写 onCreate（Activity）或 onStart（Service），重写的方式可通过 Eclipse 来选择，方法为将鼠标光标停在继承自 Activity 的空白处，单击鼠标右键展开菜单，点开执行"Source-Override/Implement Methods"功能。

最后，比较值得一提的是 onResume() 与 onPause()，这两个方法为 Activity 在 onCreate 之后运行过程中的生命周期，当程序失去前端焦点、或者被关闭，就会触发 Activity 的 onPause() 状态；当应用程序被再次唤醒，则会回到 onResume() 状态，故在编写与 User 互动的程序过程中，需注意 User 暂时离开 Activity（或前往不同的 Activity、不同的 Service）前，需要处理的工作都会摆在 onPause() 当中执行。以一个通过网络 FTP 下载 mp3 的 Activity 为例，暂停下载工作可以写在 onPause 里，需要接续前一次的下载等处理，则由 onResume 负责

在多数情况下，每个 Android 应用运行在自己的 Linux 进程中. 当一个应用的某段 code 需要运行的时候这个进程将会被创建，直到不再需要该应用或系统要为其他的应用释放内存的时候才停止.

一个非常重要且少有的特性是，应用进程的存活时间不是由这个应用直接控制的. 而是由系统决定的，系统会根据每个已知的正在运行的应用情况来定夺，包括，该应用对用户的重要性和系统全部可用内存.

对于开发人员来讲，了解每个应用组件(尤其是，Activity，Service，和 IntentReceiver)对于应用进程存活时间的影响是非常重要的. 如果没有正确使用，可能会导致应用进程在处理重要工作的时候被系统杀掉.

在对应用进程生命周期的理解中，一个典型的错误就是当一个 IntentReceiver 接收到 Intent 之后，会在自己的 onReceiveIntent()方法中开起一个线程，而后 return 这个方法. 一旦这个方法 return，系统会认为这个 IntentReceiver 不在处于活跃状态，也就认为他的宿主进程不再需要(除非还包有其他活跃的应用组件). 以至于当系统需要回收内存的时候会随时释 kill 掉这个进程，中止其中的子线程. 解决这个问题的办法是在 IntentReceiver 中启动一个 Service，这样系统会知道在这个进程中还有活跃的任务需要完成.

为了决定在内存较低的时候杀掉哪个进程，Android 会根据运行在这些进程内的组件及他们的状态把进程划分成一个"重要程度层次". 其重要的程度按以下规则排序:

1. 前端进程可以是一个持有运行在屏幕最前端并与用户交互的 Activity 的进程(onResume 方法被调用时)，也可以是持有一个正在运行的 IntentReceiver(也就是说他正在执行自己的 onReceiveIntent 方法)的进程. 在系统中，只会有少数这样的进程，并且除非内存已经低到不够这些进程运行，否则系统不会主动杀掉这些进程. 这时，

设备通常已经达到了需要内存整理的状态，所以杀掉这些进程是为了不让用户界面停止响应.

2 可视进程是持有一个被用户可见，但没有显示在最前端 (onPause 方法被调用时)的Activity的进程. 举例来说，这种进程通常出现在一个前端Activity以一个对话框出现并保持前一个 Activity 可见时. 这种进程被系统认为是极其重要的，并且通常不会被杀掉，除非为了保持所有前端进程正常运行不得不杀掉这些可见进程.

3 服务进程是持有一个Service的进程，该Service是由startService()方法启动的，尽管这些进程用户不能直接看到，但是通常他们做的工作用户是十分关注的(例如，在后台播放 mp3 或是在后台下载 上传文件)，所以，除非为了保持所有的前端进程和可视进程正常运行外，系统是不会杀掉服务进程的.

4 后台进程是持有一个不再被用户可见的 Activity(onStop()方法被调用时)的进程. 这些进程不会直接影响用户体验. 加入这些进程已经完整的,正确的完成了自己的生命周期(访问 Activity 查看更多细节)，系统会在为前三种进程释放内存时随时杀掉这些后台进程. 通常会有很多的后台进程在运行，所以这些进程被存放在一个 LRU 列表中，以保证在低内存的时候，最近一个被用户看到的进程会被最后杀掉.

当需要给一个进程分类的时候，系统会在该进程中处于活动状态的所有组件里掉选一个重要等级最高作为分类依据. 查看 Activity，Service,和 IntentReceiver 的文档，了解每个组件在进程整个生命周期中的贡献. 每一个 classes 的文档详细描述他们在各自应用的生命周期中所起得作用.

3.2.4 解析Android程序

在 Android 应用程序中有四个构建块：

- 活动(Activity)
- 活动内容接受器(Intent Receiver)
- 服务(Service)
- 内容提供器(Content Provider)

1. 活动（Activity）

Activity 是 Android 构造块中最基本的一种，在应用中，一个 activity 通常就是一个单独的屏幕。 每一个 activity 都被实现为一个独立的类，并且继承于 Activity 这个基类。这个 activity 类将会显示由几个 Views 控件组成的用户接口，并对事件做出响应。大部份的应用都会包含多个的屏幕。例如， 一个短消息应用程序将会有一个屏幕用于显示联系人列表，第二个屏幕用于写短消息，同时还会有用于浏览旧短消息及进行系统设置的屏幕。每一个这样的屏幕,

就是一个 activity。从一个屏幕导航到另一个屏幕是很简单的。在一些应用中，一个屏幕甚至会返回值给前一个屏幕。

当一个新的屏幕打开后，前一个屏幕将会暂停，并保存在历史堆栈中。用户可以返回到历史堆栈 中的前一个屏幕。当屏幕不再使用时，还可以从历史堆栈中删除。默认情况下，Android 将会保留从主 屏幕到每一个应用的运行屏幕。

Android 使用了 Intent 这个特殊类，实现在屏幕与屏幕之间移动。Intent 类用于描述一个应用想要做什么事。 在 Intent 的描述结构中，有两个最重要的部分：动作和动作对应的数据。典型的动作类型有：MAIN（activity 的门 户）、VIEW、PICK、EDIT 等。而动作对应的数据则以 URI 的形式进行表示。例如：要查看一个人的联系方式，你需要 创建一个动作类型为 VIEW 的 intent，以及一个表示这个人的 URI。

一个屏幕到另一个屏幕之间的导航是通过解析意图（Intent）来实现的。当向前导航时，activity 将会调用 startActivity (Intent myIntent)方法。然后，系统会在所有安装的应用程序中定义的 IntentFilter 中查找，找到最匹配 myIntent 的 Intent 对应的 activity。新的 activity 接收到 myIntent 的通知后，开始运行。当 startActivity 方法被调用将触发解析 myIntent 的动作，这个机制提供了两个关键好处：

- Activity能够简单的通过在Intent表中发送请求，从其他组件中复用功能
- Activity能够在任何时候由一个带有相同IntentFilter的Activity替换

一个 Android 应用程序最基本的功能单位是行为（Activity）—— 是 android.app.Activity 类的一个对象。一个行为可以做很多事情，但是它本身并不能使自己显示在屏幕上。为了解决这个问题，你可以使用视图（views）和视图组（viewgroups）——它们是 Android 平台上的最基本的用户界面元素。

2. 活动内容接受器(Intent Receiver)

当你希望你的应用能够对一个外部的事件（如当电话呼入时，或者数据网络可用时，或者到了晚上时）做出响应， 你可以使用一个Intent Receiver。虽然Intent Receiver在感兴趣的事件发生时，会使用NotificationManager 通知用户，但它并不能生成一个UI。Intent Receiver在AndroidManifest.xml中注册，但也可以在代码中使用 Context.registerReceiver()进行注册。当一个intent receiver被触发时，你的应用不必对请求调用intent receiver，系统会在需要的时候启动你的应用。各种应用还可以通过使用Context.broadcastIntent()将它们自己的 intent receiver广播给其它应用程序

3. 服务(Service)

一个Service是一段长生命周期的，没有用户界面的程序。比较好的一个例子就是一个正在从播放列表中播放歌曲的媒 体播放器。在一个媒体播放器的应用中，应该会有多个activity，让使用者可以选择歌曲并播放歌曲。然而，音乐重放这个 功能并没有对应的activity，因为使用者当然会认为在导航

到其它屏幕时音乐应该还在播放的。在这个例子中，媒体播放器 这个activity会使用Context.startService()来启动一个service，从而可以在后台保持音乐的播放。同时，系统也将 保持这个service一直执行，直到这个service运行结束。另外，我们还可以通过使用Context.bindService()方法，连接 到一个service上（如果这个service还没有运行将启动它）。当连接到一个service之后，我们还可以service提供的接口 与它进行通讯。拿媒体播放器这个例子来说，我们还可以进行暂停、重播等操作。

4. 内容提供器(Content Provider)

应用程序能够将它们的数据保存到文件中、SQL数据库中，甚至是任何有效的设备中。当你想将你的应用数据与其它的 应用共享时，Content Provider将会很有用。一个Content Provider类实现了一组标准的方法，从而能够让其它的应用 保存或读取此Content Provider处理的各种数据类型。

3.3 开发基础

1. Android类库常用类型解析

android类库的相关内容在这篇文章中总结了一下。可以作为初学者们的学习参考对象，来充分掌握这一方面的应用，满足自己的需求。

Android 是由谷歌公司推出的一款基于 Linux 平台的开源手机操作系统平台。在这一新推出的 Android 操作系统中，有很多比较新的知识值得我们编程人员去深入的研究。比如 Android 类库的使用技巧等。

在 Android 类库中，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

以上就是我们为大家总结的 Android 类库相关内容。

2. Android屏幕元素相关概念详解

在 Android 手机操作系统中有很多比较重要的知识点需要我们在学习的过程中详细了解以方便我们将来的应用。比如 Android 屏幕元素等等。在这里就会为大家详细介绍一下有关 Android 屏幕元素分层结构。

android.app.Activity

对于一个 Android 应用来说，android.app.Activity 类实例是一个最基本的功能单元。一个 Activity 实例可以做很多的事情，但是它本身无法显示在屏幕上，而是借助于 Viewgroup 和 View，这两个才是 Android 平台上最基本的两个用户界面表达单元。

android.view.ViewGroup

ViewGroup 是一个特殊的 View 类，它继承于 android.view.View。它的功能就是装载和管理下一层的 View 对象和 ViewGroup 对象。ViewGroup 是布局管理器（layout）及 view 容器的基类。ViewGroup 中，还定义了一个嵌套类 ViewGroup.LayoutParams。这个类定义了一个显示对象的位置、大小等属性，view 通过 LayoutParams 中的这些属性值来告诉父级，它们将如何放置。

在这里，继承于 ViewGroup 的一些主要的布局类如下：

1、FrameLayout：最简单的一个布局对象。它里面只显示一个显示对象。Android 屏幕元素中所有的显示对象都将会固定在屏幕的左上角，不能指定位置。但允许有多个显 示对象，但后一个将会直接在前 一个之上进行覆盖显示，把前一个部份或全部挡住（除非后一个是透明的）。

2、LinearLayout：以单一方向对其中的显示对象进行排列显示，如以垂直排列显示，则布局管理器中将只有一列；如以水平排列显示，则布局管理器中将只有一行。同时，它还可以对个别的显示对象设置显示比例。

3、TableLayout：以拥有任意行列的表格对显示对象进行布局，每个显示对象被分配到各自的单元格之中，但单元格的边框线不可见。

4、AbsoluteLayout：允许以坐标的方式，指定显示对象的具体位置，左上角的坐标为(0, 0)，向下及向右，坐标值变大。这种布局管理器由于显示对象的位置定死了，所以在不同的设备上，有可能会出现最终的显示效果不一致。

5、RelativeLayout：允许通过指定显示对象相对于其它显示对象或父级对象的相对位置来布局。如一个按钮可以放于另一个按钮的右边，或者可以放在布局管理器的中央。
在 Android 中，提供了很多的布局管理器，这里也不一一列举，开发者可以根据实际需要，选择合适的布局管理器。

android.view.View

View 是所有 view 类的基类，一个 view 通常占用 Android 屏幕元素上的一个矩形区域，并负责绘图及事件处理。View 是所有窗体部件的基类，是为窗体部件服务的，这里的窗体部件即 UI 控件，如一个按钮或文本框。Android 已经为我们提供了一系列的标准 UI 控件供我们直接使用，同时，我们也可以通过继承于 View 类或 View 的子类，来实现我们自定义的 UI 控件。

要定制我们自己的 UI 控件，需要重载 View 类中的一些方法，以下表格列出 View 提供出来的，供重载的方法，这些方法不必都要重载，但至少要实现 onDraw(android.graphics.Canvas)方法。

当你为一个 activty 添加一个可见的 view, 并且运行这个 activty 时,android 通常情况下会自动按照下列顺序来触发 view 的相关事件

- onAttachedToWindow
- onMeasure
- onSizeChanged
- onLayout
- onDraw

对于 Android 应用中的一个屏幕，Android 屏幕元素是按层次结构来描述的。要将一个屏幕元素层次树绑定在一个屏幕上显示，Activity 会调用它的 setContentView()方法并且传入这个层次树的根节点引用。当 Activity 被激活并且获得焦点时，系统会通知 activity 并且请求根节点去计算并绘制树，根节点就会请求它的子节点去绘制它们自己。

Android 屏幕元素中每个树上的 ViewGroup 节点会负责绘制它的子节点。ViewGroup 会计算它的有效空间，布局所有的子显示对象，并最终调用所有的子显示对象的 Draw()方法来绘制显示对象。各个子显示对象可以向父对象请求它们在布局中的大小和位置，但最终决定各个子显示对象的大小和位置的是父对象

3. Android资源应用技巧剖析

Android 资源的应用可以帮助我们实现界面的一些特定需求。那么在这篇文章中大家将会了解到这其中的应用技巧，方便大家的应用。

在这篇文章中我们将会针对 Android 资源的相关概念为大家详细讲解有关界面布局的一些应用，加深大家对界面处理的理解。

1.添加菜单 menu.add(0, Menu.FIRST+1, 1, R.string.menu_open);

menu.add(0, Menu.FIRST+2, 2, R.string.menu_edit);代码中的 R.string.menu_open/menu_edit

这些其实是指 Android 资源文件中的 ID,映射到具体的资源，这里是映射到字符串资源 menu_open, menu_edit，其具体的值可以看 res/values/string.xml 在这里定义了字符串的值：

```
1.  < ?xml version="1.0" encoding="utf-8"?>
2.  < resources>
3.  < string name="hello">Hello World, HelloActivity!< /string>
4.  < string name="app_name">HelloWorld< /string>
5.  < string name="menu_open">Open< /string>
6.  < string name="menu_edit">Edit< /string>
7.  < string name="menu_update">Update< /string>
8.  < string name="menu_close">Close< /string>
9.  < /resources>
```

在 Android 中，Activity 显示的布局也可在 Android 资源中定义，并且以可视化的方式来操作布局对应的 XML 文件。可以看 res/layout/main.xml 这就是一个布局文件，这里指定了这个布局里有哪些界面元素以及如何组织，相对位置，绝对位置等信息。来看看其中内容：

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

```
14.  android:layout_height="fill_parent"
15.  >
16.  < TextView android:id="@+id/TextView01"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="MyTest OK yest!"><  /TextView>
17.  < /LinearLayout>
```

这里就描述了布局为 LinearLayout,包含了一个 TextView,TextView 的值为 MyTest.这个 XML 文件被编译后，可以使用 R.layout.main 的 ID 来从资源中取得。

于是 Activity 可以用 setContentView(R.layout.main)来直接从 Android 资源取得布局,来绘制界面元素。

另一类常用的 Android 资源就是图片在 res/drawable/下面有一些图片，你也可以新加一些图片到这里。然后就可以通过.R.drawable.xxx 的 ID 来从资源中取得对应的图片。

4. Android Activity类应用技巧

Android Activity 类在 Android 操作系统的应用中是常重要的。那么如何才能正确的在应用中来操作这一类呢？在一个应用中，每一个显示的屏幕都是一个 Activity。所以学习 Android，必须要对 Activity 有一定的了解。在其他论坛中也有一些关于 Activity 的介绍，我在这里就想谈谈我对 Activity 学习的一些看法。首先 Activity 的生命周期很重要，Activity 主要包含六个方法，分别是 onCreate，onStart，onResume，onPause，onStop，onDestory。

onCreate 和 onDestory 对应。onStart 和 onStop 对应，onResume 和 onPause 对应。

这几个函数大概是这样定义的，当启动一个 Android Activity 类的时候，onCreate 方法首先会被启动，然后接着是 onStart 和 onResume，也会启动，一般地，等这几个函数都启动完了之后你这个 Activity 就可以被显示出来了。当然我这里说的是一般的情况，如果你要是在这三个函数初始化的时候启动了一个后台的 Service，那么还要等待 ServiceConnection 执行完毕才能够被显示出来，这里可能有人要问什么是 Service 了，在以后我会介绍它，这里大家就先了解下 Service 的回调函数也会影响 Acitivity 的启动就可以了。

这是 Activity 启动时会调用的三个函数，在 Acitivity 销毁的时候会调用 onPause，onStop，onDestory。当调用完 onDestory 之后,你的 Acitivity 也就被销毁完毕了,这时候你在调用 Activity 的 isFinishing 的时候，就会返回 true，但是此时 Activity 的 this 指针还可以被使用，如果你在 Activity 单起一个线程做其他事情的话，那么上下文变量 context 指针还是能够被使用的。

当然 Android Activity 类有可能还处于其他状态，不一定就是被显示或者被销毁，很有可能这个 Activity 启动了另一个 Activity，这个时候前先的那个 Activiy 就会被放到系统的堆栈中，等被启动的 Activity 返回的时候，它又重新被显示出来，这个流程是这样的，一个 Activity 启动了另外的一个 Activity，那么它就会调用 onPause 函数，进入一种停滞的状态，然后被启动的 Activity 被销毁返回后，又会调用 onResume 函数。

对于 Activity 的这种机制，我感觉在初始化的工作最好放到一个自己定义的一个接口中，因为由于 Activity 状态的改变，你的 Activity 的 Layout 就有可能会被改变。说到这里就要谈一下什么叫做 Layout，每个 Activity 的界面的布局就是一个 Layout，每个 Activity 都要有这样一个布局它才能够被显示出来，一般地，我们都会把一个 Layout 放到一个 XML 文件当中，然后直接调用 Activity 的 setContentView 函数来填充这个 Activity，如果 Layout 不放在 xml 文件中，也可以用代码生成一个动态的 Layout，也就是说用 Activity。

this 指针生成一个 Layout。这个给大家推荐一个非常好用的工具叫做 droiddraw，论坛里就有链接，站长好像发过贴。这个工具非常好用，不用看教程半小时就能学会，上面有一些特定的控件，把控件摆好布局后直接能生成 xml 文件。把这个 xml 文件放到项目的 res/layout 文件夹下面就可以了。生成好 Layout 文件后，你就要为你程序要用到的一些控件设定 ID，具体怎么设定大家可以在 google 的 Android 主页里有，叫 gettingstarted，那个写得很明白，我就不跟这里重述了。

接下来还是谈下 Android 的这几个主要的函数，我还有些建议就是在 onCreate 函数中尽量少写代码，把尽肯能多的东西放到 onResume 和那个自定义的初始化函数里去写，onResume 这个函数被调用的几率是非常高的，这里大家在模拟器上开发可能没有感觉到 onResume 的重要性，在真机上测试就会发现，当屏幕变黑进入等待状态，然后你手动恢复屏幕变亮时也会进入 onResume 状态，所以我感觉把一些刷新控件的方法放到 onResume 函数中来做还是非常必要的。onPause，onStop，还有 onDestroy 函数都是用来做一些清理工作的，比如说一些变量要被释放，一些线程要被停滞等等都可以放到这里来做。

下面我来总结一下 Android Activity 类一个大概的设计思路:

首先设计一个方法，这个方法主要作用就是初始化 Activity 的控件，进行各种条件判断，对 Acitvity 来进行不用的布局初始化，这里举个例子来解释下为什么要初始化不同的布局，例如你的这个 Activity 从 SD 卡读取了一些信息，那么当你拔出 SD 卡的时候这些信息肯定也就没有了，那么你就要进行另外的一个布局来显示这个 Activity，这个时候你就可以重用这个函数来进行布局的初始化。

onCreate 函数:注册你要用到的变量，比如说 service，receiver，这些变量是无论你的 Activity 是在前台还是在后台都能够被响应到的，然后调用上面那个用来初始化的函数初始化布局信息。

onStart 函数:注册一些变量。这些变量必须在 Android Activity 类在前台的时候才能够被响应。

onResume 函数:调用一些刷新 UI 的函数，每当 Activity 调用到这里时就要刷新一下 UI 各控件的状态。

onPause 函数:一般是做一些变量的设置，因为这个时候 Activity 马上就要切到后台处理，可能有些变量就要被释放掉或者状态要做些相应的调整。

onStop 函数:反注册在 onStart 函数中注册的变量。

onDestory 函数:反注册在 onCreate 函数中注册的变量。

上面谈了些 Android Activity 类的最常用的一些方法，当然还有很多方法没有谈到，有很多方法我也没有用过，其他方法大家可以参考 google 的文档。

接下来我来谈谈 Activity 中最简单的一些通信方法，这里我先定义两个名字为方便接下来的叙述，启动另外一个 Activity 的那个 Activity 我们称之为主 Activity，被启动的那个 Activity 我们称之为子 Activity。

主 Activity 和子 Acitivity 通信的方式有很多种这里介绍两种最简单的方法。

方法一:通过 Intent 来进行参数的传递，在 Intent 中有各种 putXXX 方法来存放各种参数，然后在子 Activity 接收到这个 Intent 时能够从这个 Intent 里取出这个参数，利用 getIntnet()。getXXXExtra()方法就可以了。

方法二:当一个主 Activity 想从一个子 Activity 接受消息时可以使用 StartActivityforResult 方法，例如这样启动一个 Activity，startActivityForResult(i, REQUEST_CODE)；然后在主 Activity 中的 onActivityResult 方法对 requestCode 进行判断来对子 Android Activity 类不同的返回处理不同的情况，另外子 Activity 也可以利用 setResult 方法来设置主 Activity 方法中的 resultCode，这样主 Activity 也可以根据子 Activity 的不同的 resultCode 来处理不同的情况。

5. Android构建模块详细步骤

在这里，我们会分步骤详细为大家讲解一下 Android 构建模块的相关知识。

你可以认为一个 Android 的应用是不同种类的模块集合。这些模块大部分都是十分松散地联合到一起，联合到你可以准确地把它们描述为一个联合体的程度，而不是单一的粘合的应用。

一般地，这些模块大都运行在同一个系统进程。它可能或者非常普遍地在这个进程中创建多个线程，如果你需要，那么也有可能创建完整的独立子进程。这样的情况不是很常见，因为 Android 做了很大努力来让进程对你的代码透明。

这里是 Android 构建模块中最重要的部分:

AndroidManifest.xml

这个文件是一个控制文件，它来告诉系统你创建的顶层的模块都要干什么，这些模块包括 Activities，Services，Intent Receivers 和 Content Providers。例如，这实际上就是制定你的 Activity 能接收那个 Intent 的一种粘合剂。

Activities

基本上，Activity 是一个有生命周期的对象，是做一些工作的一块代码；如果需要的话，这个工作可以是包含显示 UI 给用户。当然，如果不必要，Activity 也可以不显示 UI。典型的说，你将制定你应用里的某个 Activity 当作你应用或者说程序的入口点。

Views

视图是知道如何把它自己画到屏幕上的对象。Android UI 是有视图树组成的。如果你想完成某些自定义的图像技术，比如你正在写一个游戏，或者正在构建一个不寻常的行的 UI Widget，那么你需要创建一个视图。

Intents

Intent 是代表要做某些事情或者某个意图的一个简单的消息对象。例如，你的程序想显示一个 WEB 页面，那么它想要浏览一个 URI 的意图，通过创建一个 Intent 实例并把它处理给系统来实现。系统来定位其他的代码（这种情况下，是浏览器），这段代码知道如何来处理这个 Intent 并运行它。Intent 也可以被用来广播有趣的事件给系统范围内（比如 Notificaiton）

Services

Service 是可以运行在后台的代码。它可以运行在自己的进程内，或者另一个程序进程的上下文 Context 中，这个依需要所决定。其他模块通过远程方法调用而绑定到某一个服务上。一个 Service 的例子就是媒体播放器；即使用户退出了媒体选择界面，它依然可以让它的音乐保持播放状态，但界面完成时，是 Service 来保持音乐继续播放的。

Notificaitons

一个 Notificaiton（为了不跟 Alarm 混淆我还是用英文）就是出现在状态栏上的一个小图标。用户可以和这个图标交互来获取信息。大家都知道的 Notificaiton 就是短信消息。呼啸历史和语

音邮件，但是应用程序可以创建它们自己的图标。Notificaiton 是用来提醒用户需要用户注意的最优机制。

Content Providers

Content Provide 是提供访问设备上数据的数据存储仓库；典型的例子就是 CP 用来访问用户联系人列表。你的程序可以范围别的程序通过 CP 暴露出来的数据。并且你也可以定义你自己的 CP 来暴露你自己的数据。

6. Android事件侦听器回调方法浅谈

Android 操作系统中，对于事件的处理是一个非常基础而且重要的操作。许多功能的实现都需要对相关事件进行触发才能达到自己的目的。比如 Android 事件侦听器是视图 View 类的接口，包含一个单独的回调方法。这些方法将在视图中注册的侦听器被用户界面操作触发时由 Android 框架调用。下面这些回调方法被包含在 Android 事件侦听器接口中：

onClick()

包含于 View.OnClickListener。当用户触摸这个 item（在触摸模式下），或者通过浏览键或跟踪球聚焦在这个 item 上，然后按下“确认”键或者按下跟踪球时被调用。

onLongClick()

包含于 View.OnLongClickListener。当用户触摸并控制住这个 item（在触摸模式下），或者通过浏览键或跟踪球聚焦在这个 item 上，然后保持按下“确认”键或者按下跟踪球（一秒钟）时被调用。

onFocusChange()

包含于 View.OnFocusChangeListener。当用户使用浏览键或跟踪球浏览进入或离开这个 item 时被调用。

onKey()

包含于 View.OnKeyListener。当用户聚焦在这个 item 上并按下或释放设备上的一个按键时被调用。

onTouch()

包含于 View.OnTouchListener。当用户执行的动作被当做一个触摸事件时被调用，包括按下，释放，或者屏幕上任何的移动手势（在这个 item 的边界内）。

onCreateContextMenu()

包含于 View.OnCreateContextMenuListener。当正在创建一个上下文菜单的时候被调用（作为持续的“长点击”动作的结果）。参阅创建菜单 Creating Menus 章节以获取更多信息。

这些方法是它们相应接口的唯一“住户”。要定义这些方法并处理你的事件，在你的活动中实现这个嵌套接口或定义它为一个匿名类。然后，传递你的实现的一个实例给各自的 View.set...Listener() 方法。（比如，调用 setOnClickListener()并传递给它你的 OnClickListener 实现。）

下面的例子说明了如何为一个按钮注册一个点击侦听器：

```
1.  // Create an anonymous implementation of OnClickListener
2.  private OnClickListener mCorkyListener = new OnClickListener() {
3.  public void onClick(View v) {
4.  // do something when the button is clicked
5.  }
6.  };
7.  protected void onCreate(Bundle savedValues) {
8.  ...
9.  // Capture our button from layout
10. Button button = (Button)findViewById(R.id.corky);
11. // Register the onClick listener with the implementation above
12. button.setOnClickListener(mCorkyListener);
13. ...
14. }
```

你可能会发现把 OnClickListener 作为活动的一部分来实现会便利的多。这将避免额外的类加载和对象分配。比如：

```
15. public class ExampleActivity extends Activity implements OnClickListener {

16. protected void onCreate(Bundle savedValues) {
17. ...
18. Button button = (Button)findViewById(R.id.corky);
19. button.setOnClickListener(this);
20. }
21. // Implement the OnClickListener callback
22. public void onClick(View v) {
23. // do something when the button is clicked
24. }
25. ...
```

```
26.  }
```

注意上面例子中的 onClick()回调没有返回值，但是一些其它 Android 事件侦听器必须返回一个布尔值。原因和事件相关。对于其中一些，原因如下：

• onLongClick() - 返回一个布尔值来指示你是否已经消费了这个事件而不应该再进一步处理它。也就是说，返回 true 表示你已经处理了这个事件而且到此为止；返回 false 表示你还没有处理它和/或这个事件应该继续交给其他 on-click 侦听器。

• onKey() -返回一个布尔值来指示你是否已经消费了这个事件而不应该再进一步处理它。也就是说，返回 true 表示你已经处理了这个事件而且到此为止；返回 false 表示你还没有处理它和/或这个事件应该继续交给其他 on-key 侦听器。

• onTouch() - 返回一个布尔值来指示你的侦听器是否已经消费了这个事件。重要的是这个事件可以有多个彼此跟随的动作。因此，如果当接收到向下动作事件时你返回 false，那表明你还没有消费这个事件而且对后续动作也不感兴趣。那么，你将不会被该事件中的其他动作调用，比如手势或最后出现向上动作事件。

记住按键事件总是递交给当前焦点所在的视图。它们从视图层次的顶层开始被分发，然后依次向下，直到到达恰当的目标。如果你的视图（或者一个子视图）当前拥有焦点，那么你可以看到事件经由 dispatchKeyEvent()方法分发。除了从你的视图截获按键事件，还有一个可选方案，你还可以在你的活动中使用 onKeyDown() and onKeyUp()来接收所有的事件。

注意: Android 将首先调用事件处理器，其次是类定义中合适的缺省处理器。这样，从这些事情侦听器中返回 true 将停止事件向其它 Android 事件侦听器传播并且也会阻塞视图中的缺事件处理器的回调函数。因此当你返回 true 时确认你希望终止这个事件。

3.4 用户人机界面

1. Android消息传递应用功能解析

Android手机操作系统中有一种叫做 Intent 的消息传递机制。这在实际编程中是一个核心技术，值得我们去深入的研究。在这里大家将会了解到有关 Android 消息传递的一些基础应用技巧，帮助大家理解。

每一个 Cursor、ContentResolver 做为一个小的注册中心，相关观察者可以在这个中心注册，更新消息由注册中心分发给各个观察者。而在 MFC 或 Winform 中，都会形成一个消息网，让消息在网中流动，被各节点使用、吃掉或者在出口死掉。

相比之下，我个人觉得基于 Intent 的 Android 消息传递机制是有所不同的。它应该会有一个全局性的注册中心，这个注册中心是隐性的，整个 Android 系统中就那么一个。所有的消息接收者，都被隐形的注册到这个中心。包括 Activity，Service 和 IntentReceiver。其实说隐形注册是不确切的，所有注册都还是我们手动告诉注册中心的，只是与传统的方式不一样，我们通常不是通过代码，而是通过配置文件来做。在应用的 Manifest 中，我们会为一些 Activity 或 Service 添加上 Intent-filter，或在配置文件中添加<receiver></receiver>项。这其实就相当于向系统的注册中心注册了相关的 Intent-filter 和 receiver（这个事情完全可以通过代码来做，只是这样就失去了修改的灵活性）。

当程序有一个消息希望发出去的时候，它需要将消息封装成一个 Intent，并发送。这时候，应该是有一个统一的中心（恩，有可能 Android 底层实现的时候不是，但简单这样看是没问题的...）接受到这个消息，并对它进行解析、判定消息类型（这个步骤降低了耦合...），然后检查注册了相匹配的 filter 或 receiver，并创建或唤醒接收者，将消息分发给它。这样做有很多好处。虽然这种传递有的时候不如点对点的传递快（这有些需要速度的地方，我们看到 Android 会通过直接通信来做），但有时候又因为它只经过一跳（姑且这么叫吧...），比复杂的流动又要更快。

更重要的是，它耦合性低，在手机平台这种程序组件多变的条件下使用十分适合。并且它可以很容易实现消息的精确或模糊匹配，弹性很大。（我个人曾想在开发一个 C++二次平台的时候引入这样的机制，但在 C++中，建立一套完整的数据 marshal 机制不容易，相比之下，用 java 来做会简单很多...）

恩，废话说了很多，具体讲讲 Android 消息传递中 Intent 的使用。当你有一个消息需要传递，如果你明确知道你需要哪个 Activity 或者其他 Class 来响应的话，你可以指定这个类来接受该消息，这被称为显性发送。你需要将 Intent 的 class 属性设置成目标。这种情况很常见，比如 startActivity 的时候，会清楚当前 Activity 完了应该是哪个 Activity，那就明确的发送这个消息。

但是，有的时候你并不确定你的消息是需要具体哪个类来执行，而只是知道接收者该符合哪些条件。比如你只需要有一个接收者能显示用户所选的数据，而不想制定某个具体的方法，这时候你就需要用到隐形发送（传统上，我们可能会考虑用多态，但显然这种方式更为灵活...）。

在 Android 中，你可以为 Intent 指定一个 action，表示你这个指令需要处理的事情。系统为我们定义了很多 Action 类型,这些类型使系统与我们通信的语言(比如在 Activity 里面加一个 Main

的 filter，该 activity 就会做成该应用的入口点），当然你也可以用于你自己的应用之间的通信（同样当然，也可以自定义...）。强烈建议，在自己程序接收或发出一个系统 action 的时候，要名副其实。比如你响应一个 view 动作，做的确实 edit 的勾当，你发送一个 pick 消息，其实你想让别人做 edit 的事，这样都会造成混乱。

当然只有 Action 有时候是不够的，在 Android 中我们还可以指定 catalog 信息和 type/data 信息，比如所有的显示数据的 Activity，可能都会响应 View action。但很多与我们需要显示的数据类型不一样，可以加一个 type 信息，明确的指出我们需要显示的数据类型，甚至还可以加上一个 catalog 信息，指明只有你只有按的是“中键”并发出这样的消息才响应。

从上面可以看出，Android 的 Intent 可以添加上 class, action, data/type, catalog 等消息，注册中心会根据这些信息帮你找到符合的接收者。其中 class 是点对点的指示，一旦指明，其他信息都被忽略。Intent 中还可以添加 key/value 的数据，发送方和接收方需要保持统一的 key 信息和 value 类型信息，这种数据的 marshal 在 java 里做，是不费什么力气的。

Android 消息传递的 Intent 发送，可以分成单播和广播两种。广播的接收者是所有注册了的符合条件的 IntentReceiver。在单播的情况下，即使有很多符合条件的接收者，也只要有一个出来处理这个消息就好（恩，个人看法，没找到确切条款或抉择的算法，本来想实验一下，没来得及...），这样的情况很容易理解，当你需要修改某个数据的时候，你肯定不会希望有十个编辑器轮流让你来处理。当广播不是这样，一个 receiver 没有办法阻止其他 receiver 进行对广播事件的处理。这种情况也很容易理解，比如时钟改变了，闹钟、备忘录等很多程序都需要分别进行处理。在自己的程序的使用中，应该分清楚区别，合理的使用。

2. 按钮的使用

1） 方法一

1.新建一个简单的android 工程

2.修改res/layout/main.xml文件内容.添加一个文本对象ViewText,两个按钮.

```
<?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/tv"
```

```
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="简单的按钮 demo"
        />

<Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="确定"
        />
<Button android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="退出"
        />
</LinearLayout>
```

2 编写主类代码.显示按钮并响应其点击事件，按下确定按钮时Activity类的Title的文字改变，按下退出按钮时，结束应用程序.

```
import android.app.Activity;
import android.view.View;
import android.view.View.OnClickListener;
import android.os.Bundle;
import android.widget.Button;
public class ButtonDemo extends Activity implements OnClickListener {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //从资源文件中获取按钮对象的引用
        Button sButton1 = (Button)findViewById(R.id.button);
        Button sButton2 = (Button)findViewById(R.id.button2);
        //注册点击事件监听者
        sButton1.setOnClickListener(this);
        sButton2.setOnClickListener(this);
    }
        /**
```

```
         * 按钮响应事件
         */
        public void onClick(View aView) {

                switch (aView.getId()) {
                        case R.id.button:
                                setTitle("this is OK button");
                                break;
                        case R.id.button2:
                                finish();
                                break;
                }
                }
}
```

2） 方法二

在onCreate中添加如下代码：

```
btn.setOnClickListener(listener);
btn2.setOnClickListener(listener2);
```

然后添加两个 OnClickListener：

```
OnClickListener listener = new OnClickListener() {
public void onClick(View v) {
setTitle("this is OK button");
}
};

OnClickListener listener2 = new OnClickListener() {
public void onClick(View v) {
  finish();
}
```

或者在 setOnClickListener直接加入OnClickListener：

```
btn2.setOnClickListener(new View.OnClickListener(){
      @Override
      public void onClick(View arg0) {
```

```
        // TODO Auto-generated method stub
              setTitle("this is cancle button!");
        //finish();
        }
});
```

注：定义 private final int WC = LinearLayout.LayoutParams.WRAP_CONTENT;
LinearLayout.LayoutParams 有两个参数：
FILL_PARENT,视图和其父亲一样大
WRAP_CONTENT,视图大到能包括其内容

3） Android Button应用法则

我们曾经在其他文章中为大家详细介绍了一些可视化编程以及基于界面的编程方法，那么今天大家将会了解到有关 Android Button 的具体应用，以此更进一步加深对界面编程的认知程度。

Android 界面编程有两种基本的方法，一种是在代码中，动态创建一个个组件，及把这些组件用 Layout 来进行组合成复杂的界面展现。一种是用图形化的方式来编写 布局 Layout，这些布局被保存在 XML 文件中，会编译成资源，被程序中的 Activity 来加载（setContentView()），再通过 findViewById 方式来获得每一个界面组件的引用进行操作。对于大多数人来说，喜欢最直观的方式，既代码中动态生成的方式。我们就先从这里说起，至于可视化编程及布局资源的方式以后专门来讲述。

一，Android Button 布局管理（Layout）

每一个界面组件都是 View 的子类，都可以单独占用一个屏幕，但是真正的有用的界面都是这些组件的组合，在 Android 中都是用各种 Layout 来进行布局管理，这与传统的 J2SE 中的一些 AWT,SWING 界面方式基本相同，这里就不多说。

二，Android Button 一个单独的界面元素：

在前面说到 Hello World 例子中，讲过这样一段代码。在 Activity 中.

```
o  public class HelloActivity extends Activity {
o  /** Called when the activity is first created. */
o  @Override
o  public void onCreate(Bundle savedInstanceState) {
o  super.onCreate(savedInstanceState);
o  TextView tv = new TextView(this);
o  tv.setText("Hello, World!");
o  this.setContentView(tv);
o  }
o  }
```

这里并没有用到 Layout,这就是单独的组件方式。也可以改为：

```
o  super.onCreate(savedInstanceState);
o  Button btn = new Button(this);
o  btn.setText("TestButton");
o  this.setContentView(btn);
```

编译运行，会有一个全屏的 Button，当然这不是你想要的实用的界面.那我们就用 Layout 来布局

```
o  super.onCreate(savedInstanceState);
o  Button btn = new Button(this);
o  btn.setText("TestButton");
o  Button btn2 = new Button(this);
o  btn2.setText("TestButton2");
o  LinearLayout layout = new LinearLayout(this);
o  layout.setOrientation(LinearLayout.VERTICAL);
o  layout.addView(btn);
o  layout.addView(btn2);
o  this.setContentView(layout);
```

编译运行，你就可以看到了两个上下排列的 Android Button，当然对于布局管理器的使用，做过 PC 上 AWT，SWING 的人都不陌生，这里就不赘述。

那如何响应事件呢： 大家猜一猜？想必大家不难猜到，在 AWT 中，在手机的 J2ME 中，都是用 Listener 来处理事件响应，Android 也未能脱俗。这与 Blackberry,Symbian 中的 Observer 是同一个道理。都是使用了设计模式的观察者模式。下面来看一个能响应事件的例子。

```
o  import android.app.Activity;
o  import android.os.Bundle;
o  import android.view.View;
o  import android.view.View.OnClickListener;
o  import android.widget.Button;
o  import android.widget.LinearLayout;
o  public class HelloActivity extends Activity implements OnClickListener
    {
o  Button btn = null;
o  Button btn2 = null;
o  public void onClick(View v) {
o  if (v == btn)
o  {
```

```
o   this.setTitle("You Clicked Button1");
o   }
o   if (v == btn2)
o   {
o   this.setTitle("You Clicked Button2");
o   }
o   }
o   @Override
o   public void onCreate(Bundle savedInstanceState) {
o   super.onCreate(savedInstanceState);
o   btn = new Button(this);
o   btn2 = new Button(this);
o   btn.setText("TestButton1");
o   btn2.setText("TestButton2");
o   btn.setOnClickListener(this);
o   btn2.setOnClickListener(this);
o   LinearLayout layout = new LinearLayout(this);
o   layout.setOrientation(LinearLayout.VERTICAL);
o   layout.addView(btn);
o   layout.addView(btn2);
o   this.setContentView(layout);
o   }
o   }
```

Android Button 操作步骤是：

一，生成两个 Button，配置 Click 事件监听者为 HelloActivity，此类实现了 OnClickListener 接口。

二，放入布局，按布局显示两个 Button

三，按下其中一个 Button,生成 Click 事件，调用 HelloActivity 的 OnClick 接口函数。

四，对于 View 参数的值，判断是哪个 View(Button)。改写 Activity 的 Titile 内容。注意，可别去对比 View.getId()，缺省情况下，每个组件的 Id 值都为-1,除非人为设定 Id 值，用可视化编程时，为自动为其生成一个 Id 值。

3. 通知对话框(AlertDialog)的使用

这里介绍通知对话框 android.app.AlertDialog 类的使用

- AlertDialog 实例的生成，
  1.生成 AlertDialog.Builder 类的实例，并且设置属性。
  2.调用 AlertDialog.Builder.create()方法，生成 AlertDialog 的实例。

- AlertDialog 的表示，调用 AlertDialog.show()方法。
- AlertDialog 标题的设置，调用 AlertDialog.Builder.setTitle()方法。
- AlertDialog 消息的设置，调用 AlertDialog.Builder.setMessage()方法。
- AlertDialog 按钮点击处理的追加，调用 AlertDialog.Builder.setTitle()方法。
- 调用 AlertDialog.Builder 类的 setPositiveButton()、setNeutralButton()、setNegativeButton()方法，把 DialogInterface.OnClickListener 实例作为参数传递过去

- AlertDialog 取消按钮的设置，调用 AlertDialog.Builder.setCancelable()方法。

1） 一、AlertDialog.Builder

Android中的alertDialog的创建一般是通过其内嵌类AlertDialog.Builder来实现的。所以首先浏览一下这个builder所提供的方法：

setTitle()：给对话框设置title.

setIcon()：给对话框设置图标。

setMessage()：设置对话框的提示信息

setItems()：设置对话框要显示的一个list，一般用于要显示几个命令时

setSingleChoiceItems()：设置对话框显示一个单选的List

setMultiChoiceItems()：用来设置对话框显示一系列的复选框。

setPositiveButton()：给对话框添加"Yes"按钮。

setNegativeButton()：给对话框添加"No"按钮。

2） 二、常见对话框：

在了解完这几个常用的方法之后，看一个小例子，创建一个用来提示的对话框：

```
Dialog dialog = new AlertDialog.Builder(AlertDialogSamples.this)
                                .setIcon(R.drawable.alert_dialog_icon)
                                .setTitle(“title”)
                                .setMessage(“这里是提示信息语句”)
                                .setPositiveButton(“Ok”, new
DialogInterface.OnClickListener() {
                                        public void
onClick(DialogInterface dialog, int whichButton) {

                                                /* User clicked OK so do
some stuff */
                                        }
                                })
                                .setNeutralButton(“Cancel”, new
DialogInterface.OnClickListener() {
                                        public void
onClick(DialogInterface dialog, int whichButton) {

                                                /* User clicked
Something so do some stuff */
                                        }
                                })
                                .setNegativeButton(R.string.alert_dial
og_cancel, new DialogInterface.OnClickListener() {
                                        public void
onClick(DialogInterface dialog, int whichButton) {

                                                /* User clicked Cancel
so do some stuff */
                                        }
                                })
                                .create();
dialog.show();//如果要显示对话框，一定要加上这句
```

另外在我的以前的一篇博客中的代码中介绍了如何创建一个包含single choice或者command list的对话框，具体请参考这里：http://blog.chinaunix.net/u/20947/showart_1962223.html

3） 三、包含定制view的对话框：

很多时候，我们需要在对话框中显示一个特定的view,比如说用户登录对话框，就需要显示要用户输入用户名和密码的editBox等。

要想让对话框显示一个view,我们就要用到AlertDialog.Builder的setView(View view)方法来，可以看到该方

法的参数是一个view,所以我们就需要先取得这个view,这个时候我们还要用到一个新的class,那就是LayoutFlater，它的作用就是将一个layout 的xml文件转化成一个view实例。

4） 四、ABOUT的对话框：

```
private void openOptionsDialog()
{
  new AlertDialog.Builder(this)
  .setTitle(R.string.app_about)
  .setMessage(R.string.app_about_msg)
  .setPositiveButton(R.string.str_ok,
     new DialogInterface.OnClickListener()
     {
      public void onClick(DialogInterface dialoginterface, int i)
      {
      }
      }
     )
  .show();
 }
```

4. Android列表框应用技巧讲解

Android列表框中的一些相关设置将会在这篇文章中为大家详细介绍。希望本文推出的内容可以在一定程度上解决大家的一些实际问题。

1） Listview用法一

ArrayAdapter 它接受一个数组或者 List 作为参数来构建。

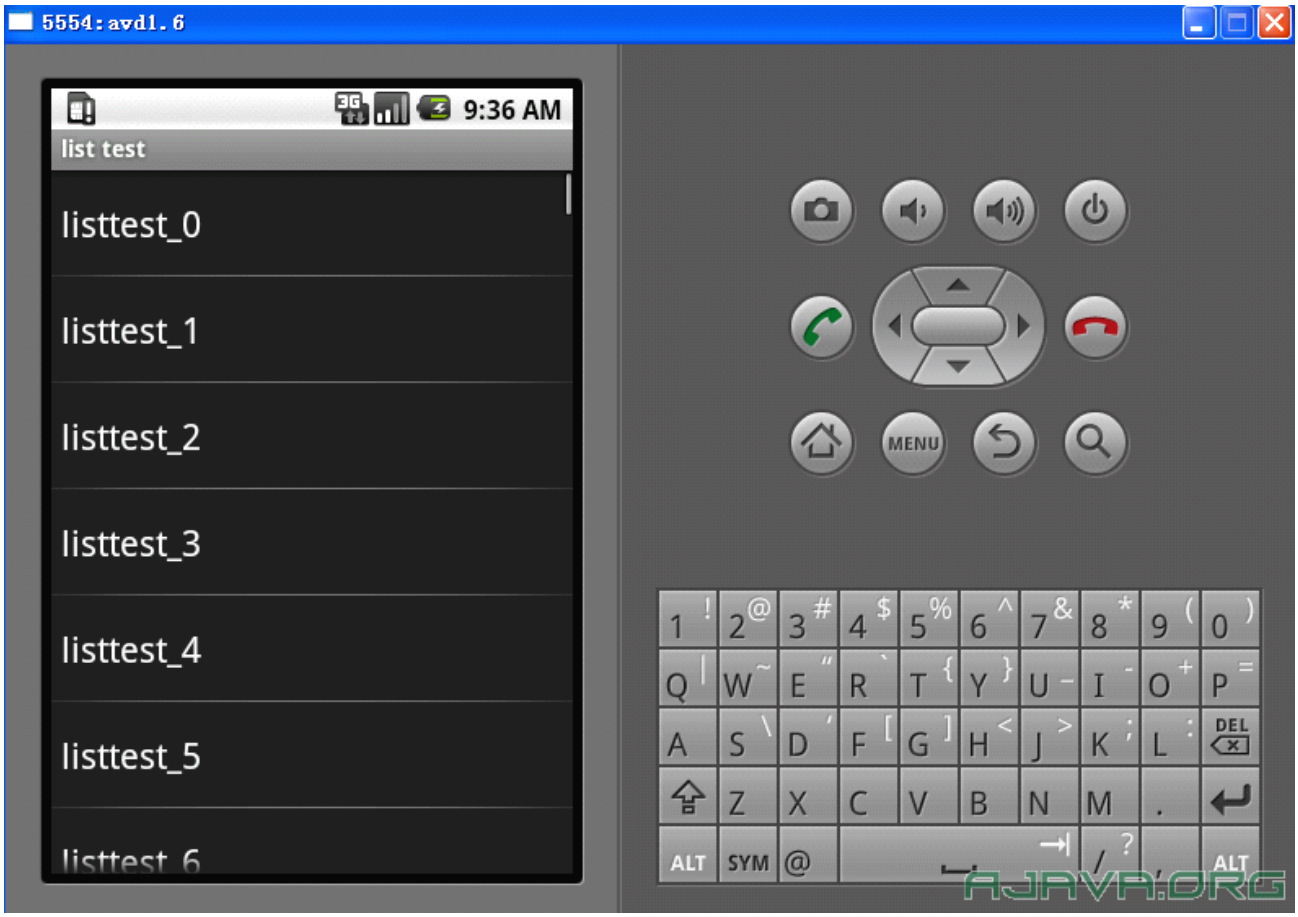
一下通过简单例子说明：

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.layou
t.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我们不需要加载我们自己的layout 而是用系统已经实现的layout很快速的实现了listview

2） ListView用法二

在这里我们先来了解一下 Android 列表框的一些应用技巧，以加深对这方面的认知程度。

- 设置 Adapter，调用 setAdapter()方法。
- 追加 Item 被点击时候的处理，调用 setOnItemClickListener()方法。
- 追加 Item 被选择时候的处理，调用 setOnItemSelectedListener()方法。

Android 列表框例程源码(Java)

```
o   ArrayAdapter< String> adapter = new ArrayAdapter< String>(
o       this,
o       android.R.layout.simple_list_item_1);
o       adapter.add("red");
o       adapter.add("green");
```

```
o       adapter.add("blue");
o       ListView listView = (ListView) findViewById(id.listView);
o       listView.setAdapter(adapter);
o       listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
o       @Override
o       public void onItemClick(
o       AdapterView< ?> parent,
o       View view,
o       int position,
o       long id) {
o       ListView listView = (ListView) parent;
o       Log.v("Test", "id = " + id + "("
o       + listView.getItemAtPosition(position).toString() + ")");
o       }
o       });
o       listView.setOnItemSelectedListener(new OnItemSelectedListener() {
o       @Override
o       public void onItemSelected(
o       AdapterView< ?> parent,
o       View view,
o       int position,
o       long id) {
o       ListView listView = (ListView) parent;
o       Log.v("Test", "id = " + id + "("
o       + listView.getSelectedItem().toString() + ")");
o       }
o       @Override
o       public void onNothingSelected(AdapterView< ?> parent) {
o       }
o       });
```

Android 列表框例程源码(Resource)

```
o   < ListView android:id="@+id/listView"
o       android:layout_width="fill_parent"
o       android:layout_height="fill_parent" />
```

3）　ListView用法三

ListView 是一个经常用到的控件，ListView 里面的每个子项 Item 可以使一个字符串，也可以是一个组合控件。先说说 ListView 的实现：

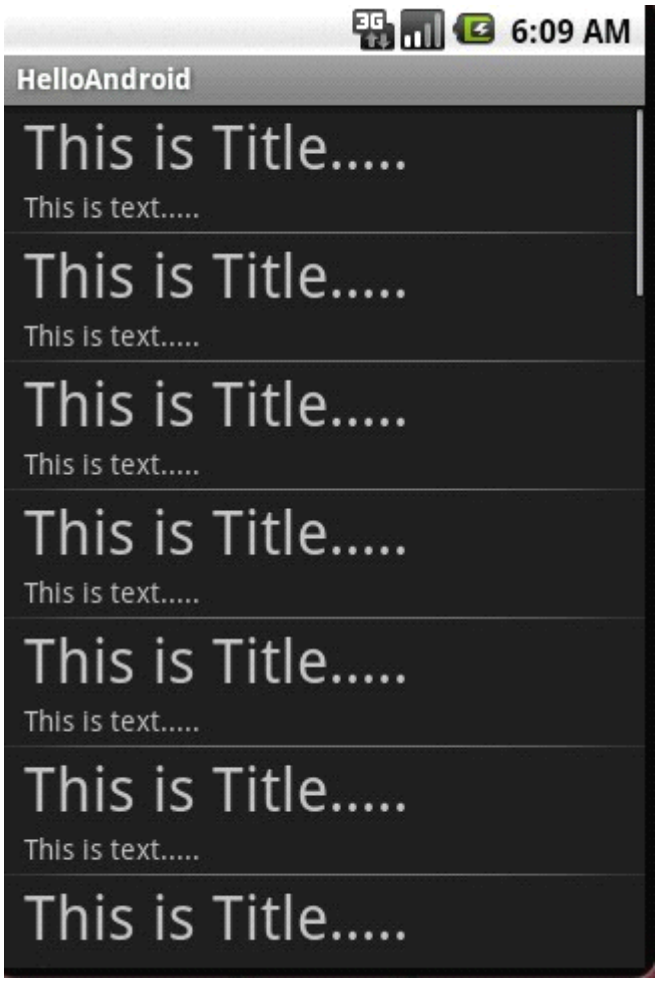
1.准备 ListView 要显示的**数据**　；

2.使用　一维或多维　**动态数组**　保存数据；

2.构建**适配器**　，　简单地来说，　适配器就是　**Item 数组**　，　**动态数组**　有多少元素就生成多少个 Item；

3.把　**适配器**　添加到 ListView，并显示出来。

接下来，看看本文代码所实现的 ListView：



接下来，就开始 UI 的 XML 代码：

main.xml 代码如下，很简单，也不需要多做解释了：

view plaincopy to clipboardprint?

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
        android:id="@+id/LinearLayout01"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        xmlns:android="http://schemas.android.com/apk/res/android">

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

my_listitem.xml 的代码如下，my_listitem.xml 用于设计 ListView 的 Item：

view plaincopy to clipboardprint?

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
        android:layout_width="fill_parent"
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_height="wrap_content"
        android:id="@+id/MyListItem"
        android:paddingBottom="3dip"
        android:paddingLeft="10dip">
        <TextView
                android:layout_height="wrap_content"
                android:layout_width="fill_parent"
                android:id="@+id/ItemTitle"
                android:textSize="30dip">
        </TextView>
        <TextView
                android:layout_height="wrap_content"
                android:layout_width="fill_parent"
                android:id="@+id/ItemText">
        </TextView>
</LinearLayout>
```

解释一下，里面用到的一些属性：

1. paddingBottom="3dip"，Layout 往底部留出 3 个像素的空白区域

2. paddingLeft="10dip"，Layout 往左边留出 10 个像素的空白区域

3. textSize="30dip"，TextView 的字体为 30 个像素那么大。

最后就是 JAVA 的源代码：

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

```
3.      setContentView(R.layout.main);
4.      //绑定 XML 中的 ListView，作为 Item 的容器
5.      ListView list = (ListView) findViewById(R.id.MyListView);
6.
7.      //生成动态数组，并且转载数据
8.      ArrayList<HashMap<String, String>> mylist = new ArrayList<HashMap<String, Strin
    g>>();
9.      for(int i=0;i<30;i++)
10.     {
11.         HashMap<String, String> map = new HashMap<String, String>();
12.         map.put("ItemTitle", "This is Title.....");
13.         map.put("ItemText", "This is text.....");
14.         mylist.add(map);
15.     }
16.     //生成适配器，数组===》ListItem
17.     SimpleAdapter mSchedule = new SimpleAdapter(this, //没什么解释
18.                                                 mylist,//数据来源
19.                                                 R.layout.my_listitem,//ListItem 的
    XML 实现
20.
21.                                                 //动态数组与 ListItem 对应的子
    项
22.                                                 new String[] {"ItemTitle", "ItemTex
    t"},
23.
24.                                                 //ListItem 的 XML 文件里面的两个
    TextView ID
25.                                                 new int[] {R.id.ItemTitle,R.id.Item
    Text});
26.     //添加并且显示
27.     list.setAdapter(mSchedule);
28. }
```

4） ListView用法四

这篇接下来也是围绕 ListView 和 Item，更加深入地介绍它们的用法。

首先，先来看看本文代码运行的结果，本文的 Item 比上一篇中的 Item 多出左边的图标：

main.xml 的源代码，跟上一篇的一样，这里就不作解释了，直接贴出 my_imageitem.xml 的代码，就是它实现 ImageItem 的 UI：

view plaincopy to clipboardprint?

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <RelativeLayout
3.          android:id="@+id/RelativeLayout01"
4.          android:layout_width="fill_parent"
5.          xmlns:android="http://schemas.android.com/apk/res/android"
6.          android:layout_height="wrap_content"
7.          android:paddingBottom="4dip"
8.          android:paddingLeft="12dip">
9.          <ImageView
10.               android:layout_width="wrap_content"
11.               android:layout_height="wrap_content"
12.               android:id="@+id/ItemImage">
13.         </ImageView>
14.         <TextView
15.               android:text="TextView01"
16.               android:layout_height="wrap_content"
17.               android:textSize="30dip"
18.               android:layout_width="fill_parent"
19.               android:layout_toRightOf="@+id/ItemImage"
20.               android:id="@+id/ItemTitle">
21.         </TextView>
```

```
        <TextView
              android:text="TextView02"
              android:layout_height="wrap_content"
              android:layout_width="fill_parent"
              android:layout_toRightOf="@+id/ItemImage"
              android:layout_below="@+id/ItemTitle"
              android:id="@+id/ItemText">
        </TextView>
</RelativeLayout>
```

解释一下 my_imageitem.xml 的代码：这里使用了 RelativeLayout 布局，控件的关键的属性是：

ItemTitle 的属性 android:layout_toRightOf="@+id/ItemImage" ，ItemTitle 在 ItemImage 的右边；

ItemText 的属性 android:layout_toRightOf="@+id/ItemImage"，ItemText 在 ItemImage 的右边，android:layout_below="@+id/ItemTitle"， ItemText 在 ItemTitle 的下面。

最后，贴出 JAVA 的源代码，这里的源代码跟上一篇的很类似，只是修改了一部分，引入 Item Image：

view plaincopy to clipboardprint?

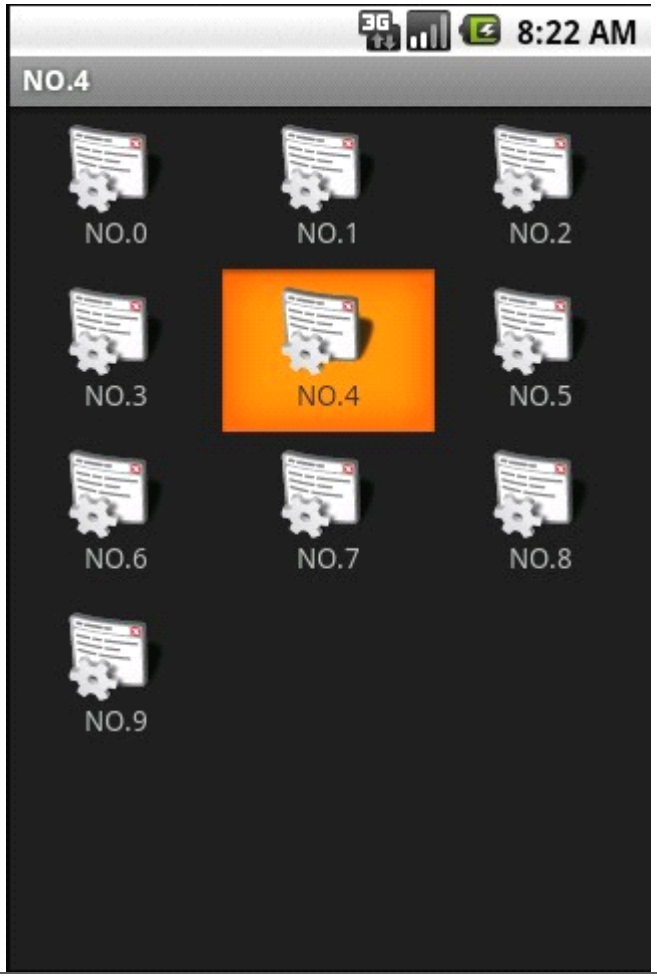
```
@Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //绑定 XML 中的 ListView，作为 Item 的容器
        ListView list = (ListView) findViewById(R.id.MyListView);

        //生成动态数组，并且转载数据
        ArrayList<HashMap<String, Object>> lstImageItem = new ArrayList<HashMap<Stri
ng, Object>>();
        for(int i=0;i<10;i++)
        {
         HashMap<String, Object> map = new HashMap<String, Object>();
         map.put("ItemImage", R.drawable.icon);//添加图像资源的 ID
         map.put("ItemTitle", "This is Title.....");
         map.put("ItemText", "This is text.....");
         lstImageItem.add(map);
        }
        //生成适配器的 ImageItem <====> 动态数组的元素，两者一一对应
        SimpleAdapter saImageItems = new SimpleAdapter(this, //没什么解释
                                                       lstImageItem,//数据来源
                                                       R.layout.my_imageitem,//ListItem 的
XML 实现

                                                       //动态数组与 ImageItem 对应的子
项
```

```
                                                    new String[] {"ItemImage","ItemTitl
    e", "ItemText"},
                                                    //ImageItem 的 XML 文件里面的一个
    ImageView,两个 TextView ID
                                                    new int[] {R.id.ItemImage,R.id.Item
    Title,R.id.ItemText});
            //添加并且显示
            list.setAdapter(saImageItems);
        }
```

2. GridView(九宫图)

GridView 跟 ListView 都是比较常用的多控件布局，而 GridView 更是实现九宫图的首选!本文就是介绍如何使用 GridView 实现九宫图。GridView 的用法很多，网上介绍最多的方法就是自己实现一个 ImageAdapter 继承 BaseAdapter，再供 GridView 使用，类似这种的方法本文不再重复，本文介绍的 GridView 用法跟前文 ListView 的极其类似。。。。也算是我偷懒一下，嘻嘻嘻嘻。。。。

先来贴出本文代码运行的结果：



本文需要添加/修改 3 个文件：main.xml、night_item.xml、JAVA 源代码。

main.xml 源代码如下，本身是个 GirdView，用于装载 Item：

view plaincopy to clipboardprint?

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <GridView xmlns:android="http://schemas.android.com/apk/res/android"
3.      android:id="@+id/gridview"
4.      android:layout_width="fill_parent"
5.      android:layout_height="fill_parent"
6.      android:numColumns="auto_fit"
7.      android:verticalSpacing="10dp"
8.      android:horizontalSpacing="10dp"
9.      android:columnWidth="90dp"
10.     android:stretchMode="columnWidth"
11.     android:gravity="center"
12. />
```

介绍一下里面的某些属性：

android:numColumns="auto_fit" ，GridView 的列数设置为自动

android:columnWidth="90dp"，每列的宽度，也就是 Item 的宽度
android:stretchMode="columnWidth"，缩放与列宽大小同步
android:verticalSpacing="10dp"，两行之间的边距，如：行一(NO.0~NO.2)与行二(NO.3~NO.5)间距为 10dp
android:horizontalSpacing="10dp"，两列之间的边距。

接下来介绍 night_item.xml，这个 XML 跟前面 ListView 的 ImageItem.xml 很类似：

view plaincopy to clipboardprint?

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <RelativeLayout
3.          xmlns:android="http://schemas.android.com/apk/res/android"
4.          android:layout_height="wrap_content"
5.          android:paddingBottom="4dip" android:layout_width="fill_parent">
6.          <ImageView
7.                  android:layout_height="wrap_content"
8.                  android:id="@+id/ItemImage"
9.                  android:layout_width="wrap_content"
10.                 android:layout_centerHorizontal="true">
11.         </ImageView>
12.         <TextView
13.                 android:layout_width="wrap_content"
14.                 android:layout_below="@+id/ItemImage"
15.                 android:layout_height="wrap_content"
16.                 android:text="TextView01"
17.                 android:layout_centerHorizontal="true"
```

```
                android:id="@+id/ItemText">
            </TextView>
</RelativeLayout>
```

最后就是 JAVA 的源代码了，也跟前面的 ListView 的 JAVA 源代码很类似，不过多了"选中"的事件处理：

view plaincopy to clipboardprint?

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        GridView gridview = (GridView) findViewById(R.id.gridview);

        //生成动态数组，并且转入数据
        ArrayList<HashMap<String, Object>> lstImageItem = new ArrayList<HashMap<Strin
    g, Object>>();
        for(int i=0;i<10;i++)
        {
          HashMap<String, Object> map = new HashMap<String, Object>();
          map.put("ItemImage", R.drawable.icon);//添加图像资源的 ID
       map.put("ItemText", "NO."+String.valueOf(i));//按序号做 ItemText
          lstImageItem.add(map);
        }
        //生成适配器的 ImageItem <====> 动态数组的元素，两者一一对应
        SimpleAdapter saImageItems = new SimpleAdapter(this, //没什么解释
                                                       lstImageItem,//数据来源
                                                       R.layout.night_item,//night_item 的
    XML 实现

                                                       //动态数组与 ImageItem 对应的子
    项
                                                       new String[] {"ItemImage","ItemText
    "},

                                                       //ImageItem 的 XML 文件里面的一个
    ImageView,两个 TextView ID
                                                       new int[] {R.id.ItemImage,R.id.Item
    Text});
        //添加并且显示
        gridview.setAdapter(saImageItems);
        //添加消息处理
        gridview.setOnItemClickListener(new ItemClickListener());
    }

    //当 AdapterView 被单击(触摸屏或者键盘)，则返回的 Item 单击事件
    class  ItemClickListener implements OnItemClickListener
    {
public void onItemClick(AdapterView<?> arg0,//The AdapterView where the click happe
    ned
                                    View arg1,//The view within the AdapterView that
    was clicked
                                    int arg2,//The position of the view in the adapte
    r
                                    long arg3//The row id of the item that was clicke
    d
                                    ) {
      //在本例中 arg2=arg3
      HashMap<String, Object> item=(HashMap<String, Object>) arg0.getItemAtPosition(a
    rg2);
```

```
41.     //显示所选 Item 的 ItemText
42.     setTitle((String)item.get("ItemText"));
43. }
44.
45.   }
```

3. Android键盘操作

对手机有所了解的朋友可能知道，Android 手机操作系统是这一领域中占据着重要地位的系统。下面我们可以通过对 Android 键盘操作的相关介绍，来对这一系统的应用方式以及功能有一个初步的了解。

在 Android 中是通过触屏及键盘来操作程序的，我们如何响应一般的键盘及触笔动作呢？通过对 Android 一些基本界面元素的操作的了解，如果你再熟悉 MVC 你可以猜到 Android 将会如何处理键盘事件，恭喜你，猜对了，仍在 Activity 中改事件响应函数来做到。

一般是如下三个 Android 键盘操作的函数：

onKeyDown, onKeyUp, on, onKeyMultiple

参看如下 Android 键盘操作代码：

```
o   import android.app.Activity;
o   import android.app.ProgressDialog;
o   import android.os.Bundle;
o   import android.view.KeyEvent;
o   import android.view.Menu;
o   import android.view.MenuItem;
o   import android.widget.TextView;
o   public class TestProgress extends Activity {
o   private ProgressDialog progress = null;
o   @Override
o   public void onCreate(Bundle savedInstanceState) {
o   super.onCreate(savedInstanceState);
o   setContentView(R.layout.main);
o   }
o   @Override
o   public boolean onCreateOptionsMenu(Menu menu) {
o   super.onCreateOptionsMenu(menu);
o   menu.add(0, Menu.FIRST+1, 1, "Open Progress");
o   menu.add(0, Menu.FIRST+2, 2, "Exit");
o   return true;
```

```
o  }
o  @Override
o  public boolean onOptionsItemSelected(MenuItem item) {
o  super.onOptionsItemSelected(item);
o  switch (item.getItemId())
o  {
o  case Menu.FIRST +1:
o  {
o  progress = new ProgressDialog(this);
o  progress.setTitle("Progress!!");
o  progress.setMessage("Please wait for the operation...");
o  progress.setCancelable(true);
o  progress.show();
o  //progress = ProgressDialog.show(this, "Progress!",
   "Please wait for operation...");
o  break;
o  }
o  case Menu.FIRST +2:
o  {
o  finish();
o  break;
o  }
o  }
o  return true;
o  }
o  @Override
o  public boolean onKeyDown(int keyCode, KeyEvent event) {
o  // TODO Auto-generated method stub
o  super.onKeyDown(keyCode, event);
o  setTitle("you pressed key:" + String.valueOf(keyCode));
o  return true;
o  }
o  @Override
o  public boolean onKeyMultiple(int keyCode, int repeatCount,
   KeyEvent event) {
o  // TODO Auto-generated method stub
o  super.onKeyMultiple(keyCode, repeatCount, event);
o  TextView tv = (TextView)this.findViewById(R.id.mainview);
```

```
o   tv.setText("you have press key:[" + String.valueOf(keyCode) + "]
    for:" + String.valueOf(repeatCount) + "Times!");
o   return true;
o   }
o   @Override
o   public boolean onKeyUp(int keyCode, KeyEvent event) {
o   // TODO Auto-generated method stub
o   super.onKeyUp(keyCode, event);
o   setTitle("you release key:" + String.valueOf(keyCode));
o   return true;
o   }
o   }
```

Android 键盘操作的相关应用就为大家介绍到这里。

4. Android进度条

在这篇文章中，我们会针对Android进度条的相应方法进行详细的介绍，以方便大家对此的了解。并能在实际应用中对此进行正确的设置，以满足我们的需求。

Android 手机操作系统中有很多功能在实际应用中体现了非常大的作用。比如在这里为大家介绍的 Android 进度条的实现，就可以帮助大家解决一些相应的问题。那么大家就一起来看看这方面的相关方法吧。

●进度条分不确定（indeterminate=true）和确定（indeterminate=false）2 种。

默认值是不确定（indeterminate=true）Android 进度条。

●Android 进度条有 4 种风格可以使用。

默认值是 progressBarStyle。

设置成 progressBarStyleSmall 后，图标变小。

设置成 progressBarStyleLarge 后，图标变大

设置成 progressBarStyleHorizontal 后，变成横向长方形。

●确定（indeterminate=false）进度条中的最大值的设定，调用 setMax()方法。

●Android 进度条中当前进度值的设置，调用 setProgress()方法。

●第 2 进度值的设置，调用 setSecondaryProgress()方法。

例程源码(Java)

```
ProgressBar progressBarHorizontal =
(ProgressBar) findViewById(id.progressBarHorizontal);
progressBarHorizontal.setMax(100);
progressBarHorizontal.setProgress(30);
progressBarHorizontal.setSecondaryProgress(70);
```

Android 进度条例程源码(Resource)

```
< ProgressBar android:id="@+id/progressBarHorizontal"
android:layout_height="wrap_content"
android:layout_width="fill_parent"
style="?android:attr/progressBarStyleHorizontal" />
```

5. 菜单的使用

android提供了三种菜单类型，分别为**options menu，context menu，sub menu**。

options menu就是通过按home键来显示，context menu需要在view上按上2s后显示。这两种menu都有可以加入子菜单，子菜单不能种不能嵌套子菜单。

options menu最多只能在屏幕最下面显示6个菜单选项，成为icon menu，icon menu不能有checkable选项。多余6的会以more icon menu来调出，成为expanded menu。options menu通过activity的**onCreateOptionsMenu**来生成，这个函数只会在menu第一次生成时调用。任何想改变options menu的想法只能在**onPrepareOptionsMenu**来实现，这个函数会在menu显示前调用。**onOptionsItemSelected** 处理选中的菜单项。

context menu是跟某个具体的view绑定在一起，在activity种用registerForContextMenu来为某个view注册context menu。context menu在显示前都会调用onCreateContextMenu来生成menu。onContextItemSelected选中的菜单项。

android还提供了对菜单项进行分组的功能，可以把相似功能的菜单项分成同一个组，这样就可以通过调用setGroupCheckable，setGroupEnabled,setGroupVisible来设置菜单属性，而无须单独设置。

1） option Menu

在此介绍最简单的菜单应用代码如下：

```
public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    menu.add(0,SAVE,0,"保存");
    menu.add(0,RETURN_MENU,1,"返回主菜单");
    menu.add(0,QUIT,2,"退出");
    return super.onCreateOptionsMenu(menu);

}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    TextView tv = (TextView)findViewById(R.id.tv01);
    switch(item.getItemId()){
        case SAVE:tv.setText("保存按钮被点击");break;
        case RETURN_MENU:tv.setText("返回主菜单按钮被点击")
        case QUIT:tv.setText("退出按钮被点击");break;
    }
    return super.onOptionsItemSelected(item);
}
```

运行后，需要点击下方圆圈状菜单按钮，，即可出现菜单，对点击进行响应。如图所示：

当菜单超过三个时，会出现more的选项：

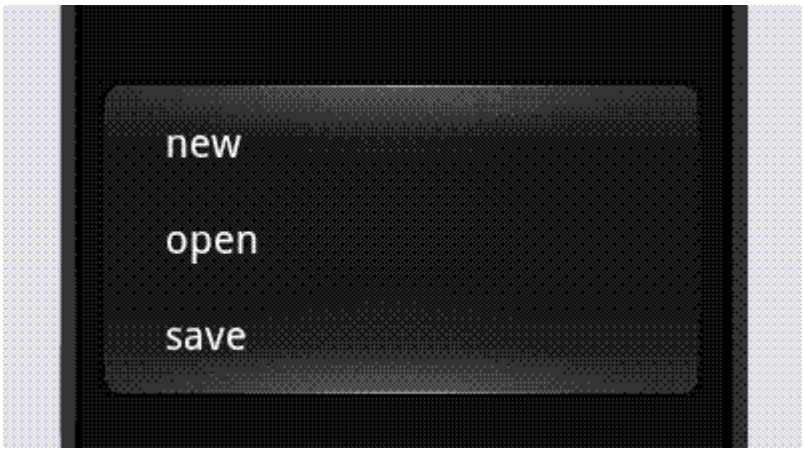
2） sub Menu

将onCreateOptionsMenu内代码改成如下代码，即可实现submenu：

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
        SubMenu fileMenu = menu.addSubMenu("File");
        SubMenu editMenu = menu.addSubMenu("Edit");
        fileMenu.add("new");
        fileMenu.add("open");
        fileMenu.add("save");
        editMenu.add("undo");
        editMenu.add("redo");


    return super.onCreateOptionsMenu(menu);

}
```

运行后可以出现两级菜单：

在点击file之后，出现：



3） Android菜单构造技巧

今天为大家介绍的Android菜单的创建方法主要是通过xml文件来实现的。我们将通过对相关编程代码的解读来充分了解这一应用技巧。

在 Android 手机操作系统中，可以用很多方法来实现一个相同的功能。这就取决于编程人员的个人爱好以及所适用的环境等等。比如 Android 菜单的创建，就可以用诸如动态等多种方法来实现。

今天学习如何通过 xml 文件的方法来构造一个 Android 菜单。首先，在 res 下建一个 menu 文件夹，在此文件夹下建一个 menu.xml，内容为下：

```
1.  < ?xml version="1.0" encoding="utf-8"?>
2.  < menu xmlns:android="http://schemas.android.com/apk/res/android">
3.  < item android:id="@+id/settings"
4.  android:title="@string/settings_label"
5.  android:alphabeticShortcut="@string/settings_shortcut" />
6.  //更多的项在此添加
7.  < /menu>
```

然后在 res/values 下的 strings.xml 添加如下字符串资源：

```
8.  < string name="settings_label">Settings...< /string>
9.  < string name="settings_title">Sudoku settings< /string>
10.  < string name="settings_shortcut">s< /string>
11.  < string name="music_title">Music< /string>
12.  < string name="music_summary">Play background music< /string>
13.  < string name="hints_title">Hints< /string>
14.  < string name="hints_summary">Show hints during play< /string>
```

还是和上篇一样，在 activity 类重写基类的 onCreateOptionsMenu 事件，添加如下代码：

```
15.  @Override
```

```
16.  public boolean onCreateOptionsMenu(Menu menu) {
17.  super.onCreateOptionsMenu(menu);
18.  MenuInflater inflater = getMenuInflater();
19.  inflater.inflate(R.menu.menu, menu);
20.  return true;
21.  }
```

这样，一个 Android 菜单已经建好了，注意，这里用到了 MenuInflater 类，使用该类的 inflate 方法来读取 xml 文件并且建立菜单。注意该 xml 菜单只有一项，如果需要更多的项可以在后面继续添加。

接着，就是实现各个菜单项的事件了。在 activity 类重写基类的 onOptionsItemSelected 方法：

```
22.  @Override
23.  public boolean onOptionsItemSelected(MenuItem item) {
24.  switch (item.getItemId()) {
25.  case R.id.settings:
26.  startActivity(new Intent(this, Settings.class));
27.  return true;
28.  // More items go here (if any) ...
29.  }
30.  return false;
31.  }
```

在这个响应的 Android 菜单的事件里，我们建立一个新的 activity。该 activity 是通过类 Settings 来呈现的。我们知道如果要呈现一个 activity 有两种方法：(1)通过代码布局来实现(2)通过 xml 文件来实现。每个方法都有优缺点，在这里我们通过 xml 文件呈现 view。步骤如下：

(1) 首先在 res 文件下建立一个 xml 文件夹，在 xml 文件夹下建立 Settings.xml 文件。Settings.xml 文件如下：

```
32.  < ?xml version="1.0" encoding="utf-8"?>
33.  < PreferenceScreen
34.  xmlns:android="http://schemas.android.com/apk/res/android">
35.  < CheckBoxPreference
36.  android:key="music"
37.  android:title="@string/music_title"
38.  android:summary="@string/music_summary"
39.  android:defaultValue="true" />
```

```
40.  < CheckBoxPreference
41.  android:key="hints"
42.  android:title="@string/hints_title"
43.  android:summary="@string/hints_summary"
44.  android:defaultValue="true" />
45.  < /PreferenceScreen>
```

（2）建立类 Settings.java。代码如下：

```
46.  package org.example.sudoku;
47.  import android.os.Bundle;
48.  import android.preference.PreferenceActivity;
49.  public class Settings extends PreferenceActivity {
50.  @Override
51.  protected void onCreate(Bundle savedInstanceState) {
52.  super.onCreate(savedInstanceState);
53.  addPreferencesFromResource(R.xml.settings);
54.  }
55.  }
```

至此，一个完整的 Android 菜单已经建立好了~。

1.2 Android 的数据存储

1.2.1 Android数据存储访问机制

我们将会在这篇文章中为大家详细介绍Android数据存储中的数据类型以及存储机制等方面的内容。希望可以给大家带来一些帮助。

大家在开发 Android 操作系统的时候，可能会经常碰到关于数据存储方面的一些操作。在这里我们会为大家详细介绍一下有关 Android 数据存储的一些基本概念以及应用技巧。在 Android 系统中，所有应用程序数据都是私有的，任何其他应用程序都是无法访问的。

1. 如何使得应用程序的数据可以被外部访问呢？

答案是使用 android 的 content provider 接口，content provider 可以使应用程序的私有数据暴露给其它 application.

有两种选择来暴露 application data,一种是建立自己的 content provider,另外一种是使用已有的 content provider 前提是数据类型一致。

2. Android 数据存储的数据类型

Android 提供了一系列的 content type. 包括 image, audio, and video files and personal contact information 等等.

3. Android 数据存储机制

Android 提供了存储和获取数据的以下几种机制

3.1. Preference

Preference 提供了一种轻量级的存取机制，主要是可以通过关键字读取和存储某个 Preference value.

比如载系统启动的时候得到上次系统退出时候保存的值。

```
o  view plaincopy to clipboardprint?
o       . . .
o       public static final String PREFS_NAME = "MyPrefsFile";
o       . . .
o       @Override
o       protected void onCreate(Bundle state){
o       super.onCreate(state);
o       . . .
o       // Restore preferences
o       SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
o       boolean silent = settings.getBoolean("silentMode", false);
o       setSilent(silent);
o       }
o       @Override
o       protected void onStop(){
o       super.onStop();
o       // Save user preferences. We need an Editor object to
o       // make changes. All objects are from android.context.Context
o       SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
o       SharedPreferences.Editor editor = settings.edit();
o       editor.putBoolean("silentMode", mSilentMode);
o       // Don't forget to commit your edits!!!
o       editor.commit();
```

```
o       }
o       . . .
o       public static final String PREFS_NAME = "MyPrefsFile";
o       . . .
o       @Override
o       protected void onCreate(Bundle state){
o       super.onCreate(state);
o       . . .
o       // Restore preferences
o       SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
o       boolean silent = settings.getBoolean("silentMode", false);
o       setSilent(silent);
o       }
o       @Override
o       protected void onStop(){
o       super.onStop();
o       // Save user preferences. We need an Editor object to
o       // make changes. All objects are from android.context.Context
o       SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
o       SharedPreferences.Editor editor = settings.edit();
o       editor.putBoolean("silentMode", mSilentMode);
o       // Don't forget to commit your edits!!!
o       editor.commit();
o       }
```

3.2. Files

通过 Android 数据存储中的 File 机制你可以直接存储一个文件到你手机文件系统路径比如 SD 卡中。

需要注意的是 ， 默认情况下存储的文件是不可以被其他 application 是访问的 !!

Context.openFileInput() 返回 java 的标准文件输入对象。

Context.openFileOutput() 返回 java 的标准文件输出对象。

3.3. Databases.

Android 使用 SQLite 数据库。

可以通过调用 SQLiteDatabase.create() and 以及子类 SQLiteOpenHelper.

Android 还提供了 sqlite3 database tool， 你可以通过这个工具像 MySQL tool 那样来直接访问，修改数据库

3.4. Network.

最后你也可以通过网络来存储数据，使用下面两个包的 java class.

```
o  java.net.*
o  android.net.*
```

Android 数据存储的相关应用就为大家介绍到这里。

1.2.2 SQLite数据库

我们在这篇文章中为大家总结的Android数据库操作的方式都包括有：创建数据库；操作数据库；数据显示以及导出数据库等等。

在 Android 这样一个开源的手机操作系统中，对于数据库的操作是不可避免而且非常重要的。在这里我们就为大家详细介绍一下 Android 数据库操作的一些基本应用技巧，以方便大家的学习。

Android 数据库操作 1. 创建数据库

Android 提供了标准的数据库创建方式。继承 SQLiteOpenHelper ,实现 onCreate 和 onUpgrade 两个方法，有个好处就是便于数据库版本的升级。

```
1.
private static class DatabaseHelper extends SQLiteOpenHelper {
2.  DatabaseHelper(Context context) {
3.  super(context, DATABASE_NAME, null, DATABASE_VERSION);
4.  }
5.  @Override
6.  public void onCreate(SQLiteDatabase db) {
7.  }
8.  @Override
9.  public void onUpgrade(SQLiteDatabase db, int oldVersion,
10.  int newVersion) {
11.  }
12.  }
```

Android 数据库操作 2. 操作数据库

之前一直疑惑于一个功能操作是否要每次打开数据库完之后，要立即关闭数据库？ 还是一个应用程序只开一次数据库，等到应用程序退出时再关闭数据库？ 其中的顾虑在性能和多线程。 想

到一个方法 ，将数据库操作类作为单件，将数据库的 lock 设置为 false. 关闭数据库的 lock.， 而在每一个 Table 的修改时加上相应的 Table 锁。

```
13.  public class WebViewDatabase {
14.  public static synchronized WebViewDatabase getInstance(Context context) {

15.  // use per table Mutex lock, turn off database lock, this
16.  // improves performance as database's ReentrantLock is expansive
17.  mDatabase.setLockingEnabled(false);
18.  }
19.  // synchronize locks
20.  private final Object mHttpAuthLock = new Object();
21.  public boolean hasHttpAuthUsernamePassword() {
22.  synchronized (mHttpAuthLock) {
23.  return hasEntries(TABLE_HTTPAUTH_ID);
24.  }
25.  }
26.  }
```

想想看，将将底层数据封装成 ContentProvider，供应用程序调用， 标准的做法，就是如果对 ContentProvider 不是很熟悉的话，就有点麻烦了。

Android 数据库操作 3. 数据显示

Cursor 前面说过，是一个指向数据源的随机迭代器显示数据。将 View 绑定到 Cursor 通常要设置这样几个参数。一个是每一行的样式，称作 Row Layout，其实就是一个普通的 Layout 的 XML 文件。还有就是一个列和现实控件的对应关系。那个控件显示哪个列的值，这是需要配置的。为了定制一个良好的数据显示控件，最简单你可以定制很 PP 的 Row Layout，复杂一点就是可以重载绑定控件 View，或者是适配器 ListAdapter。

要使用 Cursor 动态绑定 View，每个表有一个_id 列。

重新绑定 Cursor， 并刷新页面

```
27.  Cursor.requery().
28.  Adapter. notifyDataSetChanged();
```

想到一个问题，数据量非常大的时候，会不会出现内存不足的现象？Cursor 是动态绑定 View. 深入去看 android 的代码,CursorWindow 内部提供了 Buffer， 供将数据库的数据拷贝到该 Buffer. 作为 View 显示的缓冲区，其大小是有限的。根据 V iew 的变化重新填充 Buffer.

Android 数据库操作 4. 导出数据库

我们要查看手机的数据库内容，每次都要从手机导出，然后 sqlite 工具查看。其实用 adb shell sqlite 直接查看手机的数据库， 在这抛砖引玉了，有志之人将其做一个工具，封装 sqlite 语句，便可以直接操作手机数据库。

1.2.3 文件存储

Android 开源手机操作系统可以让开发人员在模拟器中进行相应的操作以使系统满足用户的各种需求。在这里我们可以通过对 Android 读写文件的相关操作来体验一下这款操作系统的编写方式。

众所周知 Android 有一套自己的安全模型， 具体可参见 Android 开发文档。当应用程序(.apk)在安装时就会分配一个 userid，当该应用要去访问其他资源比如文件的时候，就需要 userid 匹配。默认情况下 ，任何应用创建的文件，数据库， sharedpreferences 都应该是私有的（位于 /data/data/your_project/files/），其余程序无法访问。除非在创建时指明是 MODE_WORLD_READABLE 或者 MODE_WORLD_WRITEABLE,只要这样其余程序才能正确访问。

因为有这种 Android 读写文件的方法在安全上有所保障，进程打开文件时 Android 要求检查进程的 user id。所以不能直接用 java 的 api 来打开，因为 java 的 io 函数没有提这个机制 。

无法用 java 的 api 直接打开程序私有的数据 ,默认路径为/data/data/your_project/files/

```
o   FileReader file = new FileReader("Android.txt");
```

这里特别强调私有数据！言外之意是如果某个文件或者数据不是程序私有的，既访问它时无须经过 Android 的权限检查，那么还是可以用 java 的 io api 来直接访问的。所谓的非私有数据是只放在 sdcard 上的文件或者数据，

可以用 java 的 io api 来直接打开 sdcard 上文件。

```
o   FileReader file = new FileReader("/sdcard/Android.txt");
```

如果要打开程序自己私有的文件和数据，那必须使用 Activity 提供 openFileOutput 和 openFileInput 方法。

创建程序私有的文件，由于权限方面的要求，必须使用 activity 提供的 Android 读写文件方法

```
o   FileOutputStream os = this.openFileOutput("Android.txt", MODE_PRIVATE);
```

```
o   OutputStreamWriter outWriter = new OutputStreamWriter (os);
```

读取程序私有的文件，由于权限方面的要求，必须使用 activity 提供的方法

```
o   FileInputStream os =this.openFileInput("Android.txt");
o   InputStreamReader inReader = new InputStreamReader(os);
```

Android 读写文件的相关操作就为大家介绍到这里。

1.2.4 Android使用XML技巧

在Android操作系统中，关于可视化编程的方法有很多种。这里主要就是针对Android使用XML的方法来为大家介绍一些这方面的应用。

大家可能还记得，我们在上一篇文章中向大家详细介绍了 Android ListView 的相关应用，它主要就是针对于可视化编程。在这里我们会通过 Android 使用 XML 来为大家详细介绍另一种可视化编程方法。

就如跨平台 UI 界面库一样，Android 也是使用 XML 文件来存贮界元素持布局，现在流行的一些界面组件都是采用 Android 使用 XML 的方式。

在 Android 中，res/layout 资源目录下，会有一个或多个.xml 文件，这就是一个界面的布局文件。我们打开一个来看看。我打开当前工程目录下的 res/layout/main.xml 文件。

```
1.  < ?xml version="1.0" encoding="utf-8"?>
2.  < LinearLayout xmlns:android=
   "http://schemas.android.com/apk/res/android"
3.  android:orientation="vertical"
4.  android:layout_width="fill_parent"
5.  android:layout_height="fill_parent"
6.  >
7.  < TextView
8.  android:layout_width="fill_parent"
9.  android:layout_height="wrap_content"
10. android:text="@string/hello"
11. android:id="@+id/mainview"
12. />
13. < /LinearLayout>
```

这个文件很简单的布局是指用了一个 LinearLayout 来布局，里面只有一个 TextView 界面元素，也就是一个 View. 当 Activity 加载 View 时，就可以在 onCreate 中直接加载。

this.setContentView(R.layout.main);其中 R.layout.main 就是一个素引值，是由 android 开发环境编译生成的，是映射到 res/layout/main.xml 的。

所以 setContentView(R.layout.main);等价于，按装 main.xml 的布局来配置一个 layout.然后加载，与如下代码效果一致

```
14.  LinearLayout layout = new LinearLayout(this);
15.  TextView tv = new TextView(this);
16.  tv.setText(R.string.hello);
17.  layout.addView(tv);
18.  this.setContentView(layout);
```

其中 R.string.hello 也是一个资源映射的 ID，是指加载 res/values/string.xml 中的 hello 对应的值。

1.3 Android 的通信开发

1.3.1 Android电话功能

Android 手机操作系统是一款基于 Linux 平台的开源系统。开发人员可以根据不同的需求对其进行修改等操作。在这系统中有很多比较重要的功能值得我们去研究。比如 Android 电话功能就是其中一个基础知识点。

第一部分 Android 电话功能概述

Android 的 Radio Interface Layer (RIL)提供了电话服务和的 radio 硬件之间的抽象层。

Radio Interface Layer RIL(Radio Interface Layer)负责数据的可靠传输、AT 命令的发送以及 response 的解析。应用处理器通过 AT 命令集与带 GPRS 功能的无线通讯模块通信。

AT command 由 Hayes 公司发明，是一个调制解调器制造商采用的一个调制解调器命令语言，每条命令以字母"AT"开头。

JAVA Framework

代码的路径为：

- frameworks/base/telephony/java/android/telephony
- android.telephony 以及 android.telephony.gsm

Core native:

在 hardware/ril 目录中，提供了对 RIL 支持的本地代码，包括 4 个文件夹：

```
o  hardware/ril/include
o  hardware/ril/libril
o  hardware/ril/reference-ril
o  hardware/ril/rild
```

kernel Driver

在 Linux 内核的驱动中，提供了相关的驱动程序的支持，可以建立在 UART 或者 SDIO，USB 等高速的串行总线上。

第二部分 Android 电话功能各个部分

hardware/ril/include/telephony/目录中的 ril.h 文件是 ril 部分的基础头文件。

其中定义的结构体 RIL_RadioFunctions 如下所示：

```
o  typedef struct {
o  int version;
o  RIL_RequestFunc onRequest;
o  RIL_RadioStateRequest onStateRequest;
o  RIL_Supports supports;
o  RIL_Cancel onCancel;
o  RIL_GetVersion getVersion;
o  } RIL_RadioFunctions;
```

RIL_RadioFunctions 中包含了几个函数指针的结构体，这实际上是一个移植层的接口，下层的库实现后，由 rild 守护进程得到这些函数指针，执行对应的函数。

几个函数指针的原型为：

```
o  typedef void (*RIL_RequestFunc) (int request, void *data,
o  size_t datalen, RIL_Token t);
o  typedef RIL_RadioState (*RIL_RadioStateRequest)();
o  typedef int (*RIL_Supports)(int requestCode);
o  typedef void (*RIL_Cancel)(RIL_Token t);
o  typedef const char * (*RIL_GetVersion) (void);
```

其中最为重要的函数是 onRequest()，它是一个请求执行的函数。

2.1 rild 守护进程

rild 守护进程的文件包含在 hardware/ril/rild 目录中，其中包含了 rild.c 和 radiooptions.c 两个文件，这个目录中的文件经过编译后生成一个可执行程序，这个程序在系统的安装路径在：

- /system/bin/rild

rild.c 是这个守护进程的入口，它具有一个主函数的入口 main，执行的过程是将请求转换成 AT 命令的字符串，给下层的硬件执行。在运行过程中，使用 dlopen 打开路径为/system/lib/中名称为 libreference-ril.so 的动态库，然后从中取出 RIL_Init 符号来运行。

RIL_Init 符号是一个函数指针，执行这个函数后，返回的是一个 RIL_RadioFunctions 类型的指针。得到这个指针后，调用 RIL_register()函数，将这个指针注册到 libril 库之中，然后进入循环。

事实上，这个守护进程提供了一个申请处理的框架，而具体的功能都是在 libril.so 和 libreference-ril.so 中完成的。

2.2 libreference-ril.so 动态库

libreference-ril.so 动态库的路径是：

- hardware/ril/reference-ril

其中 Android 电话功能主要的文件是 reference-ril.c 和 atchannel.c。这个库必须实现的是一个名称为 RIL_Init 的函数,这个函数执行的结果是返回一个 RIL_RadioFunctions 结构体的指针，指针指向函数指针。

这个库在执行的过程中需要创建一个线程来执行实际的功能。在执行的过程中，这个库将打开一个/dev/ttySXXX 的终端（终端的名字是从上层传入的），然后利用这个终端控制硬件执行。

2.3 libril.so 动态库

libril.so 库的目录是：

- hardware/ril/libril

其中主要的文件为 ril.cpp，这个库主要需要实现的以下几个接口为：

- RIL_startEventLoop(void);
- void RIL_setcallbacks (const RIL_RadioFunctions *callbacks);
- RIL_register (const RIL_RadioFunctions *callbacks);
- RIL_onRequestComplete(RIL_Token t, RIL_Errno e, void *response, size_t responselen);
- void RIL_onUnsolicitedResponse(int unsolResponse, void *data,
- size_t datalen);
- RIL_requestTimedCallback (RIL_TimedCallback callback, void *param,
- const struct timeval *relativeTime);

这些函数也是被 rild 守护进程调用的，不同的 vendor 可以通过自己的方式实现这几个接口，这样可以保证RIL可以在不同系统的移植。其中 RIL_register()函数把外部的RIL_RadioFunctions结构体注册到这个库之中，在恰当的时候调用相应的函数。在 Android 电话功能执行的过程中，这个库处理了一些将请求转换成字符串的功能。

1.4 开发技能总结

1.4.1 Android开发要点经验总结

Android开发要点总结如下：开源和人机界面的一致性、支持多种互动模式、告示管理、支持无缝的互动等。大家可以通过这里介绍的内容对此有一个充分的认识。

Android 手机操作系统已经推出就深受广大编程爱好者的喜爱。从而使应用这一系统的手机得到了良好的发展机遇。在这里我们会为大家详细介绍一下 Android 开发要点，以此来让大家体会这一系统给我们带来的好处。

Android 开发要点之一、开源和人机界面的一致性

从多方面来看，Android 是一种革命性的开源平台，作为开发商在创造更新以及创新服务和应用上有着许多的自由度。为了把 G1AndroidUI 做的尽可能一致，人们已经付出了大量的努力。然而，当在一个开源的环境中工作时，要维持界面和用户体验上高水平的一致性是棘手的。开发商可以(并且应该)自由地选择各种应用软件，让它们看起来应该如何以及如何表现。从产品的观点来看，要保持每一个应用软件的设计与外表、以及对基本操作平台的感觉的紧密关系是至关重要的。用户界面需要一致且可预测，让用户不会面对他们自己的电话不知所措。

Android 会让开发商逐渐开发出奢华的用户界面，这往往要通过不同的开发商向不同的方向扩展来实现，并随着时间的推移无疑将变得更好。然而，设计工程师仍然必须确保与整个用户界面范例的一致性，这样才能缩小形式和功能的碎片化并提供坚实的用户体验。

Android 开发要点之二、支持多种互动模式

Android 被设计在手机外围就可以支持广泛的设备配置。例如，它能够运行在仅具有大触摸屏控制的设备上，或者，运行在具有 4 个导航方向键的小屏幕设备上。为了做出一种适用于大量不同上下文的可扩展的互动范例，开发商付出了大量的努力。

这意味着较之于其它的平台有很少的范例差异。例如，在间接操作上下文中有一个加亮区，就像当用户使用一台 D-Pad 的时候，这个加亮区在触摸互动期间会消失。当采用触摸屏时，真的没有必要加亮图标或列表项，因为你的手指就是加亮区。

另一个差异在于，动作被分为针对项目(item)以及针对屏幕的动作。针对项目的动作是诸如把图像“通过 MMS 发送”这样的事情，并且仅仅可用于单个的项目。针对屏幕的动作是那些你想要在屏幕上对所有的项目执行的动作，如把图像“按照时间分类”。针对项目的动作可以通过长时间按住每一个项目来实现，针对屏幕的动作通过菜单键实现。因为存在当有些项目没有被加亮的情况，不推荐在菜单键下面放置有前后关系的、针对项目的动作。把这一点放在个人电脑的条件下，你如何能够刚好点一个对象而不看到鼠标在哪个位置呢？

一般来说，为 Android 设计应用软件意味着，在不了解设备将做什么配置或者在不了解是哪一种设备的情况下进行设计。因此，应用软件需要支持面向触摸屏以及非触摸屏的用户行为、假设以及互动作用。

Android 开发要点之三、告示管理

Android 具有一种非插入的独一无二的告示系统，然而，它随处可以获取并且本质上具有很强的可扩展性。

下拉窗口本质上就是对状态栏的一种扩展：它可以在所有的应用软件内调用，并为新的事件告示提供附加的动作和信息。

这种告示系统非常灵活且可扩展，任何第三方开发商均能够在这里披露新的事件告示。如果有人创建一种新的 Twitter 应用软件，这个人提供的新动作可以变为一个新的事件告示。然而，至关重要的是考虑应用软件所处的环境和事件，并经济地采用告示系统以便于尽可能最好地利用用户有限的注意广度。

Android 开发要点之四、支持无缝的互动

Android 酷毙之处在于它的内建的意图处理功能，这是一段能够要求在其它应用软件中使用某一功能的应用软件。网络浏览器就能够要求运行适当的 PDF 文件的阅读器，而在设备上能够处理这一请求的应用软件能够无缝地与该功能联用。

从框架的观点来看，这不仅是一种巨大的可扩展的系统，而且从用户的观点来看非常强大。意图处理使得开发商有可能支持无缝的互动，使得一个用户任务能够跨越多个应用软件。它是一种以人的任务为中心的方法，而不是以系统应用为中心的方法。因此，不管它称为什么，Android 就是一种非常人性化的平台。

1.4.2 Android快速启动总结

Android 手机操作系统的推出，为智能手机领域增添了不小的生机，而且其开源性帮助不少商家获得了非常好的发展机遇。Android 系统框架和上层应用是类 java（不是正统的 sun java）开发的，实现了自己的 java 虚拟机 dalvik，既然用 java 虚拟机和 java 开发，一般都会认为效率低下。其实不然，在基本主流的智能手机的软件平台上，Android 的执行速度是最快的。

那么 Android 效率为什么这么的高呢？特别是一个应用程序的启动时间很短，本文主要从以下个八方面对 Android 快速启动进行分析：

1、 资源文件的优化读取。

我们知道 Android 在 UI 开发时有个很大的好处是 xml 文件来描述 UI，这样有个好处是只要修改 UI 不用修改代码就可以修改界面的布局、显示风格和字体大小等。界面定义变得灵活方便。xml 配置 UI 在 qtopia 运用也有但是这么强大并且也不广泛，因为 xml 文件有个不足是解析 xml 的效率很低。

Android 是怎么做的呢？

Android 在编译的时候就把 xml 文件进行了优化，Android 应用程序在解析时变得非常的高效，从而实现了 Android 快速启动。我们看到 apk 文件解压后会有个优化过的资源文件。

2、 安装时进行优化 dex 文件

Android 的应用程序都打包成一个 apk 文件，实际上就是一个 zip 文件。系统第一次起来或应用程序第一次安装时，系统就把 apk 文件解压了，把可执行文件 dex 优化成 odex 文件并放在 /data/dalvik-cache 目录下。优化后的 dex 文件启动速度会加快。这解释了为什么 Android 系统第一次启动是比较慢，以后起来很快了。

可能有人会问：为什么不在编译时直接优化呢？第⑤项会回答这个问题。

3、 制作数据库

Android 的图形应用是加载整个 sd 卡内的所有图像的，但是为什么很快呢？其实 Android 提前把数据做成了数据库，所以不用每次扫描整个这个 sd 卡，大大加快了启动速度。

4、 高效的虚拟机实现 Android 快速启动

Android 是基于类 java 虚拟机 dalvik，一般的 java 虚拟机是基于栈的，而 dalvik 是基于寄存器的。实事求是说我对两者的区别了解不是很深入，不过网上有专门的相关文论进行分析。我的简单理解是栈的实现方式相对容易，相关数据是在内存中的栈里，而操作寄存器里数据的速度明显快与内存里的数据处理。

5、 充分挖掘 CPU 的性能

Android 刚出来的时候虽然支持 arm cpu，实际上只支持 armv5te 的指令集的，因为 Android 系统专门为 armv5te 进行了优化，充分利用 armv5te 的执行流水线来提高执行的效率，这也是在 500M 的三星 2440 运行效果不是很好，而在 200M 的 omap cpu 上运行比较流畅的原因了，所以在最新的代码中有专门针对 x86 和 armv4 的优化部分。

6、 优化和裁剪的 libc 库

Libc 库几乎是所以库和程序的基础，但是 Android 没有直接利用 libc 库，而是自己开发了一个库：bionic，它实现了 libc 库的绝大多数的函数并根据平台进行了优化，但是有系统很少用并且消耗资源的少数函数是不支持的。它只有几百 k，节省了空间同时也提高了执行效率。实际上体现了 20-80 原则，抓住少数重要的适当舍弃不必要的。

7、 充分利用 linux 系统特性

分析过 linux 内核的朋友知道，linux fork 一个新的进程是非常高效的，利用了 COW 机制。Android 是每个进程是个独立的虚拟机（听说这么设计是为安全考虑，某个时候进程崩溃了不会影响这个系统和其他进程。）Android 里每个进程都是基于虚拟机的，并且也要加载基本的库，实际上这些都是共享。所以 Android 启动一个新的程序实际上并不消耗很多的内存和 cpu 资源。

同时 Android 在后台有个 empty process 运行，实际上就是运行一个虚拟机，当要启动一个应用时就直接在其上继续运行，qtopia 也有这个机制。

Android 系统在开机流程中：启动虚拟机—》启动 system server –》启动 launcher。当初分析代码时疑惑为什么不直接启动 system server？（qtopia 就是直接启动 server），实际上也利用了 linux 的这个特性。

这个特性说的比较简略，不过要真的把他解释清楚可能需要很大的篇幅。

8、 高效的 paint 机制

这个特性可能跟 Android 快速启动关系不大，但是也是 Android 高效的特性之一。界面变化时大部分实际上不是全屏内容变化的，只是局部变化，Android 会根据变化的内容只是跟新局部的内容，也提高了效率。这个也提醒我们在开发应用程序时，重载 paint 方法时尽量不要 paint 全屏内容。

1.4.3 Android垃圾回收实现

Android垃圾回收的实现其实可以看做是对sp以及wp的操作。我们将会在这篇文章中对这两种操作

分别做一详细介绍。让大家充分掌握这一方面的知识。

Android 手机操作系统中的代码编写方式对于有基础的编程人员来说是比较容易的。因为它是基于 Linux 平台的操作系统。我们在这里为大家介绍的是 Android 垃圾回收这一机制，以加深大家对这一系统的了解。

- Android 虚拟设备适用你的部署目标
- Android adb 中命令的运行
- Android 菜单构造技巧
- Android 图片浏览源码解读
- Android 快速启动要点总结

个人觉得 sp 和 wp 实际上就是 Android 为其 c++实现的自动垃圾 回收机制 ，具体到内部实现，sp 和 wp 实际上只是一个实现垃圾回收功能的接口而已，比如说对＊，—>的重载，是为了其看起来跟真正的指针一样，而真正实现垃圾回收的是 refbase 这个基类。这部分代码的目录在：/frameworks/base/include/utils/RefBase.h

首先所有的类都会虚继承 refbase 类，因为它实现了达到 Android 垃圾回收所需要的所有 function，因此实际上所有的对象声明出来以后都具备了自动释放自己的能力，也就是说实际上智能指针就是我们的对象本身，它会维持一个对本身强引用和弱引用的计数，一旦强引用计数为 0 它就会释放掉自己。

首先我们看 sp，sp 实际上不是 smart pointer 的缩写，而是 strong pointer，它实际上内部就包含了一个指向对象的指针而已。我们可以简单看看 sp 的一个构造函数：

```
template< typename T>
sp< T>::sp(T* other)
: m_ptr(other)
{
if (other) other->incStrong(this);
}
```

比如说我们声明一个对象：

```
sp< CameraHardwareInterface> hardware(new CameraHal());
```

实际上 sp 指针对本身没有进行什么操作，就是一个指针的赋值，包含了一个指向对象的指针，但是对象会对对象本身增加一个强引用计数，这个 incStrong 的实现就在 refbase 类里面。新 new 出来一个 CameraHal 对象，将它的值给 sp< CameraHardwareInterface>的时候，它的强引用计数就会从 0 变为 1。因此每次将对象赋值给一个 sp 指针的时候，对象的强引用计数都会加 1，下面我们再看看 sp 的析构函数：

```
template< typename T>
sp< T>::~sp()
```

```
{
if (m_ptr) m_ptr->decStrong(this);
}
```

实际上每次 delete 一个 sp 对象的时候，sp 指针指向的对象的强引用计数就会减一，当对象的强引用技术 为 0 的时候这个对象就会被自动释放掉。

我们再看 wp，wp 就是 weak pointer 的缩写，弱引用指针的原理 ，就是为了应用 Android 垃圾回收来减少对那些胖子对象对内存的占用，我们首先来看 wp 的一个构造函数：

```
wp< T>::wp(T* other)
: m_ptr(other)
{
if (other) m_refs = other->createWeak(this);
}
```

它和 sp 一样实际上也就是仅仅对指针进行了赋值而已，对象本身会增加一个对自身的弱引用计数，同时 wp 还包含一个 m_ref 指针，这个指针主要是用来将 wp 升级为 sp 时候使用的：

```
template< typename T>
sp< T> wp< T>::promote() const
{
return sp< T>(m_ptr, m_refs);
}
template< typename T>
sp< T>::sp(T* p, weakref_type* refs)
: m_ptr((p && refs->attemptIncStrong(this)) ? p : 0)
{
}
```

实际上我们对 wp 指针唯一能做的就是将 wp 指针升级为一个 sp 指针，然后判断是否升级成功，如果成功说明对象依旧存在，如果失败说明对象已经被释放掉了。wp 指针我现在看到的是在单例中使用很多，确保 mhardware 对象只有一个，比如：

```
wp< CameraHardwareInterface> CameraHardwareStub::singleton;
sp< CameraHardwareInterface> CameraHal::createInstance()
{
LOG_FUNCTION_NAME
if (singleton != 0) {
sp< CameraHardwareInterface> hardware = singleton.promote();
if (hardware != 0) {
return hardware;
}
```

```
        }
        sp< CameraHardwareInterface> hardware(new CameraHal()); //强引用加 1
        singleton = hardware;//弱引用加 1
        return hardware;//赋值构造函数，强引用加 1
        }
        //hardware 被删除，强引用减 1
```

Android 垃圾回收的相关内容就为大家介绍到这里。

1.4.4 Android后台程序应用技巧

Android 手机操作系统是由谷歌推出的一款开源的基于 Linux 平台的操作系统，深受广大编程爱好者的喜爱。在 Android 系统中我们一直在接触着前台界面程序，其实在一开始接触 Android 时就听说了，程序就有有界面和无界面之分。

Android 后台程序就是这类无界面的程序，它在后台执行，没有影响你的界面。比如短信监听程序，执行在后台，当有短信时才给你们提示，振动或声音；比如闹钟，设定好时间后，在定时通知你；再比如 mp3 播放器，选择好音乐后，在待在后台唱着，当有电话来时，自动暂停，完后再继续播放。

其实分析下来，我们不难发现，Android 后台程序跟前台程序是一样的，也就是在执行我们指定的程序，只是留给我们两个问题， 1.因为没有界面，我们会问，怎么启动，怎么终止？ 2.因为没有界面，这程序如何通知我们一些信息或状态。

前面的学习让我们知道，一个 Activity 想 Call 另一个 Activity 时，只需要能过中介人 Intent 就可以了，同样我们与服务处理类打交道也是通过 Intent 来实现，当然，界面类是继承着 Activity，而服务类则是继承着 Service 类。

启动服务：

```
// Implicitly start a Service
startService(new Intent(MyService.MY_ACTION));
// Explicitly start a Service
startService(new Intent(this, MyService.class));
```

停止服务：

```
stopService(new Intent(this, MyService.class));
```

同样，跟 Activity 一样的生命期中，系统也会自动跟据不同的状态来调用继承函数：

```
@Override
```

```
o  public void onCreate()
o  public IBinder onBind(Intent intent)
o  public void onStart(Intent intent, int startId)
o  。。。
```

在实际的开发中，我们一般都不会直接写一个服务类，一般都会写一个与 Android 后台程序相配套的前台程序，一般的程序总会有一些配置吧～～，然后这个界面中就可以很方便地来控制后台程序的运作。

我们来回答第二个问题，就是在服务中我们怎么发起一个通知给用户，在 Andorid 中，提供了以下几种方式：

1. Toast

这是一个无模式的小窗体，会将显示的信息显示在首页面中：

实现代码是：

```
o  Context context = getApplicationContext();
o  String msg = "To the bride an groom!";
o  int duration = Toast.LENGTH_SHORT;
o  Toast toast = Toast.makeText(context, msg, duration);
o  int offsetX = 0;
o  int offsetY = 0;
o  toast.setGravity(Gravity.BOTTOM, offsetX, offsetY);
o  toast.show();
```

当然，你也可以显示更杂的，可以将一个控制直接当成一个 Toast 显示出来，也可以自定义一个控件显示出来，自定义控件的强大是大家都知道的～～

2. Notifications

这种方式是系统中比较通用的模式，通过这种方式你可以使系统：将一个图标在状态条上闪，让机器震动，发出声音等。

实现代码：

```
o  String svcName = Context.NOTIFICATION_SERVICE;
o  NotificationManager notificationManager;
```

```
notificationManager = (NotificationManager)getSystemService(svcName);

// Choose a drawable to display as the status bar icon
int icon = R.drawable.icon;
// Text to display in the status bar when the notification is launched

String tickerText = "Notification";
// The extended status bar orders notification in time order
long when = System.currentTimeMillis();
Notification notification = new Notification(icon, tickerText, when);

Context context = getApplicationContext();
// Text to display in the extended status window
String expandedText = "Extended status text";
// Title for the expanded status
String expandedTitle = "Notification Title";
// Intent to launch an activity when the extended text is clicked
Intent intent = new Intent(this, MyActivity.class);
PendingIntent launchIntent = PendingIntent.getActivity(context, 0, int
ent, 0);
notification.setLatestEventInfo(context, expandedTitle,expandedText,la
unchIntent);
```

触发方式：

```
int notificationRef = 1;
notificationManager.notify(notificationRef, notification);
```

学会了 Activity 再写个 Android 后台程序也就不难了！！

这里顺便再提一下，在 Android 系统中也提供了多线程编程，我们知道不管是前台还是后台程序，都有生命期的，当程序不活动时，我们想继续让程序执行，这里我们需要用到线程了，在 Android 系统中使用线程，跟我们直接写 java 线程程序非常想似：

```
// This method is called on the main GUI thread.
private void mainProcessing() {
// 主程序中启动线程.
Thread thread = new Thread(null, doBackgroundThreadProcessing,
"Background");
thread.start();
```

```
o   }
o   // Runnable that executes the background processing method.
o   private Runnable doBackgroundThreadProcessing = new Runnable() {
o   public void run() {
o   //线程执行内容。。。
o   }
o   };
```

Android 后台程序的相关应用就为大家介绍的这里。

1.4.5 Android实现全屏

如何才能正确的编写代码来实现全屏功能呢？我们将会通过对一段代码的解读来详细接受一下Android实现全屏的操作方法。

Android 手机操作系统是由谷歌推出的一款基于 Linux 的开源手机操作系统。我们可以在模拟器中对其进行相应的操作来实现各种功能以满足用户的需求。在这里就简要介绍一下 Android 实现全屏的相关方法。

新版本的 Android Framework 和老版本的实现起来有些不同。这里只给出新版本的 Android 实现全屏代码。

```
1.  package pub.tetris;
2.  import android.app.Activity;
3.  import android.os.Bundle;
4.  import android.view.Window;
5.  import android.view.WindowManager;
6.  public class TetrisActivity extends Activity {
7.  /** Called when the activity is first created. */
8.  @Override
9.  public void onCreate(Bundle savedInstanceState) {
10. super.onCreate(savedInstanceState);
11. /**全屏设置，隐藏窗口所有装饰*/
12. getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
13. WindowManager.LayoutParams.FLAG_FULLSCREEN);
14. /**标题是属于 View 的，所以窗口所有的修饰部分被隐藏后标题依然有效*/
15. requestWindowFeature(Window.FEATURE_NO_TITLE);
16. TileView tile = new TileView(this);
17. Tetris tetris = new Tetris();
18. tetris.init(tile);
```

```
19.  Thread engine = new Thread(tetris);
20.  setContentView(tile);
21.  engine.start();
22.  }
23.  }
```

Android 实现全屏相关操作方法就为大家介绍到这里。

1.5 Android API分析

1.5.1 Android可选API适用范围

Android 手机操作系统已经推出就受到了广大用户的好评。尤其是它开源的特点大大吸引了开发爱好者的青睐。Android 适用于各种各样的手机，从最低端直到最高端的智能手机。核心的 Android API 在每部手机上都可使用，但任然有一些 API 接口有一些特别的适用范围：这就是所谓的 Android 可选 API。

Android 可选 API 主要是因为一个手持设备并不一定要完全支持这类 API，甚至于完全不支持。例如，一个手持设备可能没有 GPS 或 Wi-FI 的硬件。在这个条件下，这类功能的 API 任然存在，但不会以相同的方式来工作。例如 Location API 任然在没有 GPS 的设备上存在，但极有可能完全没有安装功能提供者，意味着这类 API 就不能有效地使用。

你的应用应该无障碍地运行或连接在一个可能不支持你 API 的设备，因为你的设备上有这些上层接口（the classes）。当然执行起来可能什么也不会做，或者抛出一个异常。每个 API 会做些什么我们可以参考这些 API 的说明文档，你应该编写你的程序来适当的处理这类问题。

Wi-Fi API

Wi-Fi API 为应用程序提供了一种与那些带有 Wi-FI 网络接口的底层无线堆栈相互交流的手段。几乎所有的请求设备信息都是可利用的，包括网络的连接速度、IP 地址、当前状态等等，还有一些其他可用网络的信息。一些可用的交互操作包括扫描、添加、保存、结束和发起连接。

Wi-Fi API 在 android.net.wifi 包中。

定位服务（Location-Based Services）

定位服务允许软件获取手机当前的位置信息。这包括从全球定位系统卫星上获取地理位置，但相关信息不限于此。例如，未来其他定位系统可能会运营，届时，对其相应的 API 接口也会加入到系统中。

定位服务的 API 在 android.location 包中。

多媒体 API（Media APIs）

多媒体 API 主要用于播放媒体文件。这同时包括对音频（如播放 MP3 或其他音乐文件以及游戏声音效果等）和视频（如播放从网上下载的视频）的支持，并支持"播放 URI 地址"（Note:URI 即是统一资源识别地址）模式－在网络上直接播放的流媒体。技术上来说，多媒体 API 并不是 Android 可选 API，因为它总是要用到。但是不同的硬件环境上面可能有不同的编解码的硬件机制，因而它又是“可选的”。

多媒体 API 在 android.media 包中。

基于 OpenGL 的 3D 图形（3D Graphics with OpenGL）

Android 的主要用户接口框架是一个典型的面向控件的类继承系统。但不要让表面的情况迷惑了你，因为在它下面是一种非常快的 2D 和 3 D 组合的图形引擎，并且支持硬件加速。用来访问平台 3D 功能的 API 接口是 OpenGL ES API。和多媒体 API 一样，OpenGL 也不是严格意义上的“可选”，因为这些 API 会总是存在并且实现那些固定的功能。但是，一些设备可能有硬件加速环节，使用它的时候就会影响你的应用程序的表现。

OpenGL 的 API 在 android.opengl 中可以看到。

Google 和 Sun 相同，把部分高端应用作为可选 API 供手机生产商定制不同的硬件支持模块。在 JME 中 Sun 是以 JSR 方式公布而谷歌采用了 optional API

一、Location-Based Services 定位服务

Android 操作系统支持 GPS API-LBS，可以通过集成 GPS 芯片来接收卫星信号通过 GPS 全球定位系统中至少 3 颗卫星和原子钟来获取当前手机的坐标数据，通过转换就可以成为地图上的具体位置了，这一误差在手机上可以缩小到 10 米。在谷歌开发手机联盟中可以看到著名的 SiRF star。所以未来 gPhone 手机上市时集成 GPS 后的价格不会很贵。同时谷歌正在研制基于基站式的定位技术-MyLocation 可以更快速的定位与前者 GPS 定位需要花费大约 1 分钟相比基站定位更快。

二、Media APIs 多媒体接口

Android 平台上集成了很多影音解码器以及相关的多媒体 API， 通过这些可选 API，厂商可以让手机支持 MP3、MP4、高清晰视频播放处理等支持。

三、3D Graphics with OpenGL 3D 图形处理 OpenGL 可选 API

Android 平台上的游戏娱乐功能如支持 3D 游戏、或应用场景就需要用到 3D 技术，手机生产厂商根据手机的屏幕以及定位集成不同等级的 3D 加速图形芯片来加强 gPhone 手机的娱乐性，有来自高通的消息称最新的显示芯片在 gPhone 上将会轻松超过索尼 PS3。

四、Low-Level Hardware Access 低级硬件访问

这个功能主要用于控制手机的底层方面操作，由于设计底层硬件操作，将主要由各个手机硬件生产厂商来定制，支持不同设备的操作管理等支持，如蓝牙 BlueTooth 以及 Wifi 无线网络支持等。

通过本文的介绍相信大家都了解了 gPhone 手机中的技术功能，开发软件或游戏的同时可以了解到 Android 平台的高度可伸缩性，帮助手机硬件厂商控制成本

视图（Views）

视图是基类android.view.View的具体实例，它是一个存储了界面布局和屏幕区域内容的数据结构。一个视图对象可以对屏幕区域做如下操作或处理：操作绘制布局、处理焦点变化、屏幕滚动、按键动作等。

视图类作为基类服务于窗口控件（widgets）——一个完整实现的绘制交互式屏幕元素的子类集合。窗口控件可以独立操作自己的绘制和方法处理，所以你可以使用它们来更快速的创建自己的用户界面。可用的窗口控件包括：Text、EditText、InputMethod、MovementMethod、Button、RadioButton、Checkbox和ScrollView。

视图组（Viewsgroups） 视图组是类android.view.Viewgroup的一个对象。正如它的名字所示，视图组是一种特殊类型的视图对象，它的功能是包含和管理下级试图和其它试图组的集合。视图组让你可以为你的用户界面添加结构，从而建立复杂并可以作为单独实体进行访问的屏幕元素。

视图组作为基类服务于界面布局（layouts）——一个完整实现的并提供了屏幕布局的通用类型的子类集合。界面布局（layouts）提供给你一种建立试图集合结构的方法。

树型结构用户界面（A Tree-Structured UI）

在Android平台上定义一个行为（Activity）的用户界面需要使用如下图所示的视图和视图组节点组成的树型结构。树形结构的复杂程度完全有你来决定。你可以用Android预置的窗口控件和布局创建界面，也可以使用自定义的视图。