

android
developers

Android开发教程&笔记



android

开放手机联盟--Open Handset Alliance

什么是开放手机联盟？

开放手机联盟，Open Handset Alliance：是美国 Google 公司与 2007 年 11 月 5 日宣布组建的一个全球性的联盟组织。这一联盟将会支持 Google 发布的 Android 手机操作系统或者应用软件，共同开发名为 Android 的开放源代码的移动系统。开放手机联盟包括手机制造商、手机芯片厂商和移动运营商几类。目前，联盟成员数量已经达到了 43 家。



移动手机联盟创始成员：

Aplix、Ascender、Audience、Broadcom、中国移动、eBay、Esmertec、谷歌、宏达电、英特尔、KDDI、Living Image、LG、Marvell、摩托罗拉、NMS、NTT DoCoMo、Nuance、Nvidia、PacketVideo、高通、三星、SiRF、SkyPop、Sonic Network、Sprint Nextel、Synaptics、TAT、意大利电信、西班牙电信、德州仪器、T-Mobile 和 Wind River。

Mobile Operators 移动运营商类

China Mobile Communications Corporation 中国移动通信

KDDI CORPORATION 日本 KDDI 电信

NTT DoCoMo, Inc. 日本多科莫电信

SOFTBANK MOBILE Corp. 日本软银移动

Sprint Nextel(美国)

T-Mobile(德国)

Telecom Italia(意大利)

Telefónica(西班牙)

Vodafone 沃达丰电信

China Unicom 中国联通

Semiconductor Companies 半导体制造公司

AKM Semiconductor Inc

Audience

ARM

Atheros Communications

Broadcom Corporation(博通)

Ericsson (爱立信公司)

Intel Corporation (英特尔公司)

Marvell Semiconductor, Inc. (收购了 intel 手机芯片部门的公司)

Android 编程基础

NVIDIA Corporation (英伟达公司)
Qualcomm Inc.(高通公司)
SiRF Technology Holdings, Inc.(知名 GPS 芯片制造商)
Synaptics, Inc.
Texas Instruments Incorporated (德州仪器)

Handset Manufacturers 电话制造商

ASUSTeK Computer Inc. 华硕
Garmin International, Inc.
HTC Corporation (多普达的母公司)宏达电子
Huawei Technologies 华为科技
LG Electronics, Inc. 乐金电子

Motorola, Inc. 摩托罗拉
Samsung Electronics 三星电子
Sony Ericsson 索尼爱立信
Toshiba Corporation 东芝公司
lenovo 联想移动



联盟成员：Software Companies 软件提供公司

Ascender Corp.
eBay Inc.
Esmertec
Google Inc.
LivingImage LTD.
Nuance Communications, Inc.
OMRON SOFTWARE Co, Ltd. 日本欧姆龙软件
有限公司
PacketVideo (PV)
SkyPop
SONiVOX
ASUSTeK Computer Inc. 华硕
AKM Semiconductor AKM 半导体公司
ARM 公司
Borqs 播思通讯

Atheros Communications
Toshiba Corporation 东芝公司
lenovo 联想移动
软银移动 日本无线运营商软银
瑞典计算机咨询公司 Teleca AB
Garmin International, Inc. 高明
HTC Corporation (多普达的母公司)宏达电子
Huawei Technologies 华为科技
LG Electronics, Inc. 乐金电子
Motorola, Inc. 摩托罗拉
Samsung Electronics 三星电子
Sony Ericsson 索尼爱立信
沃达丰
Teleca

联盟目的

将会支持 Google 可能发布的手机操作系统或者应用软件，共同开发名为 Android 的开放源代码的移动系统。

谷歌早在 2002 年就进入了移动领域，可是由于目前的手机操作系统企业和手机企业相对封闭，提高了行业的进入门槛，移动互联网的发展远没有拥有统一标准的传统互联网发展迅速，此次推出的开源手机操作系统平台就是出于这个目的。

也有分析认为，谷歌并不想做一个简单的手机终端制造商或者软件平台开发商，而意在统一传统互联网和移动互联网。

Android 手机新概念

操作系统的选择--定制和长尾

- ◆ 重构
- ◆ MVC 和 Web APP 架构

Android 开发背景



- ◆ 计算技术、无线接入技术的发展，使嵌入式系统逐渐有能力对桌面系统常规业务进行支持。
- ◆ 谷歌长期以来奉行的移动发展战略：通过与全球各地的手机制造商和移动运营商结成合作伙伴，开发既有用又有吸引力的移动服务，并推广这些产品。Android 进一步推进了"随时随地为每个人提供信息"这一企业目标的实现。
- ◆ Open Handset Alliance 汇集了多家业界巨头。运营商如：China Mobile、NTT DoCoMo、Vodafone、T-Mobile 等；设备制造商如 ASUS、HTC、Huawei、LG、Motorola、Samsung、Sony Ericsson、Toshiba 等；芯片厂商如 ARM、Broadcom、Intel、Marvell、NVIDIA、Qualcomm 等。软件厂商如 Ascender、eBay、Esmertec、LivingImage 等。
- ◆ Android 更像一款桌面环境为 Java 的 Linux 操作系统。有助于 Google 实现其"随时随地为每个人提供信息"的企业战略。

HTC Dream/G1 具体配置

硬件

3.17 英寸 HVGA (480 x 320) ; 1150mAh 电池 ; 高通 528Mhz 7201 处理器 ; 64MB RAM、128MB ROM ; 1GB MicroSD 卡 ; QWERTY 全键盘; 310 万像素摄像头。

流媒体

支持视频格式：H.264、流媒体、3GPP、MPEG4 和 Codec 3GP ; 支持音频格式：MP3、AAC、AAC+、WMA、MPEG4、WAV、MIDI、REAL、AUDIO 和 OGG; 支持墙纸格式：JPG、BMP、PNG 和 GIF ; 铃声(MP3、AAC、AAC+和 WMA)。

接入技术

蓝牙(class 1) ; 四频(850, 900, 1800, 1900); 支持 3G, 802.11b 和 802.11g。

互联网

支持 HTTP、WAP Push 和 xHTML；支持 POP、IMAP、SMTP，以及 AOL 和 GMAIL 电子邮件服务；支持 AIM、MSN、雅虎通和 GTALK；与谷歌日历同步；与 Android Market 联机；支持谷歌“街景”服务；包装盒内附数据工具包。

更多信息

<https://sites.google.com/a/android.com/opensource/release-features>



Android 盈利模式

Android 的 App Market 模式，软件开发人员获得 7 成收入，3 成用于系统维护。难点在于位置营销。设备商通过卖设备、内置特色应用来获得盈利。也可以兼职专业软件开发人员进行赢利。Google 自身通过基于统一平台为用户提供信息来盈利。

Android 的优势

- ◆ 源代码完全开放，便于开发人员更清楚的把握实现细节，便于提高开发人员的技术水平，有利于开发出更具差异性的应用。
- ◆ 采用了对有限内存、电池和 CPU 优化过的虚拟机 Dalvik，Android 的运行速度比想象的要快很多。
- ◆ 运营商（中国移动等）的大力支持，产业链条的热捧。
- ◆ 良好的盈利模式（3/7 开），产业链条的各方：运营商、制造商、独立软件生产商都可以获得不错的利益。将移动终端的评价标准从硬件向软件转变，极大的激发了软件开发者的热情。
- ◆ Android 的源代码遵循 Apache V2 软件许可，而不是通常的 GPL v2 许可。有利于商业开发。
- ◆ 具有强大的 Linux 社区的支持。

Android 的不足

- ◆ 由于采用了 Java 作为应用开发语言，目前可用的传统第三方应用还很少，但由于 Android 是一款完全开源的移动计算平台，相信第三方应用会很快丰富起来。
- ◆ Google 提供了一套 Java 核心包(J2SE 5,J2SE 6)的有限子集，尚不承诺遵守 Java 任何 Java 规范,可能会造成 Java 阵营的进一步分裂。
- ◆ 现有应用完善度不太够，需要的开发工作量较大。

- ◆ 基于 QEMU 开发的模拟器调试手段不十分丰富，只支持通话、SMS 等，速度慢。
- ◆ 暂不具备 Push Mail 和 Office(DataViz、QuickOffice 计划近期推出)功能，目前主要面向的是普通消费者用户，对商业用户支持尚弱。

Android 带来的影响

ANDROID 的推出后可能影响的产业包括移动通信业，软件开发业，手机制造业，在以消费者为核心的状态。

对消费者的影响



- ◆ 高档手机选择面增加。
- ◆ Android 在设计初期就考虑了与现其有业务的融合，改变以往从计算机为主改成从手机使用为导向。新生应用如：Google 地图及其衍生应用、GMail、GTalk 等。
- ◆ GPS 卫星导航功能，手机照相，MP3，蓝芽等均被列为 Android 所提供支持的基本选项。
- ◆ Android 的平台基本上是免费的，虽然有部份原生链接库会要求费用，但大部份是免权利金；Android 的程序可以采用 JAVA 开发，但是因为它的虚拟机(Virtual Machine) Dalvik，是将 JAVA 的 bytecode 转成自己的格式，回避掉需要付给 SUN 有关 JAVA 的授权费用。

对手机制造者的影响

- ◆ Android 是款开源的移动计算软件平台，组建了 google 主导的拥有众多产业界巨头的产业联盟，有利于高效开发、降低成本。
- ◆ 由于是源代码开放的产品，对非主导厂商而言，可以避开与主导厂商在核心技术上面的差距，开发出更具竞争力和差异化的产品。

对运营商的影响

- ◆ 丰富的数据业务，将导致数据流量的显著增加。
- ◆ 手机来源增加，价格更为低廉。

对软件开发者的影响

- ◆ 因为 Android 移动软件平台抱持开放互通的观念，势必吸引不少自由软件的拥护者。
- ◆ 开发方向有三个重点：

- 应用软件的开发
- 特殊功能的原生链接库
- 专属应用程序框架

◆ 由于 Android 的 App Market 性质，可能催生出专门的应用软件开发商。

Android 应用现状

◆ 设备商：lenovo、琦基、戴尔、三星、摩托罗拉、华为、英特尔、Kogan、索爱、华硕、多普达、爱可视、Archos 等。



◆ 制造商：HTC、Telstra 等。

◆ 手机设计公司：播思、德信无线等。

◆ 运营商：中国移动、Sprint、T-Mobile、Teleca AB 等。

◆ 芯片商：Qualcomm、Marvell、TI、Boardcom 等。

Android 开发入门

System Requirements

The sections below describe the system and software requirements for developing Android applications using the Android SDK tools included in Android 1.1 SDK, Release 1.



Supported Operating Systems

- Windows XP (32-bit) or Vista (32- or 64-bit)
- Mac OS X 10.4.8 or later (x86 only)
- Linux (tested on Linux Ubuntu Dapper Drake)

Supported Development Environments

Eclipse IDE

- [Eclipse 3.3 \(Europa\)](#), [3.4 \(Ganymede\)](#)
 - Eclipse [JDT](#) plugin (included in most Eclipse IDE packages)
 - [WST](#) (optional, but needed for the Android Editors feature; included in [most Eclipse IDE packages](#))
- [JDK 5 or JDK 6](#) (JRE alone is not sufficient)
- [Android Development Tools plugin](#) (optional)
- **Not** compatible with Gnu Compiler for Java (gcj)

Other development environments or IDEs

- [JDK 5 or JDK 6](#) (JRE alone is not sufficient)
- [Apache Ant](#) 1.6.5 or later for Linux and Mac, 1.7 or later for Windows
- **Not** compatible with Gnu Compiler for Java (gcj)

Note: If JDK is already installed on your development computer, please take a moment to make sure that it meets the version requirements listed above. In particular, note that some Linux distributions may include JDK 1.4 or Gnu Compiler for Java, both of which are not supported for Android development

什么是 Android?

Android 是一个专门针对移动设备的软件集，它包括一个操作系统，中间件和一些重要的应用程序。Beta 版的 [Android SDK](#) 提供了在 Android 平台上使用 JaVa 语言进行 Android 应用开发必须的工具和 API 接口。

特性



- **应用程序框架** 支持组件的重用与替换
- **Dalvik 虚拟机** 专为移动设备优化
- **集成的浏览器** 基于开源的 [WebKit](#) 引擎
- **优化的图形库** 包括定制的2D 图形库，3D 图形库基于 OpenGL ES 1.0 （硬件加速可选）
- **SQLite** 用作结构化的数据存储
- **多媒体支持** 包括常见的音频、视频和静态图像格式（如 MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF）
- **GSM 电话技术** （依赖于硬件）
- **蓝牙 Bluetooth, EDGE, 3G, 和 WiFi**（依赖于硬件）
- **照相机, GPS, 指南针, 和加速度计 (accelerometer)** （依赖于硬件）
- **丰富的开发环境** 包括设备模拟器，调试工具，内存及性能分析图表，和 Eclipse 集成开发环境插件

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。

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：

- 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。
- 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
- 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)。
- 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。
- 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。

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。



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。

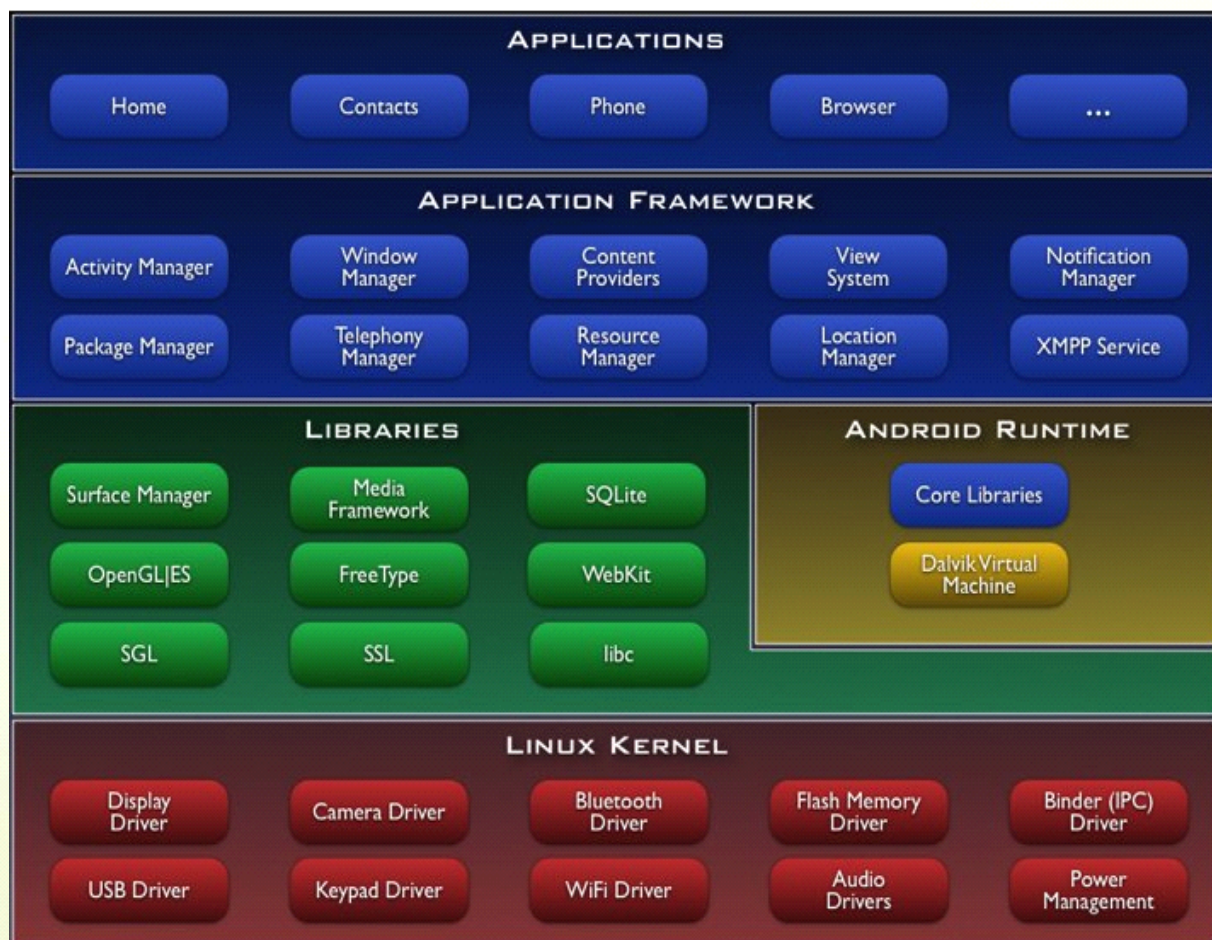
Dalvik 虚拟机依赖于 linux 内核的一些功能，比如线程机制和底层内存管理机制。

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。

Android 的系统架构

系统构架



Android 内核

- ◆ Linux 内核版本 2.6
- ◆ 位于硬件和软件堆之间的抽象层
- ◆ 核心服务：安全机制、内存管理、进程管理、网络、硬件驱动。



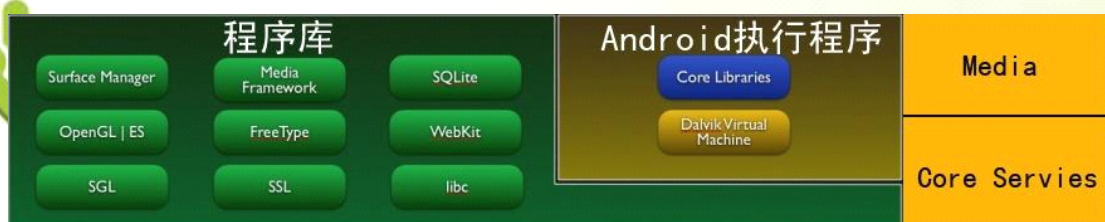
Android 依赖 Linux 内核 2.6 提供核心服务，比如安全、内存管理、进程管理、网络、硬件驱动。在这里，Linux 内核扮演的是硬件层和系统其它层次之间的一个抽象层的概念。这个操作系统并非类 GNU/Linux 的，因为其系统库，系统初始化和编程接口都和标准的 Linux 系统是有所不同的。

Android 编程基础

从 Google 目前 release 的 Linux 系统来看，其没有虚拟内存文件系统，系统所用的是 yaffs2 文件系统，具体的映像也都位于 SDK 安装目录下。通过 `emulator -console` 命令，我们可以在 host 中断下得到一个简单的可以控制 Android 的 shell，这个系统包含了一个 Toolbox，提供一些基本的命令工具，集中在 `/sbin,/system/sbin,/system/bin` 中，但是很简陋，命令种类也很少。

目前 Android 的程序安装模式是靠 Eclipse 自动进行的，通过对底层的分析可知，大致步骤就是在 `/data/app` 和 `data/data` 下存放 android 底层和普通内核没有什么大的区别，我们可以将其作为一个 Linux 来进行开发和 hacking。

Lib 和运行环境



lib

- ◆ C/C++库：被各种 Android 组件使用
- ◆ 通过应用程序框架开发者可以使用其功能
- ◆ 包括：
 - 媒体库：MPEG4 H.264 MP3 JPG PNG
 - WebKit/LibWebCore：Web 浏览引擎
 - SQLite 关系数据库引擎
 - 2D, 3D 图形库、引擎

丰富的类库支持：2D 和 3D 图像库 OpenGL ES、数据库 SQLite、对象数据库 db4o 类库、媒体库、基于 Linux 底层系统 C 库等等，让应用开发更简单多样。Google 使用 Apache 的 Harmony 类库，Harmony 某些方面速度快于 Sun 的 VM。Runtime 在 Dalvik Java VM 上，Dalvik 采用简练、高效的 byte code 格式运行，它能够在低能耗和没有应用相互干扰的情况下并行执行多个应用。

运行时环境

- ◆ 核心库提供的 Java 功能
- ◆ Dalvik 虚拟机依赖于 Linux 内核，例如线程或底层内存管理
- ◆ 设备可以运行多个 Dalvik 虚拟机，每一个 Android 应用程序在它自己的 Dalvik VM 实例中运行
- ◆ VM 执行优化的 Dalvik 可执行文件(.dex)
- ◆ Dx-工具把编译过的 Java 文件转换为 dex 文件

应用和框架

- ◆ 核心应用，例如联系人，电子邮件，电话，浏览器，日历，地图， ...
- ◆ 充分访问所有核心应用框架 API
- ◆ 简化组件的重用
- ◆ 用 Java 编写应用程序



支持的功能

+ Application framework: 可重用的和可替换的组件部分, 在这个层面上, 所有的软件都是平等的。

+ Dalvik virtul machine: 一个基于 Linux 的虚拟机。

+ Integrated browser: 一个基于开源的 WebKit 引擎的浏览器, 在应用程序层。

+ Optimized graphics: 包含一个自定义的 2D 图形库和基于 OpenGL ES 1.0 标准的 3D 实现。

+ SQLite: 数据库

+ Media support: 通用的音频, 视频和对各种图片格式的支持(MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)

+ GSM Telephony: GSM 移动网络, 硬件支持。

+ Bluetooth, EDGE, 3G, and WiFi: 都依赖于硬件支持。

+ Camera, GPS, compass, and accelerometer: 都依赖于硬件支持。

+ Rich development environment: 包含一套完整的开发工具集, 方便跟踪调试, 内存检测和性能测试, 而且提供了

Eclipse 的插件。最底层的是一个 Linux Kernel, 加载了几个移动设备必要的系统驱动 (这么说来 Android 基础系统是要以 GPL 发布了? 不知道 34 家厂商的硬件开发商们是怎么样想的); 上面是类库和 Runtime, 绿色的类库部分可以看到大名鼎鼎的 SQLite, 这个软件甚至声称自己属于公共领域(比 MIT License 还要强 @@), 字体 FreeType 是 BSD-style License 的, 图形库 OpenGL ES 只需通过产品测试, 无偿使用于产品。再向上看是应用层的东西了, 这里可以做的事情就非常多了, 各个社区, 各个厂家都可以参与进来。难怪 Android 的 sdk 可以 Apache License 发布了, 对企业和开发人员友好啊。那么 Google 自己的东西在哪里呢? 没错, 就是右边那个 runtime, 最吸引技术人员的就是这个 runtime (注意, 这个才是 Android 的核心)。Google 为它准备了一个虚拟机, 叫做 Dalvik。这个让人摸不着头脑的东西的到底是什么? 从开发平台上我们清清楚楚地得到了答案: Java

android
developers



android

android
developers

Android开发教程&笔记



android

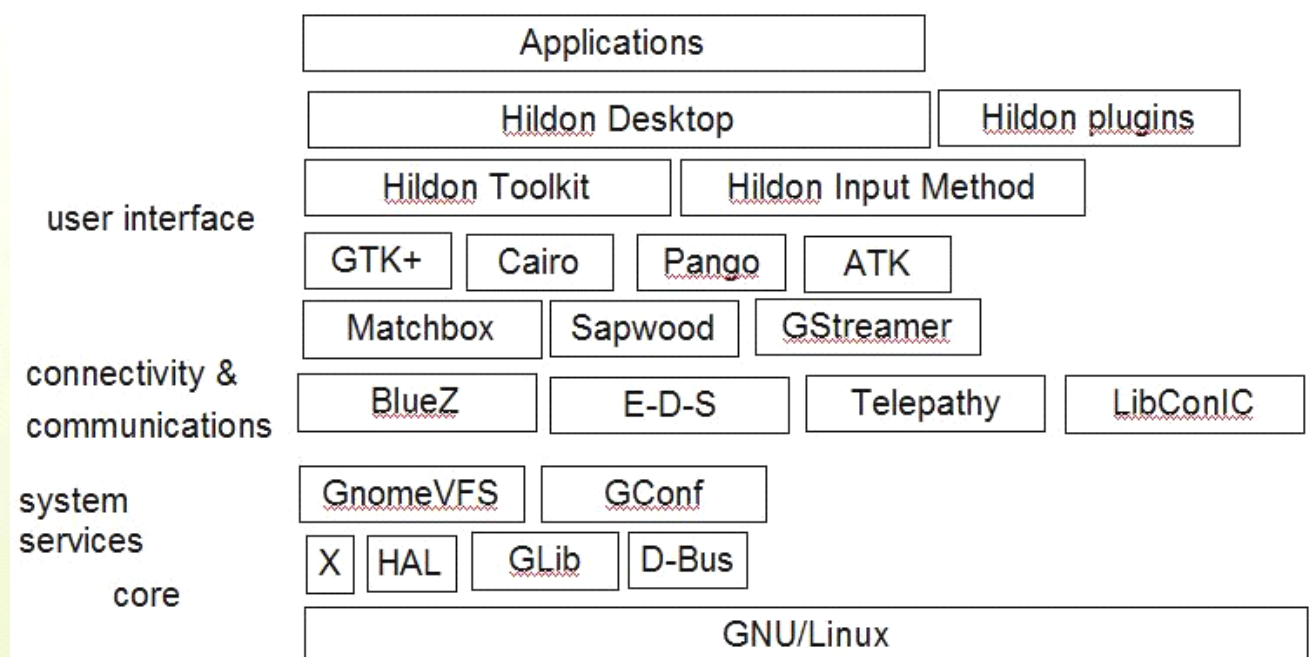
7 个 Linux 手机平台

- ◆ Maemo
- ◆ Android
- ◆ LIMO
- ◆ OpenMOKO
- ◆ GPE^2
- ◆ ALP
- ◆ QTopia Phone Edition

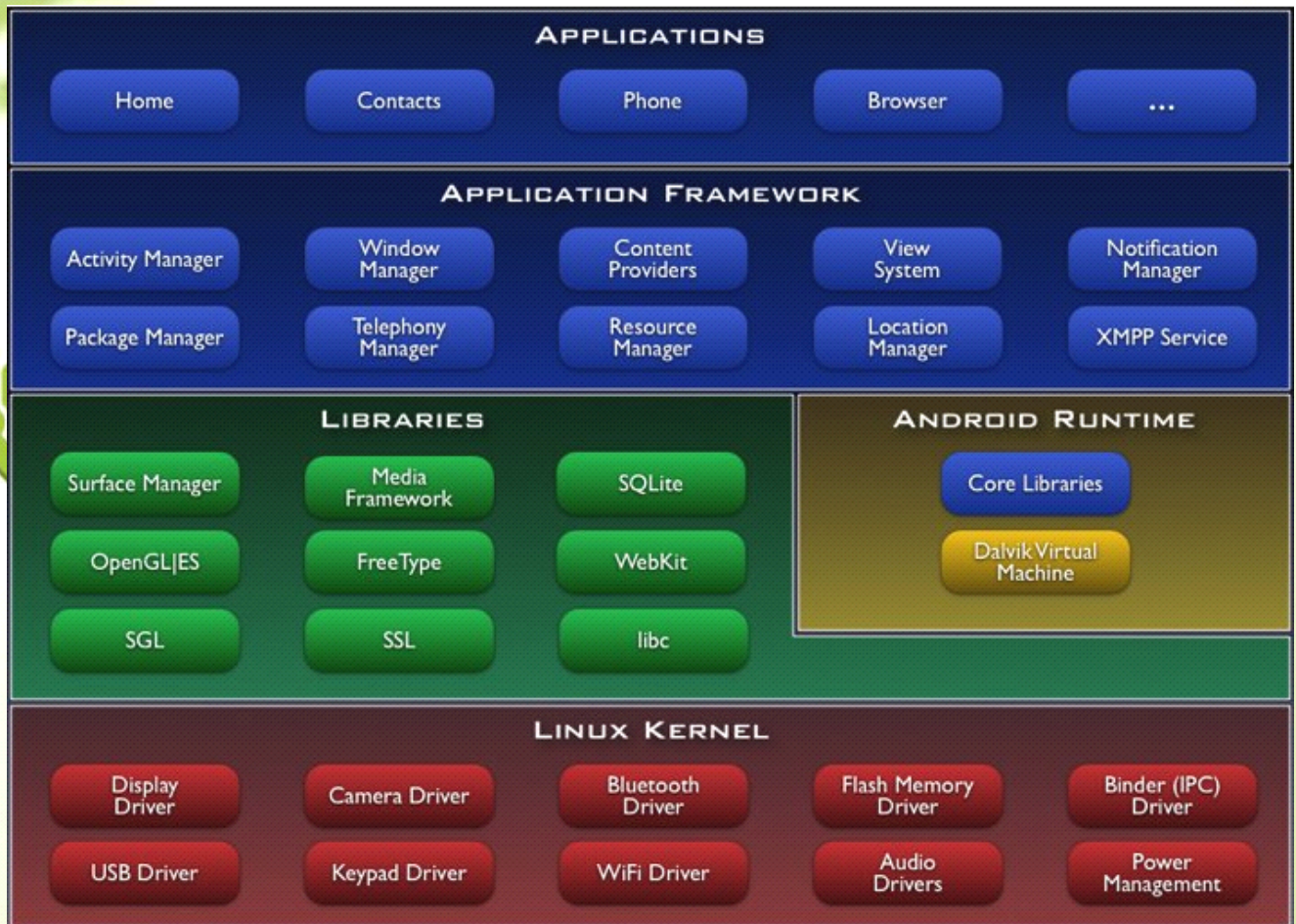


Maemo 架构

Architecture overview – logical view



Android 架构



LIMO 架构



Application Layer

Application Manager	Phone Applications	Browser	MMS/SMS	PIM Utilities	CAMERA	SIM toolkit
Setting	Contacts	Java App	Email&IM	Data Applications	Multimedia Applications	Other Applications

Application Manager & UI API

Application Engine API

Application/UI framework & Application engine Layer

Application Manager Framework	Application UI Framework	Internet Framework	Messaging Framework	Database Services	Future Expansion
-------------------------------	--------------------------	--------------------	---------------------	-------------------	------------------

Terminal Service API

Middleware Layer

System Services*	Telephony Framework	Networking Framework	Security Framework	Multimedia Framework	Database Engine	
Dev Mgmt Framework	Location Framework	Accessory Framework	Data Sync Framework	Logging Framework	DRM Framework+	Java VM

Kernel Layer

Linux Kernel*	Device Drivers	Modem Interface
---------------	----------------	-----------------

Common API

In Scope

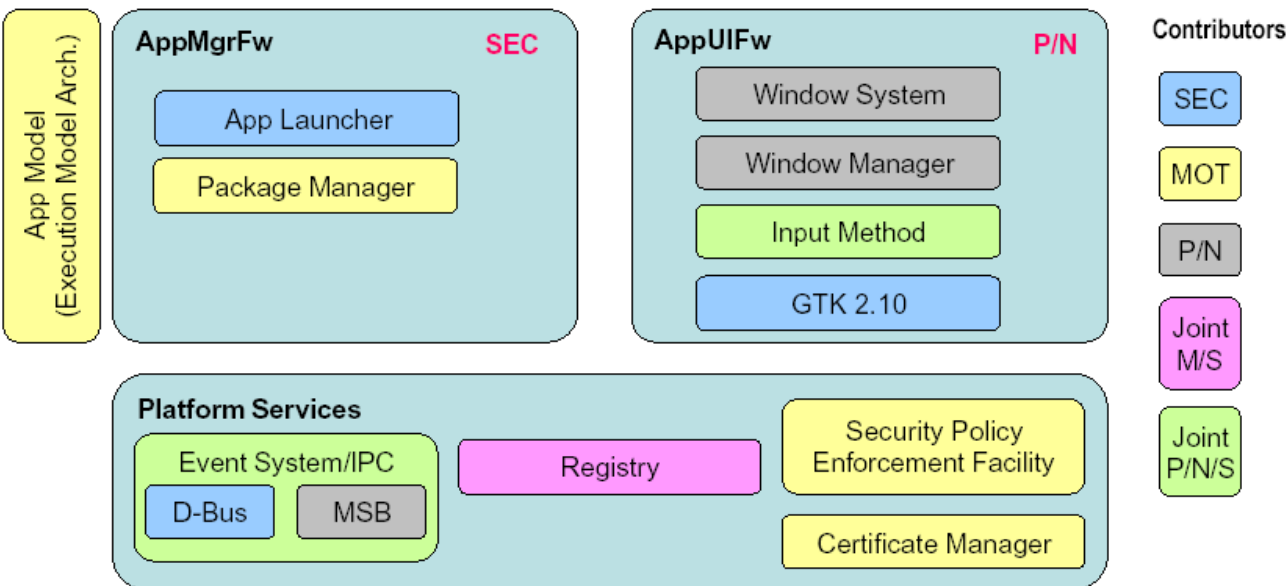
- C Only
- C & C++
- API only (No Source code)

Common code

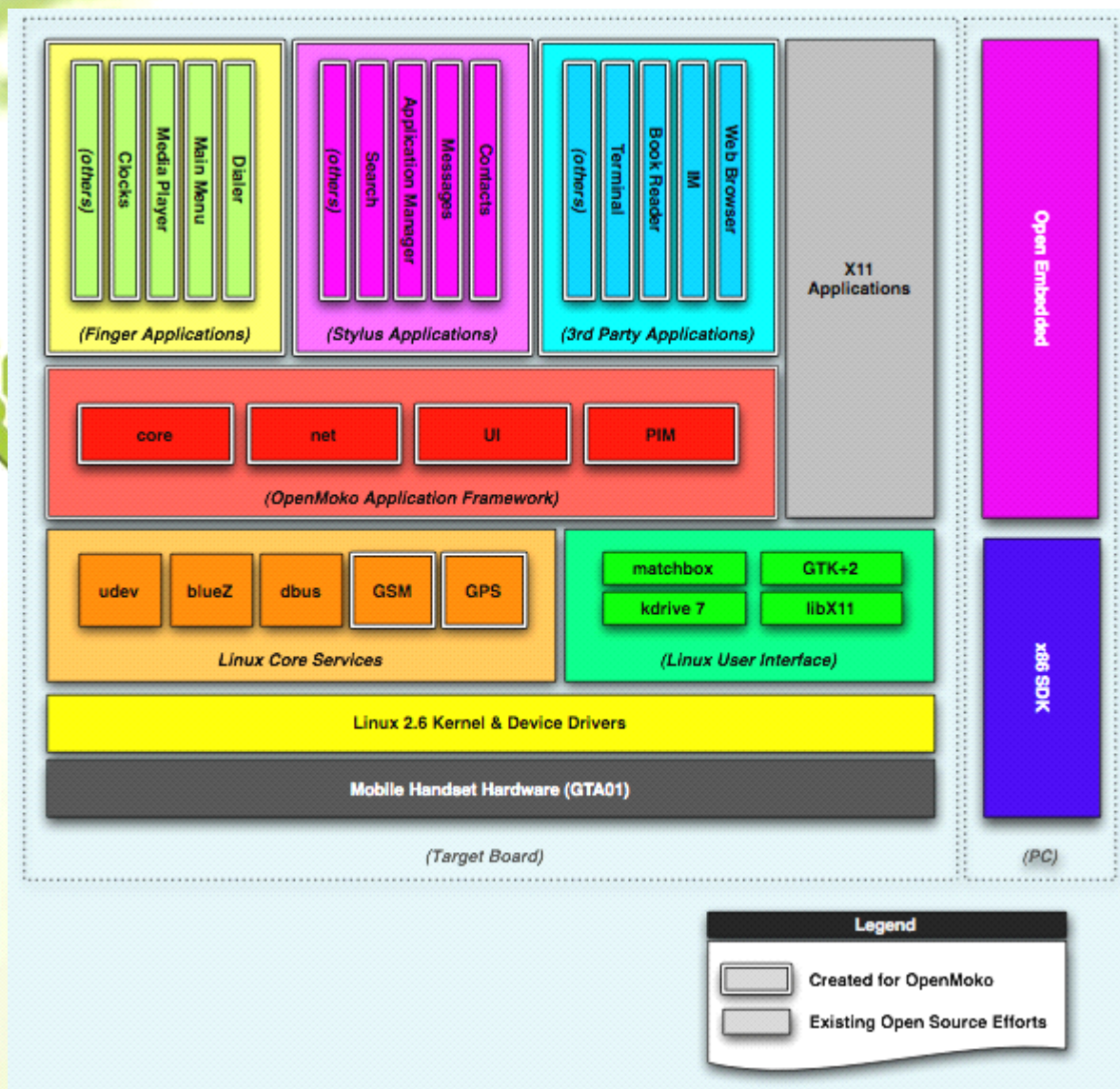
In Scope/
Founder
Contribution

Out of
Scope

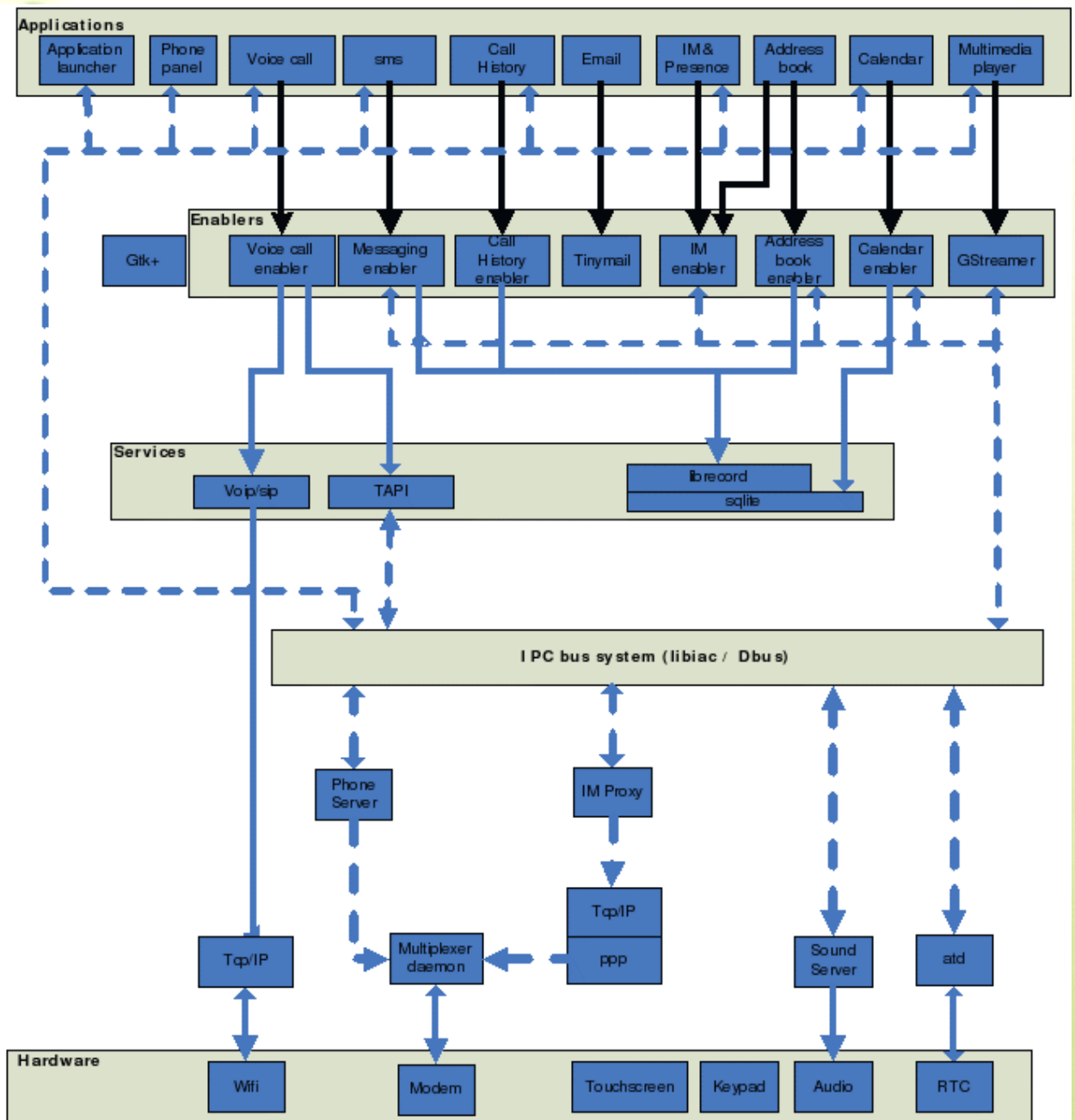
Out of
Scope+



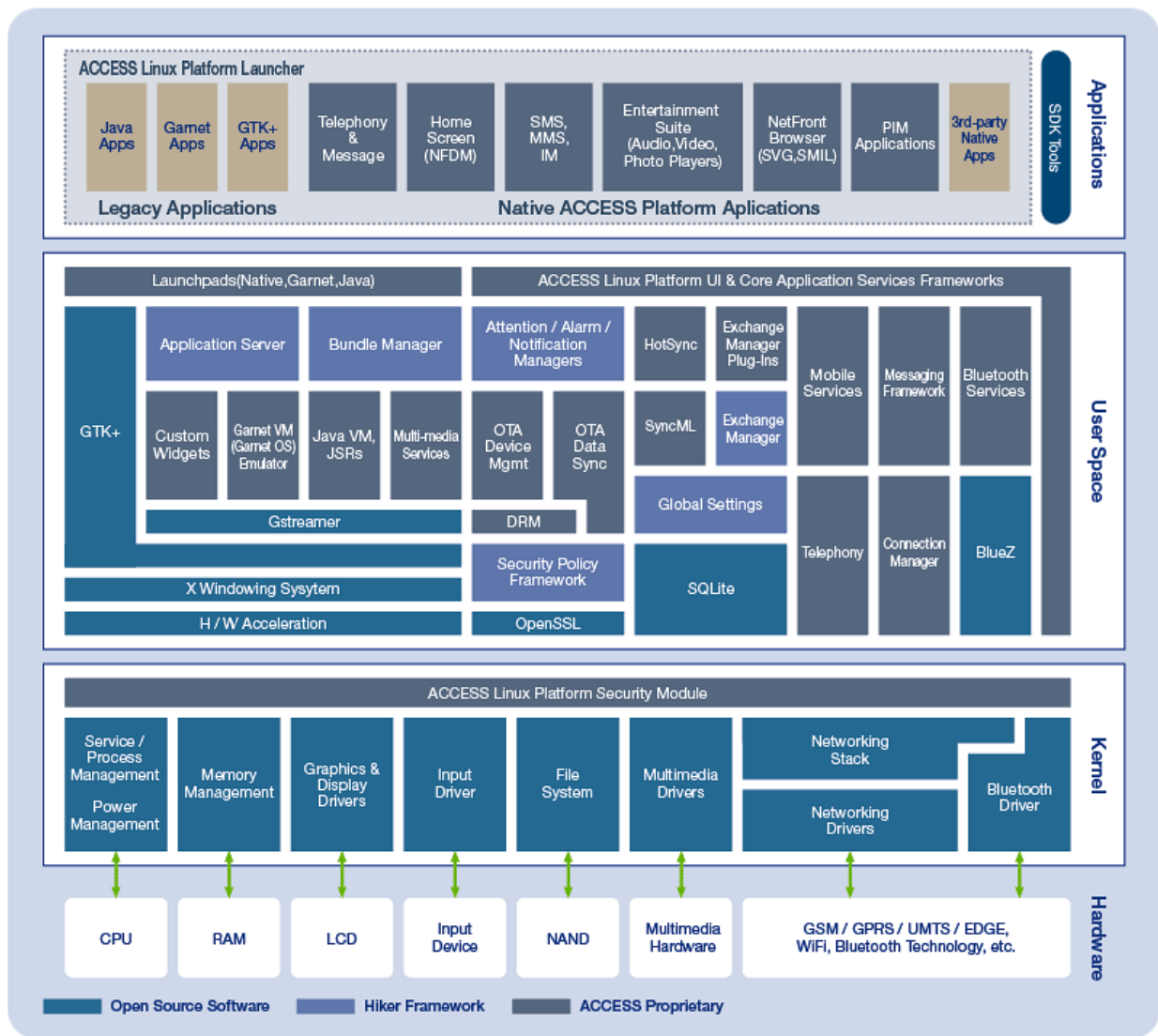
OpneMOKO 架构



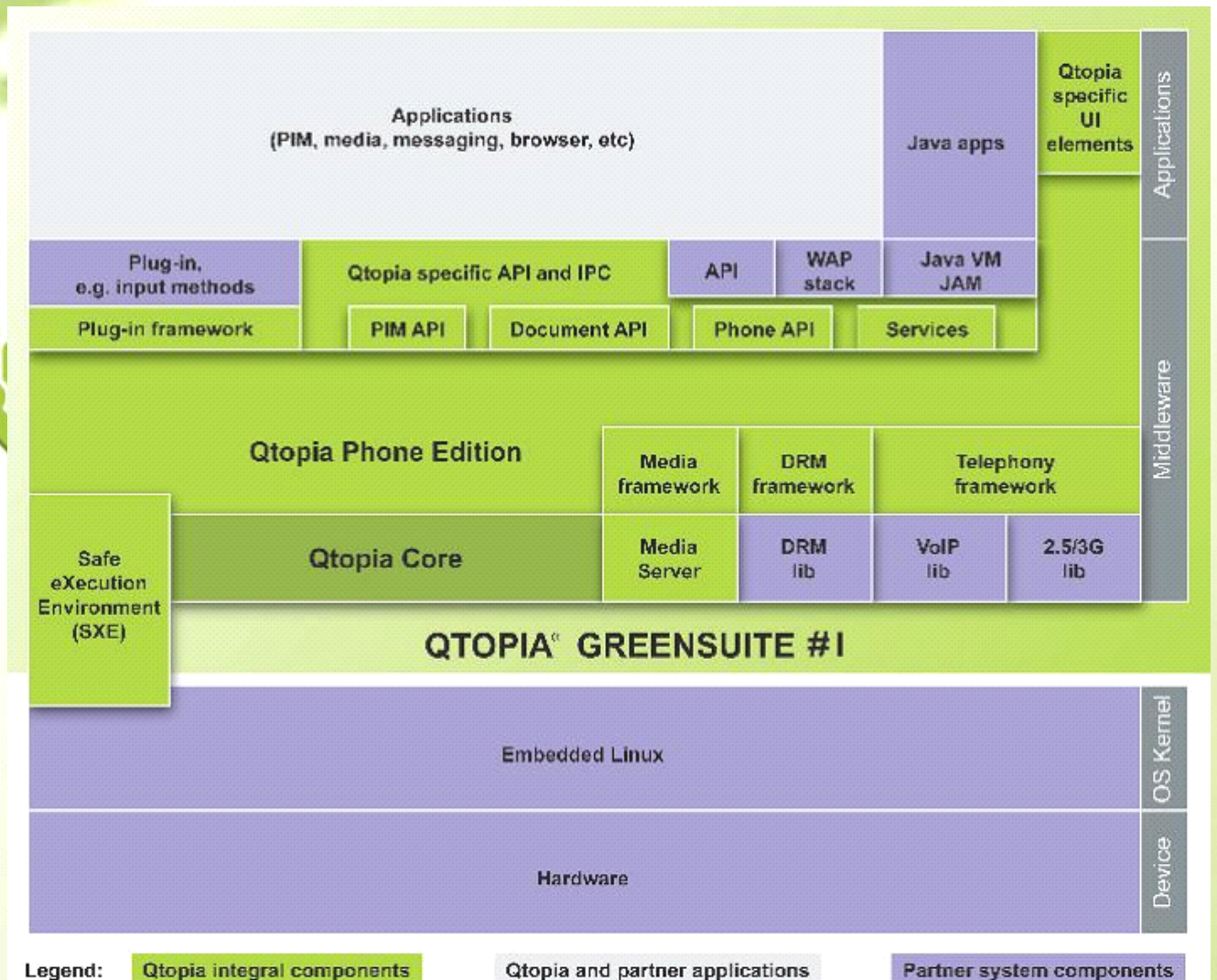
GPE^2 架构



ALP 架构



Qtopia Phone Edition 架构



进程间的通信

Linux 手机平台进程间通信

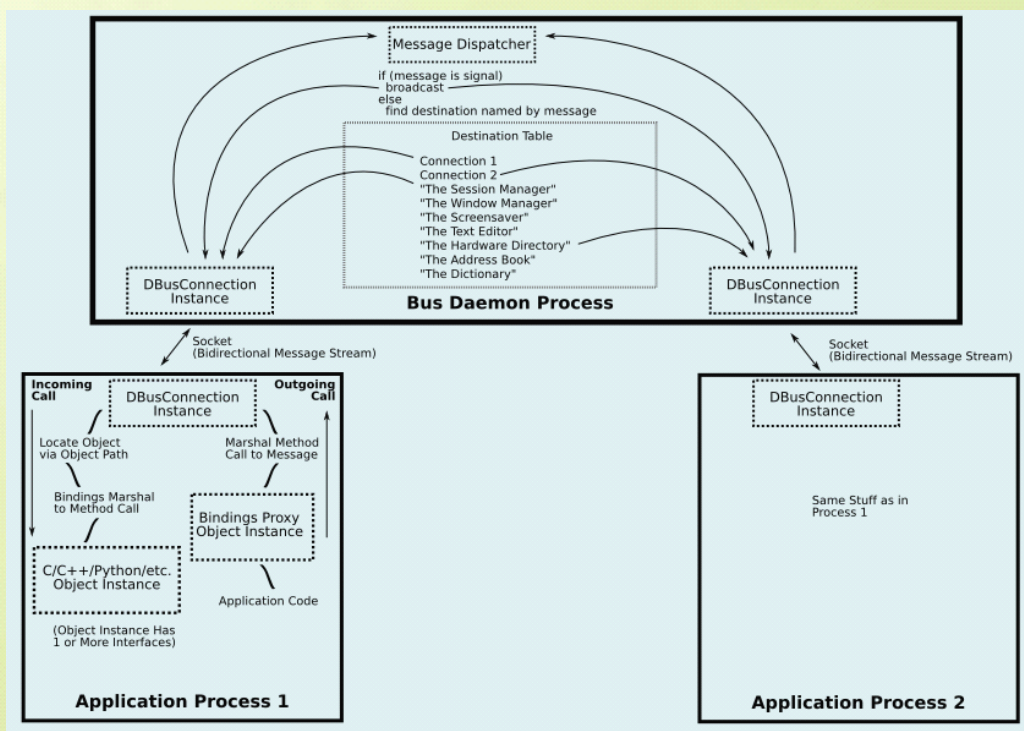
- ◆ Maemo 采用 D-BUS
- ◆ Android 采用 OpenBinder
- ◆ LiMO 采用 D-BUS
- ◆ OpenMoko 采用 D-BUS
- ◆ GPE Phone Edition 采用 D-BUS
- ◆ ALC 采用 OpenBinder
- ◆ Qtopia Phone Edition 采用 D-BUS



进程间通信种类

- ◆ D-BUS
- ◆ Openbinder
- ◆ CORBA/Corbit
- ◆ IVY
- ◆ GNET

D-BUS



Android 学习方法

- ① 了解什么是 Androi
- ② 建立开发环境
- ③ 阅读 SDK 文档
- ④ 背景知识
 - ✓ Java
 - ✓ 面向对象
 - ✓ 设计模式
 - ✓ J2ME、Brew、Symbian



建立 Android 开发环境

- ① 下载 [JDK 5 or JDK 6](#) (JRE alone is not sufficient) ->安装->设置环境变量
JAVA_HOME CLASSPATH path
- ② 下载 [Eclipse](#) 3.3 (Europa), 3.4 (Ganymede) IDE for JAVA->解压
- ③ 下载 [Android SDK](#) 解压-> path 里加入 SDK 包中的 tools 目录全路径
- ④ 下载 [ADT](#) 0.8.0 解压
- ⑤ 打开 Eclipse 安装 ADT 插件

android
developers



android

android
developers

Android开发教程&笔记



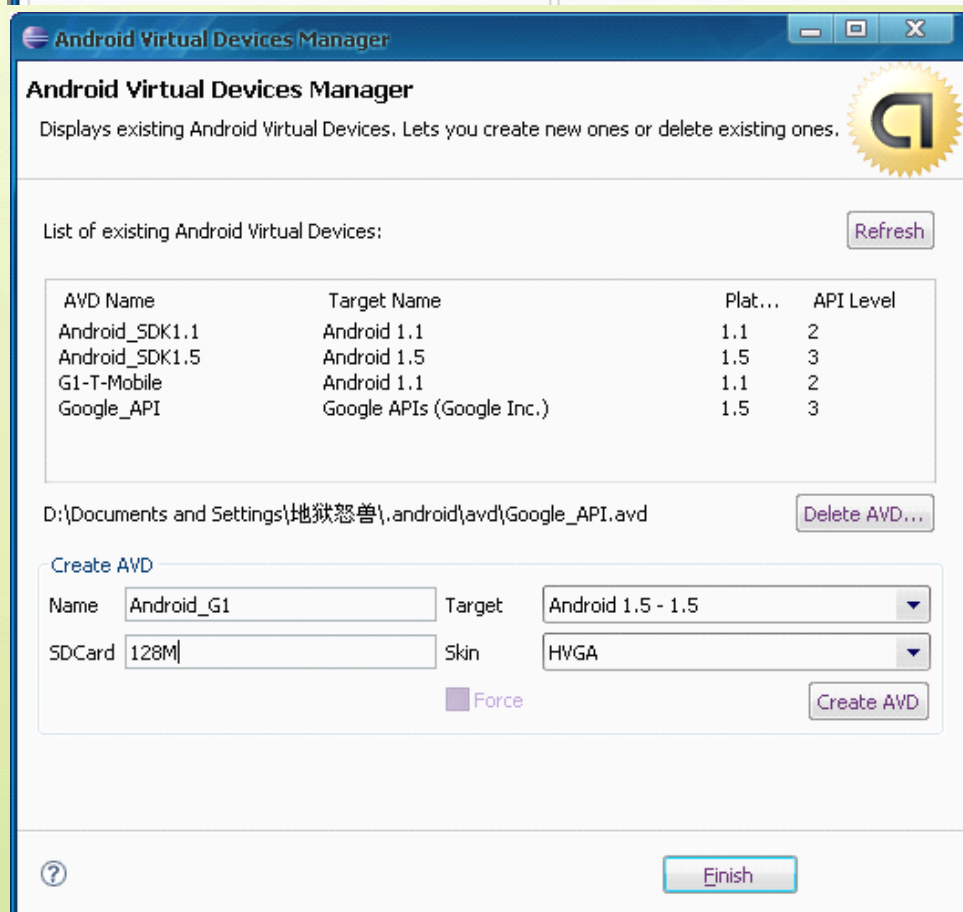
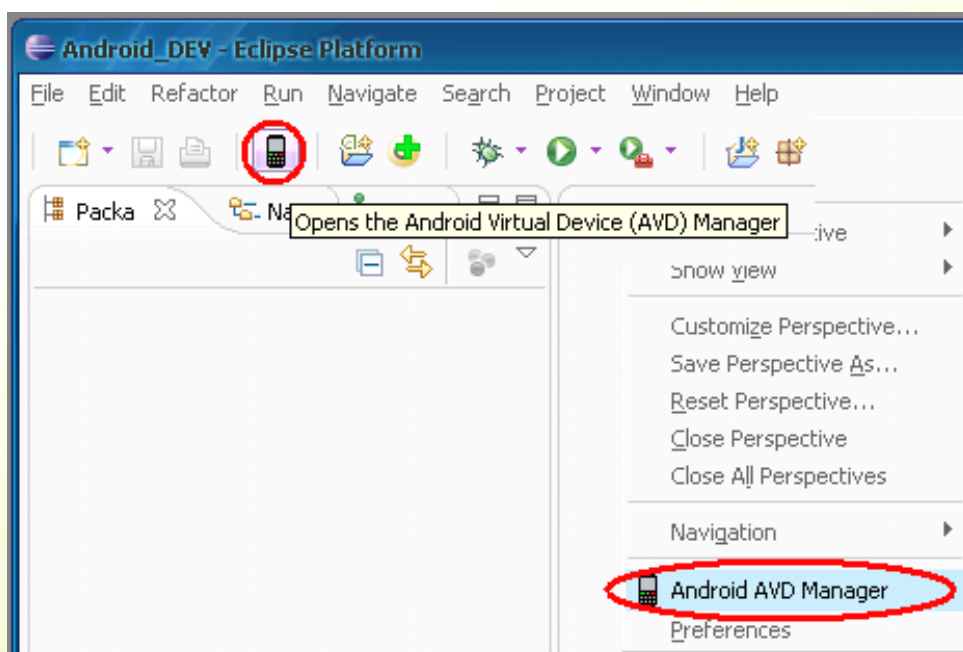
android

Android 开发环境搭建

ADV 的创建

ADT0.9.1 版本

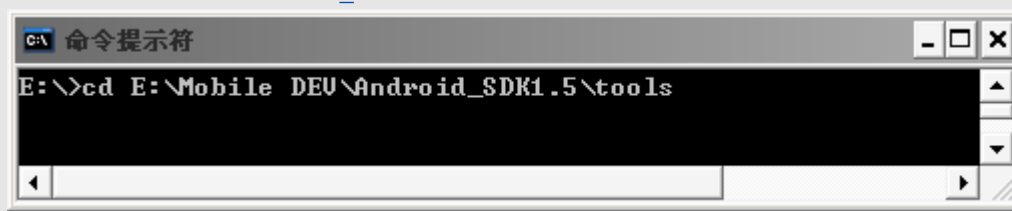
① 在 Eclipse 中创建



② 在命令行中创建

打开 CMD 命令行, 进入到 Android SDK tools 目录

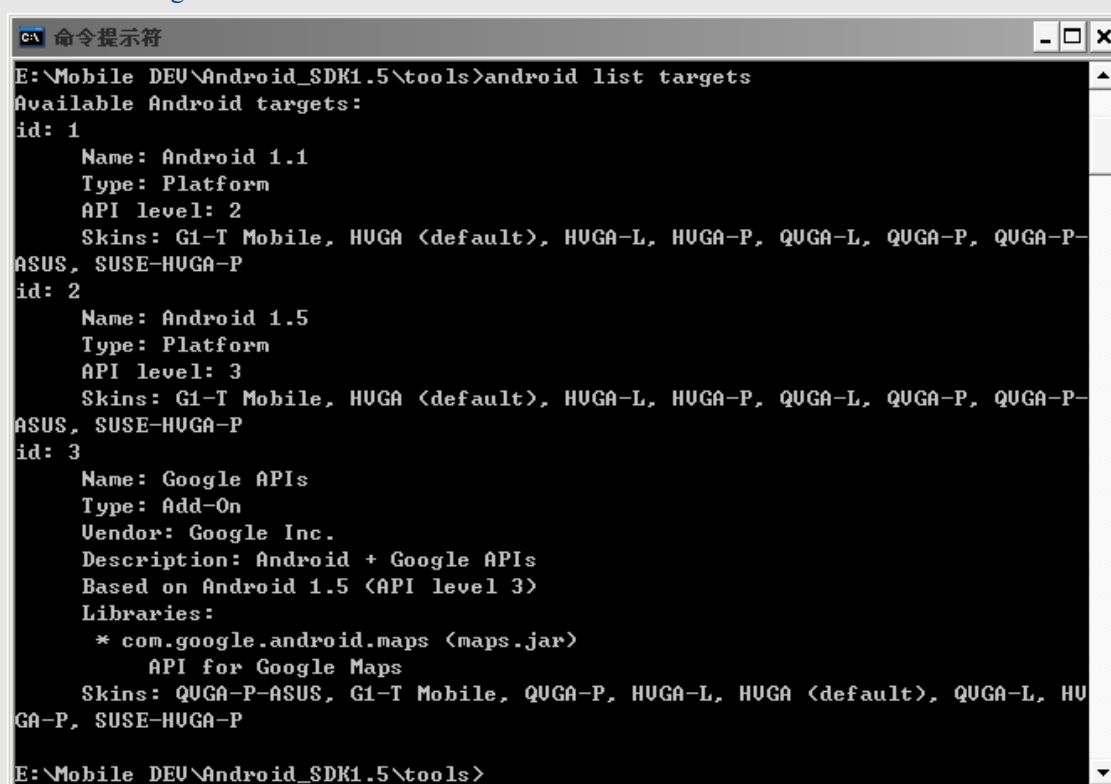
```
cd E:\Mobile DEV\Android_SDK1.5\tools
```



```
C:\ 命令提示符
E:\>cd E:\Mobile DEV\Android_SDK1.5\tools
```

使用 android 命令列出 target 值

android list targets



```
C:\ 命令提示符
E:\Mobile DEV\Android_SDK1.5\tools>android list targets
Available Android targets:
id: 1
  Name: Android 1.1
  Type: Platform
  API level: 2
  Skins: G1-T Mobile, HUGA <default>, HUGA-L, HUGA-P, QUGA-L, QUGA-P, QUGA-P-ASUS, SUSE-HUGA-P
id: 2
  Name: Android 1.5
  Type: Platform
  API level: 3
  Skins: G1-T Mobile, HUGA <default>, HUGA-L, HUGA-P, QUGA-L, QUGA-P, QUGA-P-ASUS, SUSE-HUGA-P
id: 3
  Name: Google APIs
  Type: Add-On
  Vendor: Google Inc.
  Description: Android + Google APIs
  Based on Android 1.5 (API level 3)
  Libraries:
    * com.google.android.maps <maps.jar>
      API for Google Maps
  Skins: QUGA-P-ASUS, G1-T Mobile, QUGA-P, HUGA-L, HUGA <default>, QUGA-L, HUGA-P, SUSE-HUGA-P
E:\Mobile DEV\Android_SDK1.5\tools>
```

使用 android create avd 命令来创建 AVD

行为: "create avd":

创建一个新的 Android 虚拟设备。

选项:

- t --target 新的 AVD 的 Target ID(必须)
- c --sdcard 指向一个共享的 SD 存储卡的路径或是为新的 AVD 定制的新 SD 存储卡的容量大小
- p --path 新 AVD 将被创建的位置路径
- n --name 新 AVD 的名称(必须)
- f --force 强制创建(覆盖已存在的 AVD)
- s --skin 新 AVD 的皮肤

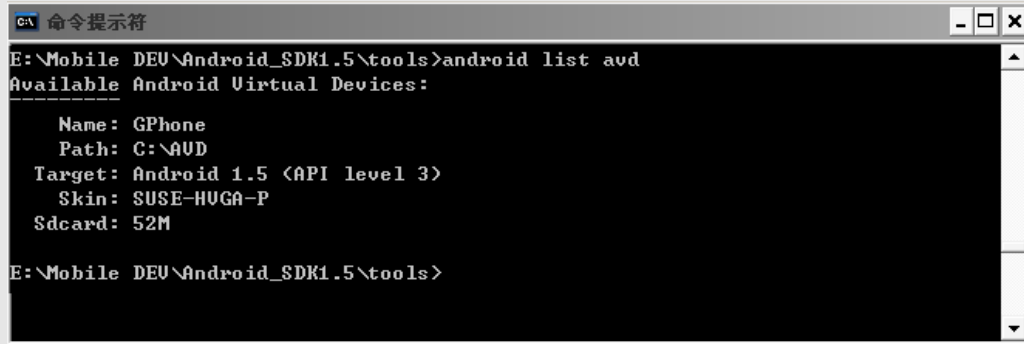
Android 编程基础

例子:将建一个名叫 GPhone 的 AVD, Target ID=2、SD 存储卡容量 52M、路径 C:\AVD\、皮肤 SUSE-HVGA-P

```
android create avd -n GPhone -t 2 -c 52M -p C:\AVD\ -s SUSE-HVGA-P
```

查看自己新创建的 ADV : list avd 命令

```
android list avd
```



```
E:\Mobile DEV\Android_SDK1.5\tools>android list avd
Available Android Virtual Devices:
-----
Name: GPhone
Path: C:\AVD
Target: Android 1.5 (API level 3)
Skin: SUSE-HVGA-P
Sdcard: 52M
E:\Mobile DEV\Android_SDK1.5\tools>
```

ADT0.9.0 版本

只能在命令行中创建

开启命令行进入 Android SDK tools 目录

```
cd E:\Mobile DEV\Android_SDK1.5\tools
```

列出 Target ID

```
andriod list target
```

创建一个新的 AVD

```
android create avd -n GPhone -t 2 -c 52M -p C:\AVD\ -s SUSE-HVGA-P
```

查看新创建的 AVD

```
android list avd
```

运行指定的 AVD

运行新创建的 AVD:GPhone

```
emulator -avd GPhone
```

Windows 平台:

Eclipse IDE 版本

-----JDK+Eclipse+Android SDK+ADT

1. 必须软件

- | | | |
|---|--------------------|-------------------------------------|
| ① | JAVA JDK SE 1.6 | jdk-6u13-windows-i586-p.exe |
| ② | Eclipse 3.4.2 | eclipse-java-ganymede-SR2-win32.zip |
| ③ | Google Android SDK | android-sdk-windows-1.5_r1.zip |
| ④ | ADT-0.9.0 | ADT-0.9.0.zip |

2. 安装过程

① 安装 JAVA JDK SE 1.6

➤ 设置环境变量

◆ JAVA_HOME

JAVA_HOME=C:\Program Files\Java\jdk1.6.0_13

◆ JAVA_JRE_HOME

JAVA_JRE_HOME=C:\Program Files\Java\jdk1.6.0_13\jre

◆ JRE_HOME

JRE_HOME=C:\Program Files\Java\jre6

◆ Android_SDK_HOME

Android_SDK_HOME=C:\Mobile Phone DEV\Android SDK

◆ CLASSPATH

CLASSPATH=.;%JAVA_HOME%\lib;%JAVA_HOME%\lib\tools.jar;%JAVA_HOME%\lib\dt.jar;%JRE_HOME%\lib;%JRE_HOME%\lib\rt.jar;%JAVA_JRE_HOME%\lib;%JAVA_JRE_HOME%\lib\rt.jar

◆ Path

要使用命令行工具必须配置

Path=%Android_SDK_HOME%\tools;%JAVA_HOME%\bin;%JRE_HOME%\bin;%JAVA_JRE_HOME%\bin;

② 解压 Eclipse 3.4.2

解压 eclipse-java-ganymede-SR2-win32.zip 到 C:\Eclipse For Android\

③ 解压 Google Android SDK

解压 android-sdk-windows-1.5_r1.zip 到 C:\Mobile Phone DEV\Android SDK

④ Eclipse 下安装 ADT 0.9.0

复制 ADT-0.9.0.zip 到 C:\

打开 C:\Eclipse For Android\eclipse.exe

设置工作路径为 C:\WorkSpace

Help->SoftWare Update->find and install ->Search for new features to install ->Next->New Archived site->选中 C:\ ADT-0.9.0.zip->OK->Finish->ADT-0.9.0.zip 选勾->Next->Accept->Next->Finish->Install All->Restart “YES”

⑤ 设置 Google Android SDK 路径

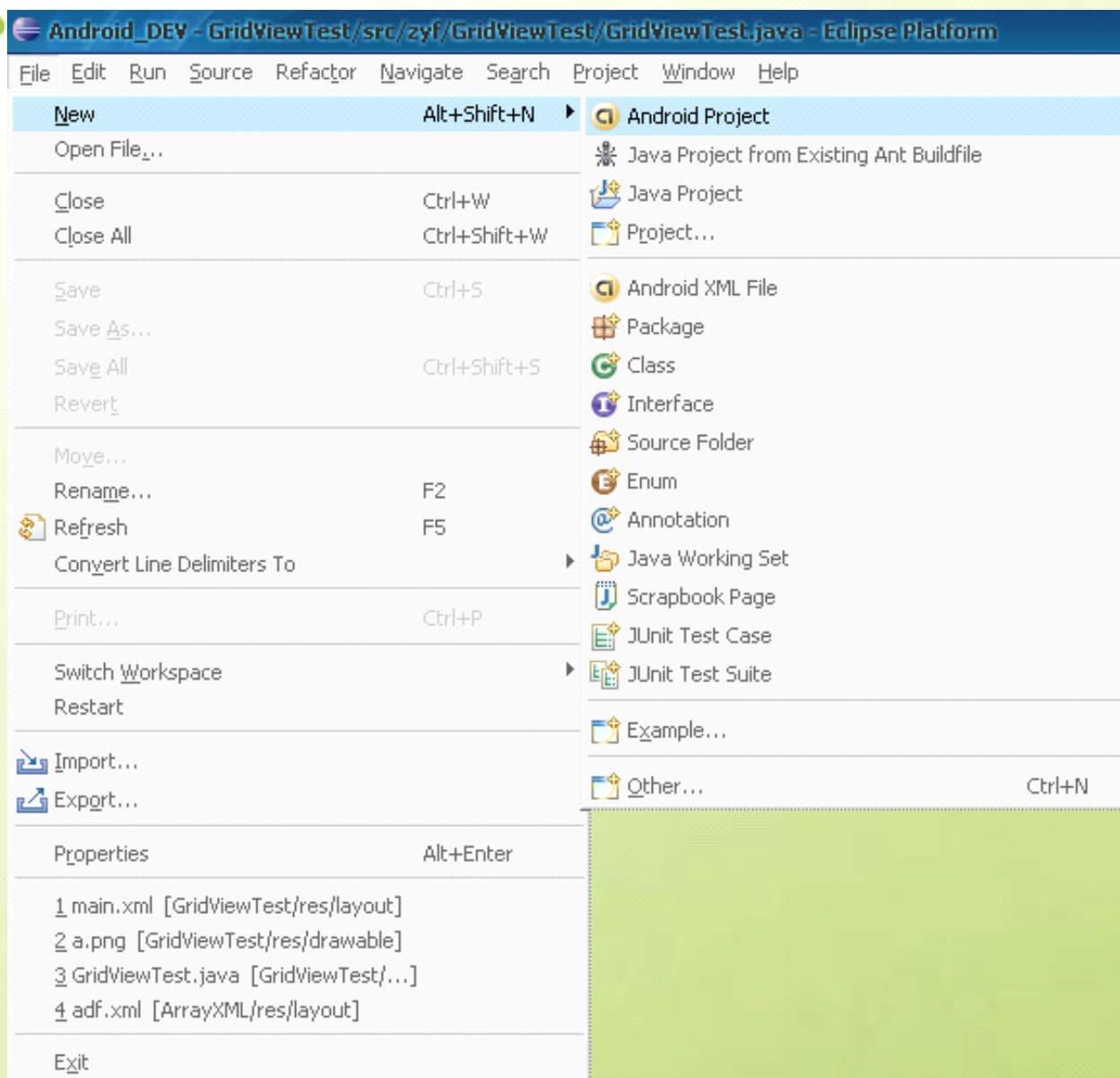
Window->preferences->选中 Android->SDK Location 中选择 Google Android SDK 的安装路径 C:\Mobile Phone DEV\Android SDK->OK



3. HelloWorld 程序实例

① 新建一个 Android Project

- Project name 设置工程名 Hello Google Android
- Package name 设置包名 zyf.android.test.hello
- Activity name 设置活动名 Hello
- Application name 设置应用程序名 Hello
- Build Target 设置 AVD API 的版本 3 Android1.5





New Android Project

Creates a new Android Project resource.

Project name:

Contents

- ☒ Create new project in workspace
- ☐ Create project from existing source
- ☒ Use default location

Location:

Build Target

Target Name	Vendor	Platform	API...
<input type="checkbox"/> Android 1.1	Android Open Source Project	1.1	2
<input checked="" type="checkbox"/> Android 1.5	Android Open Source Project	1.5	3
<input type="checkbox"/> Google APIs	Google Inc.	1.5	3

Android + Google APIs

Properties

Application name:

Package name:

☒ Create Activity:

Min SDK Version:

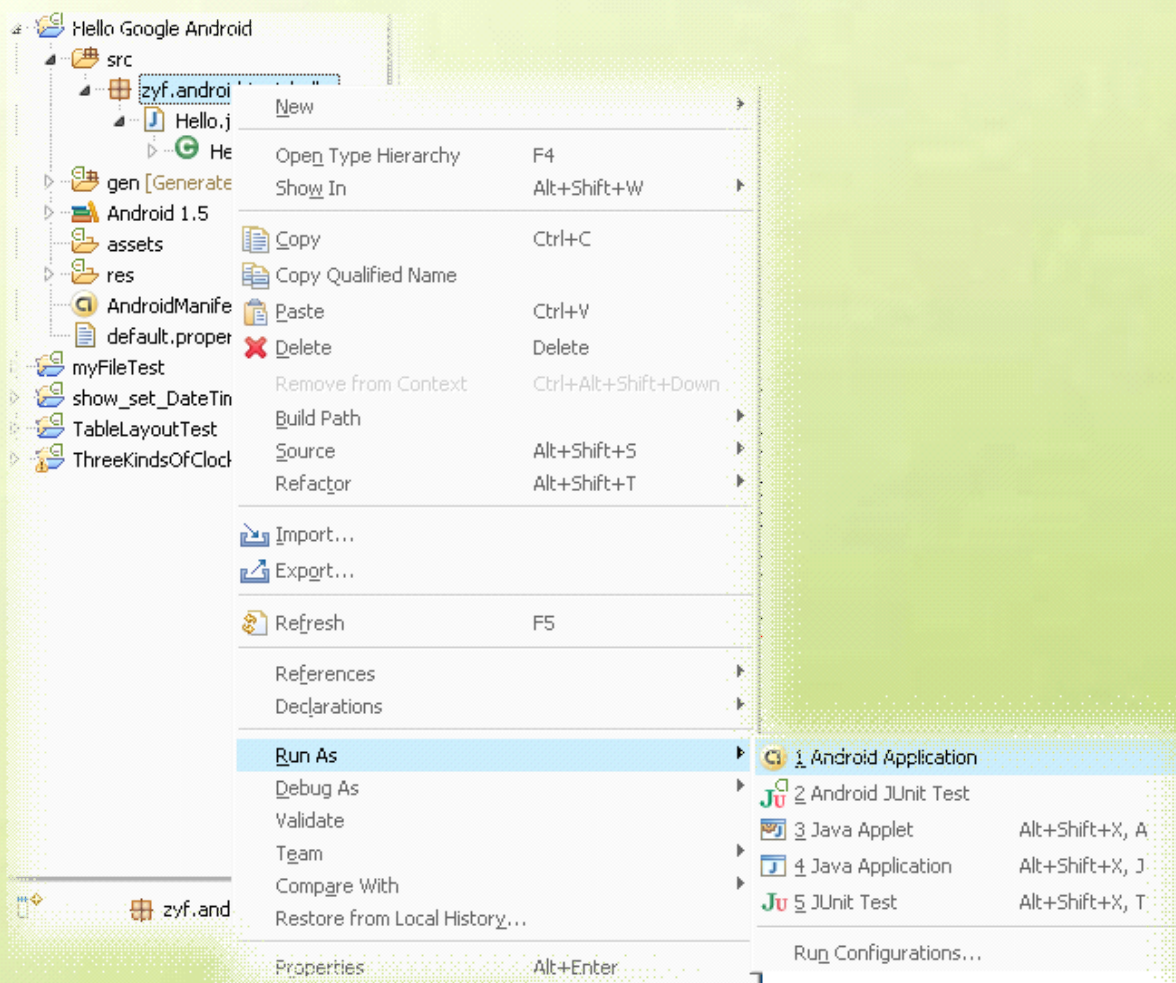
② 修改 Hello.java 文件 内容如下：

```
package zyf.android.test.hello;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
public class Hello extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // setContentView(R.layout.main);
        TextView tv = new TextView(this);

        tv.setText("这是一个测试Android的helloWorld");

        setContentView(tv);
    }
}
```

③ 运行 as Android



④ 代码分析：

在 Android 中，用户界面控件被封装成了各种 Class 叫做 Views。一个 View 是一个可以显示的控件对象，比如 RadioButton，Animation，TextLable 等。其中的一个简单的控件是 TextView：

```
TextView tv = new TextView(this);
```

传入 TextView 构造函数的参数是一个 Context 对象，通过这个对象可以使用系统提供的功能接口，比如加载资源，访问数据库和共享数据等等。Activity 类从 Context 类继承而来，所以 Activity 本身是一个 Context（Java 中的继承概念）。

TextView 对象构建以后就可以设置要显示的数据了。

```
tv.setText("这是一个测试 Android 的 helloWorld");
```

最后是连接 TextView 到屏幕，类似这样：

```
setContentView(tv);
```

setContentView() 方法可以控制具体哪一个控件和系统的 UI 联系起来（我的理解是设置为主显示 View）。如果没有设置，屏幕中将会显示空白。

⑤ 结果





Apache Ant IDE 版本

-----JDK+Android SDK +Ant

1.必须软件

- | | | |
|---|--------------------|--------------------------------|
| ① | JAVA JDK SE 1.6 | jdk-6u13-windows-i586-p.exe |
| ② | Google Android SDK | android-sdk-windows-1.5_r1.zip |
| ③ | Apache Ant | apache-ant-1.7.1-bin.zip |

2.安装过程

- ① 安装 JAVA JDK SE 1.6

➤ 设置环境变量

◆ JAVA_HOME

JAVA_HOME=C:\Program Files\Java\jdk1.6.0_13

◆ JAVA_JRE_HOME

JAVA_JRE_HOME=C:\Program Files\Java\jdk1.6.0_13\jre

◆ JRE_HOME

JRE_HOME=C:\Program Files\Java\jre6

◆ Android_SDK_HOME

Android_SDK_HOME=C:\Mobile Phone DEV\Android SDK

◆ ANT_HOME

ANT_HOME=C:\Mobile Phone DEV\Apache Ant\apache-ant-1.7.1

◆ CLASSPATH

CLASSPATH=.;%ANT_HOME%\lib;%ANT_HOME%\lib\ant.jar;%JAVA_HOME%\lib;%JAVA_HOME%\lib\tools.jar;%JAVA_HOME%\lib\dt.jar;%JRE_HOME%\lib;%JRE_HOME%\lib\rt.jar;%JAVA_JRE_HOME%\lib;%JAVA_JRE_HOME%\lib\rt.jar

◆ Path

Path=%ANT_HOME%\bin;%Android_SDK_HOME%\tools;%JAVA_HOME%\bin;%JRE_HOME%\bin;%JAVA_JRE_HOME%\bin;



② 解压 Google Android SDK

解压 android-sdk-windows-1.5_r1.zip
到 C:\Mobile Phone DEV\Android SDK

③ 解压 apache-ant-1.7.1.zip

解压 Apache Ant apache-ant-1.7.1.zip
到 C:\Mobile Phone DEV\Apache Ant\apache-ant-1.7.1



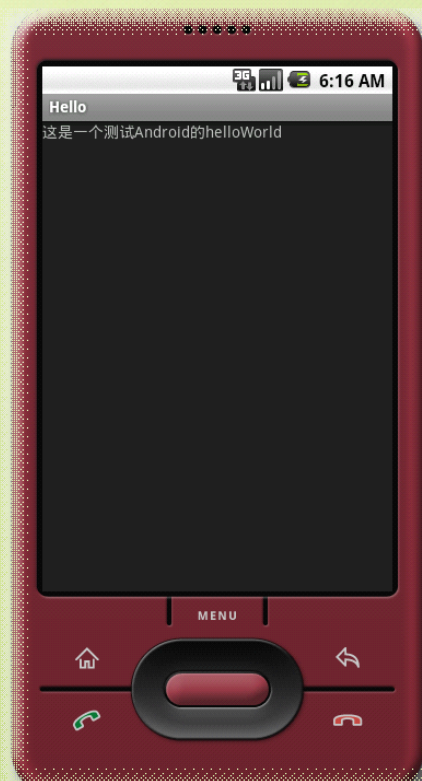
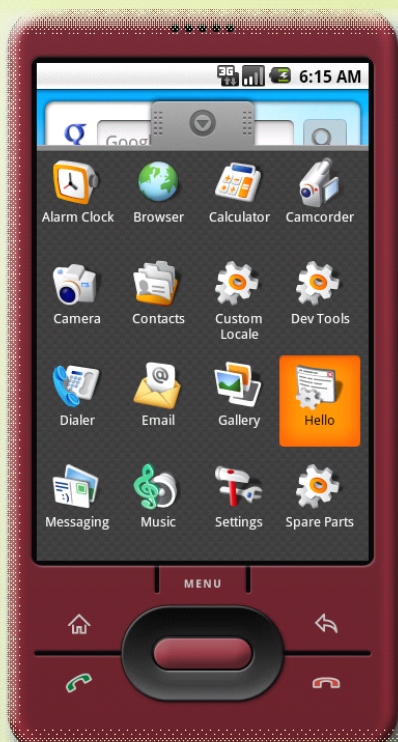
3.HelloWorld 程序实例

- ① 开始->运行->cmd
- ② cd C:\Mobile Phone DEV\WorkSpace
- ③ 使用命令行工具来创建一个新工程

```
android create project -k zyf.hello -n HelloAndroid -t 2 -a AntActivity -p ./Hello
```

- ④ cd Hello
- ⑤ ant debug
- ⑥ cd bin
- ⑦ emulator -avd Android_SDK1.5
- ⑧ adb install ./hello-debug.apk
- ⑨ 在模拟器中运行 hello 程序

结果



Linux 平台:

JDK+Eclipse+Android SDK+ADT

JDK+Android SDK +Ant



应用解析

Activity :

活动是最基本的 Android 应用程序组件，应用程序中，一个活动通常就是一个单独的屏幕。每一个活动都被实现为一个独立的类，并且从活动基类中继承而来，活动类将会显示由视图控件组成的用户接口，并对事件做出响应。大多数的应用是由多个屏幕显示组成。例如：一个文本信息的应用也许有一个显示发送消息的联系人列表屏幕，第二个屏幕用来写文本消息和选择收件人，再来一个屏幕查看消息历史或者消息设置操作等。这里每一个这样的屏幕就是一个活动，很容易实现从一个屏幕到一个新的屏幕并且完成新的活动。在某些情况下当前的屏幕也许需要向上一个屏幕活动提供返回值--比如让用户从手机中挑选一张照片返回通讯录做为电话拨入者的头像。

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。

简单理解 Activity 代表一个用户所能看到的屏幕，Activity 主要是处理一个应用的整体性工作，例如，监听系统事件(按键事件、触摸屏事件等)、为用户显示指定的 View，启动其他 Activity 等。所有应用的 Activity 都继承于 android.app.Activity 类，该类是 Android 提供的基层类，其他的 Activity 继承该父类后，通过 Override 父类的方法来实现各种功能，这种设计在其他领域也较为常见。

Intent :

调用 Android 专有类 Intent 进行架构屏幕之间的切换。Intent 是描述应用想要做什么。Intent 数据结构两个最重要的部分是动作和动作对应的数据。典型的动作类型有:MAIN (活动的门户)、VIEW、PICK、EDIT 等。而动作对应的数据则以 URI 的形式进行表示。例如:要查看某个人的联系方式，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。

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将会做什么事。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。

与之有关系的一个类叫 IntentFilter。相对于 intent 是一个有效的做某事的请求，一个 intentfilter 则用于描述一个 activity (或者 IntentReceiver) 能够操作哪些 intent。一个 activity 如果要显示一个人的联系方式时，需要声明一个 IntentFilter，这个 IntentFilter 要知道怎么去处理 VIEW 动作和表示一个人的 URI。IntentFilter 需要在 AndroidManifest.xml 中定义。

通过解析各种 intent，从一个屏幕导航到另一个屏幕是很简单的。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：

- A、Activities 能够重复利用从其它组件中以 Intent 的形式产生的一个请求；
- B、Activities 可以在任何时候被一个具有相同 IntentFilter 的新的 Activity 取代。

IntentReceiver:

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 `IntentReceiver`。虽然 `IntentReceiver` 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。`IntentReceiver` 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 `intentreceiver` 被触发时, 你的应用不必对请求调用 `intentreceiver`, 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 `intentreceiver` 广播给其它应用程序。



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。

Content Provider :

`Android` 应用程序能够将它们的数据保存到文件、`SQLite` 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, 内容提供者就可以发挥作用了。因为内容提供者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内容提供者处理的各种数据类型。

数据是应用的核心。在 `Android` 中, 默认使用鼎鼎大名的 `SQLite` 作为系统 DB。但是在 `Android` 中, 使用方法有点小小的不一样。在 `Android` 中每一个应用都运行在各自的进程中, 当你的应用需要访问其他应用的数据时, 也就需要数据在不同的虚拟机之间传递, 这样的情况操作起来可能有些困难(正常情况下, 你不能读取其他的应用的 `db` 文件), `ContentProvider` 正是用来解决在不同的应用包之间共享数据的工具。

- 所有被一个 `Android` 应用程序创建的偏好设置, 文件和数据库都是私有的。
- 为了和其他应用程序共享数据, 应用程序不得不创建一个 `Content Provider`
- 要回索其他应用程序的数据, 它自己的 `Content Provider` 必须被调用
- `Android` 本地 `Content Provider` 包括:
 - `CallLog`: 地址和接收到的电话信息
 - `Contact.People.Phones`: 存储电话号码
 - `Setting.System`: 系统设置和偏好设置
 - 等等

android
developers



android

android
developers

Android开发教程&笔记



android

Android 虚拟机 Dalvik

Dalvik 冲击

随着 Google 的 AndroidSDK 的发布，关于它的 API 以及在移动电话领域所带来的预期影响这些方面的讨论不胜枚举。不过，其中的一个话题在 Java 社区是一石激起千层浪，这就是 Android 平台的基础——Dalvik 虚拟机。



Dalvik 和标准 Java 虚拟机(JVM)首要差别

Dalvik 基于寄存器，而 JVM 基于栈。基于寄存器的虚拟机对于更大的程序来说，在它们编译的时候，花费的时间更短。

Dalvik 和 Java 运行环境的区别

Dalvik 经过优化，允许在有限的内存中同时运行多个虚拟机的实例，并且每一个 Dalvik 应用作为一个独立的 Linux 进程执行。独立的进程可以防止在虚拟机崩溃的时候所有程序都被关闭。

Dalvik 形势

Dalvik 的诞生也导致人们开始忧虑 Java 平台的第一次大规模的分道扬镳或许已经是进行时了——有人已经把 Dalvik 和微软的 JVM 以及 Sun 对微软的诉讼联系起来，等着看 Google 身上是否也会发生类似事情；另外一些人则指出，Google 并没有宣称 Dalvik 是一个 Java 实现，而微软却是这样做的。Sun 也对可能带来的阵营分裂表达了忧虑情绪，并提出和 Google 合作来保证 Dalvik 和 JVM 之间的兼容性——Google 对此的解释是，Dalvik 是对解决目前 JavaME 平台上分裂的一次尝试，也是为了提供一个拥有较少限制许可证的平台。甚至还有人怀疑这是否是 Sun 和 Google 两大阵营对 Java 之未来的一次大规模较量。

Android 中各种 JAVA 包的功能描述

在 Android 的应用程序开发中，通常使用的是 JAVA 语言，除了需要熟悉 JAVA 语言的基础知识之外，还需要了解 Android 提供的扩展的 JAVA 功能。

在一般的 JAVA 应用中，如果需用引用基础类库，通常需要使用如下的方式：

```
import javax.swing.*;
```

以上代码表示了引用 JAVA 的 GUI 组件 `Swing`，`javax.swing` 即 JAVA 中的一个包。

android 提供一些扩展的 JAVA 类库，类库分为若干个包，每个包中包含若干个类。



重要包的描述:

`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

`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

Android 的相关文件类型

Java 文件——应用程序源文件

android 本身相当一部分都是用 java 编写而成(基本上架构图里头蓝色的部份都是用 Java 开发的), android 的应用必须使用 java 来开发。



Class 文件——Java 编译后的目标文件

不像 J2se, java 编译成 class 就可以直接运行, android 平台上 class 文件不能直接在 android 上运行。由于 Google 使用了自己的 Dalvik 来运行应用, 所以这里的 class 也肯定不能在 AndroidDalvik 的 java 环境中运行, android 的 class 文件实际上只是编译过程中的中间目标文件, 需要链接成 dex 文件后才能在 dalvik 上运行。

Dex 文件——Android 平台上的可执行文件

Android 虚拟机 Dalvik 支持的字节码文件格式 Google 在新发布的 Android 平台上使用了自己的 Dalvik 虚拟机来定义, 这种虚拟机执行的并非 Java 字节码, 而是另一种字节码: dex 格式的字节码。在编译 Java 代码之后, 通过 Android 平台上的工具可以将 Java 字节码转换成 Dex 字节码。虽然 Google 称 Dalvik 是为了移动设备定做的, 但是业界很多人认为这是为了规避向 sun 申请 Javalicense。这个 DalvikVM 针对手机程式/CPU 做过最佳化, 可以同时执行许多 VM 而不会占用太多 Resource。

Apk 文件——Android 上的安装文件

Apk 是 Android 安装包的扩展名, 一个 Android 安装包包含了与某个 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打成一个压缩包。一个工程只能打进一个.apk 文件。

Android 的应用程序结构分析：HelloActivity

本例以一个简单的 HelloActivity 程序为例，简单介绍 Android 应用程序的源代码结构。事实上，Android 应用程序虽然不是很复杂，但是通常涉及了 JAVA 程序,XML 文件，Makefile 多方面的内容。HelloActivity 虽然简单，但是麻雀虽小，五脏俱全，是学习 Android 应用程序的最好示例。

第一部分：HelloActivity 的源代码

HelloActivity 工程的源代码在 Android 目录的 development/samples/HelloActivity/中，代码的结构如下所示：

```
development/samples/HelloActivity/  
|-- Android.mk  
|-- AndroidManifest.xml  
|-- res  
|   |-- layout  
|   |   `-- hello_activity.xml  
|   |-- values  
|   |   `-- strings.xml  
|-- src  
|   |-- com  
|   |   |-- example  
|   |   |   |-- android  
|   |   |   |   |-- helloactivity  
|   |   |   |   |   |-- HelloActivity.java  
|-- tests  
|   |-- Android.mk  
|   |-- AndroidManifest.xml  
|   |-- src  
|   |   |-- com  
|   |   |   |-- android  
|   |   |   |   |-- helloactivity  
|   |   |   |   |   |-- HelloActivityTest.java
```

其中 tests 是一个独立的项目，可以暂时不考虑。其他部分看作一个 Android 的一应用程序的工程。这个工程主要的组成部分如下所示：

AndroidManifest.xml：工程的描述文件，在运行时有用处

Android.mk：整个工程的 Makefile

res: 放置资源文件的目录

src/com/example/android/helloactivity/HelloActivity.java: 这是 JAVA 类文件, 这个文件的路径表示在 Andorid 的 JAVA 包的结构中的位置, 这个包的使用方式为 com.example.android.helloactivity。

第二部分: 编译的中间结果

这个 HelloActivity 工程经过编译后将生成

out/target/common/obj/APPS/HelloActivity_intermediates/ 目录, 这个目录中的内容都是 HelloActivity 工程相关的, 更具体地说都与 development/samples/HelloActivity/ 中的 Android.mk 文件相关。



```
out/target/common/obj/APPS/HelloActivity_intermediates/
|-- classes.dex (字节码)
|-- classes.jar (JAR 文件)
|-- public_resources.xml (根据 resources 结构生成的 xml)
  |-- src
    |-- R.stamp
  |-- com
    |-- example
      |-- android
        |-- helloactivity
          |-- R.java (resources 生成的文件)
```

classes.dex

是一个最重要的文件, 它是给 Android 的 JAVA 虚拟机 Dalvik 运行的字节码文件。

classes.jar

是一个 JAR 文件, JAR 的含义为 Java ARchive, 也就是 Java 归档, 是一种与平台无关的文件格式, 可将多个文件合成一个文件。解压缩之后的目录结构: (JAVA 标准编译得到的类)


```
classes
|-- META-INF
|   |-- MANIFEST.MF
|-- com
|   |-- example
|       |-- android
|           |-- helloactivity
|               |-- HelloActivity.class
|               |-- R$attr.class
|               |-- R$id.class
|               |-- R$layout.class
|               |-- R$string.class
|               |-- R.class
```

各个以 class 为扩展名的文件，事实上是 JAVA 程序经过编译后的各个类的字节码。

第三部分： 目标 **apk** 文件

目标 apk 文件是 Android 的 JAVA 虚拟机 Dalvik 安装和运行的文件，事实上这个 apk 文件将由编译的中间结果和原始文件生成。apk 文件的本质是一个 zip 包。这个 APK 包解压缩后的目录结构如下所示：

```
out/target/product/generic/obj/APPS/HelloActivity_intermediates/package.apk_FILES/
|-- AndroidManifest.xml
|-- META-INF
|   |-- CERT.RSA
|   |-- CERT.SF
|   |-- MANIFEST.MF
|-- classes.dex
|-- res
|   |-- layout
|       |-- hello_activity.xml
|-- resources.arsc
```

值得注意的是，这里的 xml 文件经过了处理，和原始的文件不太一样，不能按照文本文件的方式阅读。

第四部分： 源代码的各个文件

Android.mk 是整个工程的 “Makefile”，其内容如下所示：

- ◆ LOCAL_PATH:= \$(call my-dir)
- ◆ include \$(CLEAR_VARS)
- ◆ LOCAL_MODULE_TAGS := samples
- ◆ # Only compile source java files in this apk.
- ◆ LOCAL_SRC_FILES := \$(call all-java-files-under, src)
- ◆ LOCAL_PACKAGE_NAME := HelloActivity
- ◆ LOCAL_SDK_VERSION := current
- ◆ include \$(BUILD_PACKAGE)
- ◆ # Use the following include to make our test apk.
- ◆ include \$(call all-makefiles-under,\$(LOCAL_PATH))



这个文件在各个 Android 的工程中都是类似的，其中 LOCAL_PACKAGE_NAME 表示了这个包的名字。LOCAL_MODULE_TAGS 表示了模块的标，在这里使用的是 samples，正式的应用程序（packages 目录中的应用）中多使用 eng development。

AndroidManifest.xml 是这个 HelloActivity 工程的描述文件，其内容如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.helloactivity">
    <application android:label="Hello, Activity!">
        <activity android:name="HelloActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

其中 package 用于说明这个包的名称，android:label 中的内容是表示这个应用程序在界面上显示的标题，activity 中的 android:name 表示这个 Android 的活动的名称。

Android 编程基础

文件 `src/com/example/android/helloactivity/HelloActivity.java` 是程序主要文件，由 JAVA 语言写成

```
package com.example.android.helloactivity;
import android.app.Activity;
import android.os.Bundle;
public class HelloActivity extends Activity {
    public HelloActivity() {
    }
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.hello_activity);
    }
}
```

`com.example.android.helloactivity` 表示的是这个包的名称，在文件的头部引入了两个包 `android.app.Activity` 是一个 Android 活动（Activity）包，每一个 Android 活动都需要继承 Activity 类。

包 `android.os.Bundle` 用于映射字符串的值。

`onCreate()` 是一个重载的函数，在这个函数中实现应用程序创建的所执行的过程。其中 `setContentView()` 设置当前的视图（View）。

设置的方法是使用一个文件，这个文件因此决定了视图中包含的内容。这里使用的是 `R.layout.hello_activity`，表示从 `res/layout/` 目录中使用 `hello_activity.xml` 文件。

`res/layout/hello_activity.xml` 文件的内容如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<EditText xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/text"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:textSize="18sp"
    android:autoText="true"
    android:capitalize="sentences"
    android:text="@string/hello_activity_text_text" />
```

其中定义了一个可编辑的文本（EditText），下面的各项其实是它的各种属性，`android:text` 表示这个文本的内容，`string/hello_activity_text_text` 表示找到相应的文件，也就是 `res/value/string.xml` 文件中的 `hello_activity_text_text` 文本。

`res/value/string.xml` 的内容如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello_activity_text_text">He llo, World!</string>
</resources>
```

`hello_activity_text_text` 文本被 `res/layout/hello_activity.xml` 文件引用，正是应用程序运行时在屏幕显示的文本。

Android ADB 工具使用

adb(Android Debug Bridge)是 Android 提供的一个通用调试工具，借助这个工具，我们可以管理设备或手机模拟器的状态。

adb 功能操作：

- ◆ 快速更新设备或手机模拟器中的代码，如应用或 Android 系统升级
- ◆ 在设备上运行 shell 命令
- ◆ 管理设备或手机模拟器上预定端口
- ◆ 在设备或手机模拟器上复制、粘贴文件



adb 常用操作：

安装应用到模拟器

```
adb install app.apk
```

Android 没有提供一个卸载应用的命令，只能手动删除：

```
adb shell  
cd data/app  
rm app.apk
```

进入设备或模拟器的 Shell

```
adb shell
```

通过以上命令，可以进入设备或模拟器的 shell 环境中，在这个 Linux Shell 中，你可以执行各种 Linux 的命令，另外如果只想执行一条 shell 命令，可以采用以下方式：

```
adb shell [command]
```

如：

```
adb shell dmesg
```

会打印出内核的调试信息

发布端口

可以设置任意的端口号，做为主机向模拟器或设备的请求端口。如：

```
adb forward tcp:5555 tcp:8000
```


复制文件

可向一个设备或从一个设备中复制文件

- ◆ 复制一个文件或目录到设备或模拟器上：

```
adb push
```

如：

```
adb push test.txt /tmp/test.txt
```

- ◆ 从设备或模拟器上复制一个文件或目录

```
adb pull
```

如：

```
adb pull /android/lib/libwebcore.os
```



搜索/等待模拟器、设备实例

取得当前运行的模拟器、设备的实例列表及每个实例的状态 | 等待正在运行的设备

```
adb devices
```

```
adb wait-for-device
```

查看 Bug 报告

```
adb bugreport
```

记录无线通讯日志

无线通讯记录日志非常多，在运行时没必要记录，可以通过命令设置记录

```
adb shell
```

```
logcat -b radio
```

获取设备 ID 和序列号

```
adb get-product
```

```
adb get-serialno
```

访问数据库 SQLite3

```
adb shell
```

```
sqlite3
```

android
developers



android

android
developers

Android开发教程&笔记



android

Android 模拟器

模拟器参数

参数格式

```
emulator [option] [-qemu args]
```

option 选项

-sysdir <dir>	为模拟器在<dir>目录中搜索系统硬盘镜像
-system <file>	为模拟器从<file>文件中读取初始化系统镜像
-datadir <dir>	设置用户数据写入的目录
-kernel <file>	为模拟器设置使用指定的模拟器内核
-ramdisk <file>	设置内存 RAM 镜像文件 (默认为<system>/ramdisk.img)
-image <file>	废弃, 使用 -system <file> 替代
-init-data <file>	设置初始化数据镜像 (默认为 <system>/userdata.img)
-initdata <file>	和 "-init-data <file>"使用方法一致
-data <file>	设置数据镜像 (默认为 <datadir>/userdata-qemu.img)
-partition-size <size>	system/data 分区容量大小 (MB)
-cache <file>	设置模拟器缓存分区镜像 (默认为 零时文件)
-no-cache	禁用缓存分区
-nocache	与"-no-cache"使用方法相同
-sdcard <file>	指定模拟器 SDCard 镜像文件 (默认为 <system>/sdcard.img)
-wipe-data	清除并重置用户数据镜像 (从 initdata 拷贝)
-avd <name>	指定模拟器使用 Android 虚拟设备
-skindir <dir>	设置模拟器皮肤 在<dir>目录中搜索皮肤 (默认为 <system>/skins 目录)
-skin <name>	选择使用给定的皮肤
-no-skin	不适用任何模拟器皮肤
-noskin	使用方法与"-no-skin"相同
-memory <size>	物理 RAM 内存大小(MB)
-netspeed <speed>	设置最大网络下载、上传速度
-netdelay <delay>	网络时延模拟
-netfast	禁用网络形态
-tarce <name>	代码配置可用
-show-kernel	显示内核信息
-shell	在当前终端中使用根 Shell 命令
-no-jni	Dalvik 运行时禁用 JNI 检测
-nojni	使用方法与"-no-jni"相同
-logcat <tag>	输出给定 tag 的 Logcat 信息



-no-audio	禁用音频支持
-noaudio	与"-no-audio"用法相同
-audio <backend>	使用指定的音频 backend
-audio-in <backend>	使用指定的输入音频 backend
-audio-out <backend>	使用指定的输出音频 backend
-raw-keys	禁用 Unicode 键盘翻转图
-radio	重定向无线模式接口到个性化设备
-port <port>	设置控制台使用的 TCP 端口
-ports <consoleport>,<adbport>	设置控制台使用的 TCP 端口和 ADB 调试桥使用的 TCP 端口
-onion <image>	在屏幕上层使用覆盖 PNG 图片
-onion-alpha <%age>	指定上层皮肤半透明度
-onion-rotation 0 1 2 3	指定上层皮肤旋转
-scale <scale>	调节模拟器窗口尺寸(三种: 1.0-3.0、dpi、auto)
-dpi-device <dpi>	设置设备的 resolution (dpi 单位) (默认 165)
-http-proxy <proxy>	通过一个 HTTP 或 HTTPS 代理来创建 TCP 连接
-timezone <timezone>	使用给定的时区, 而不是主机默认的
-dns-server <server>	在模拟系统上使用给定的 DNS 服务
-cpu-delay <cpudelay>	调节 CUP 模拟
-no-boot-anim	禁用动画来快速启动
-no-window	禁用图形化窗口显示
-version	显示模拟器版本号
-report-console <socket>	向远程 socket 报告控制台端口
-gps <device>	重定向 GPS 导航到个性化设备
-keyset <name>	指定按键设置文件名
-shell-serial <device>	根 shell 的个性化设备
-old-system	支持旧版本(pre 1.4)系统镜像
-tcpdump <file>	把网络数据包捕获到文件中
-bootchart <timeout>	bootcharting 可用
-qemu args....	向 qemu 传递参数
-qemu -h	显示 qemu 帮助
-verbose	和"-debug-init"相同
-debug <tags>	可用、禁用调试信息
-debug-<tag>	使指定的调试信息可用
-debug-no-<tag>	禁用指定的调试信息
-help	打印出该帮助文档
-help-<option>	打印出指定 option 的帮助文档
-help-disk-images	关于硬盘镜像帮助
-help-keys	支持按钮捆绑(手机快捷键)
-help-debug-tags	显示出-debug <tag>命令中的 tag 可选值
-help-char-devices	个性化设备说明
-help-environment	环境变量
-help-keyset-file	指定按键绑定设置文件
-help-virtula-device	虚拟设备管理

Android 编程基础

-help-sdk-images	当使用 SDK 时关于硬盘镜像的信息
-help-build-images	当构建 Android 时，关于硬盘镜像的信息
-help-all	打印出所有帮助



进程:

在 Android 中，进程完全是应用程序的实现细节，不是用户一般想象的那样。

它们的用途很简单:

- ◆ 通过把不信任或是不稳定的代码放到其他进程中来提高稳定性或是安全性
- ◆ 通过在相同的进程中运行多个 .apk 代码来减少消耗
- ◆ 通过把重量级代码放入一个分开的进程中来帮助系统管理资源。该分开的进程可以被应用程序的其他部分单独地杀死
- ◆ 如果两个没有共享相同的用户 ID 的 .apk 试图在相同的进程中运行，这将被不允许，并且系统会为每一个 apk 程序创建不同的进程



线程

- ◆ Android 让一个应用程序在单独的线程中，指导它创建自己的线程
- ◆ 应用程序组件（Activity、service、broadcast receiver）所有都在理想的主线程中实例化
- ◆ 没有一个组件应该执行长时间或是阻塞操作(例如网络呼叫或是计算循环)当被系统调用时，这将中断所有在该进程的其他组件
- ◆ 你可以创建一个新的线程来执行长期操作

Android 释放手机资源，进程释放优先级

当系统资源消耗，Android 将会杀死一些进程来释放资源。

进程优先级顺序：

① 前台进程：

包含一个前台 Activity、包含一个正在运行的广播接收器、正在运行的服务（当前用户所需的 Activity、正在屏幕顶层运行的 Activity）

② 可视进程：

包含一个可视化的 Activity（Activity 可视的，但是不是在前台的（onPause））、例如显示在一个前台对话框之后的以前的 Activity）

③ 服务进程：

包含一个被开启的服务(处理服务，不是直接可视，例如媒体播放器，网络上传、下载)

④ 后台进程：

包含一个不可视的 Activity(带有一个当前不可视的 Activity、可以在任意时刻杀死该进程来回收内存)

⑤ 空进程

没有持有任何应用程序组件



Android 应用开发 1

分析 Hello Android

打开 Hello Android 工程

src 文件夹 源文件	HelloAndroid.java 主程序文件	R.java 资源文件	Android Library Java 库	Assets 文件夹 静态文件 打包
----------------	----------------------------	----------------	---------------------------	-----------------------

res 文件夹		
drawable 文件夹 程序图标 (ico.png)	layout 文件夹 布局 UI (main.xml)	values 文件夹 程序用到的 String、颜色**(string.xml)

AndroidManifest.xml		
描述应用程序、构成、组件、权限		
bin 文件夹		
classes.dex 编译的 java 二进制码	HelloAndroid.apk Android 安装包(APK 包)	自定义的包文件夹 存放编译后的字节码文件

Main.xml

```

<LinearLayout></LinearLayout> 整体布局 表示线性布局
xmlns:android="http://schemas.android.com/apk/res/android" 名字空间
android:orientation="vertical" 控件布局 垂直往下布局
android:layout_width="fill_parent"
android:layout_height="fill_parent" 上层控件填充满

<TextView
    android:layout_width="fill_parent" 横向填充满
    android:layout_height="wrap_content" 纵向按实际高度填充
    android:text="@string/hello" 要引用到的hello字符串
/> 图形空间 派生于 View

<Button
    android:id="@+id/widget40_button_OK" button控件ID
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" 按实际宽度高度显示填充
    android:text="OK"
></Button>

```

R.java

通过 res 文件夹下的 xml 文件定义自动生成的，main.xml ico.png string.xml 是配套的关联，进行修改后 R.java 自动重新生成

AndroidManifest.xml

有关版本，程序信息，java 包，程序图标，程序记录信息等。

Manifest.xml 文件轮廓



```
<manifest>
  <uses-permission>
  <permission>
  <application>
    <activity>
      <intent-filter>
        <action>
        <category>
        <data>
    <service>
    <receiver>
    <provider>
```


添加编辑框与按钮



```
package zyf.Study.AndroidSturdyByMyself;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
public class AndroidSturdyByMyself extends Activity {
    private EditText getNameEditText;
    private Button button_Login;
    private TextView show_Login_TextView;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        getNameEditText=(EditText) findViewById(R.id.widget29_getName_EditText);
        button_Login=(Button) findViewById(R.id.widget30_Login_Button);
        show_Login_TextView=(TextView) findViewById(R.id.widget31_showLogin_TextView);
        button_Login.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                show_Login_TextView.setText(getNameEditText.getText()+"欢迎您进入");
            }
        });
    }
}
```

使用 Intent 启动另一个 Activity

```
Intent showNextPage_Intent=new Intent();
showNextPage_Intent.setClass(UsingBundel.this,NextPageActivity.class);
startActivity(showNextPage_Intent);
```

在多个 **Activity** 之间切换时候，注意每个 **Activity** 都应在 **AndroidManifest.xml** 中有所声明定义（如下）



```
<application android:label="@string/app_name"
    android:icon="@drawable/chinazphone">
    <activity android:name=".UsingBundel"
        android:label="@string/app_name">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".NextPageActivity"
        android:label="@string/nextpage"></activity>
</application>
```

新的 **Activity**
在 **AndroidManifest.xml** 中
必须定义声明

在不同 Task 中启动 Activity

```
Intent.FLAG_ACTIVITY_NEW_TASK
```


Android 应用开发 2

Activity

何谓 Activity:

最简单的就是你可以把 Activity 看成一个 User Interface Program, 原则上它会提供使用者一个交互式的接口功能, 那一个 Activity 只有一个 UI 吗? 非也, 举例来说: 一个 email 程序, 就可能包含三个 Activity:



- 邮件列表的 Activity
- 显示邮件内容的 Activity
- 写新邮件或回复邮件的 Activity

◆ 所有的 Activity 在系统里由 Activity 堆栈所管理, 当一个新的 Activity 被执行后, 它将会被放置到堆栈的最顶部, 并且变成 running Activity, 而先前的 Activity 原则上还是会存在于堆栈中, 但它此时不会是在前景的情况, 除非刚刚那个新的 Activity 离开。

Intent 与 Intent filters

Intent:

Android 使用了一个很特别的类别 Intent, 用来从一个画面跳另一个画面, Intent 是用来描述一个程序想要做些什么事情。在 Intent 的数据结构中有两个很重要的部分, 一个是动作 (action) 及对数据产生反应 (data to act upon)。Action 主要的内容有 MAIN (程序的入口点), VIEW, PICK, EDIT 等等。Data 则是用 URI 的形式来表示。比如: 想要查看一个人的联络数据时, 你需要建立一个 Intent, 它包含了 VIEW 的动作 (action) 及指向该人数据的 URI 描述句。

Intent Filter :

当 Intent 要求做某件事时, IntentFilter 被用来描述这个 Activity 能够做些什么事情。比如一个 Activity 要能够显示个人联络数据, 你就必需要在 intentFilter 说明你要如何处理个人联络数据, 并用 ACTION_VIEW 呈现出来。IntentFilter 都会在 AndroidManifest.xml 清单里面声明。

Broadcast Intent Receiver

◆ 当你想要写一个程序来对外部的事件做些处理时，可以实用 **Broadcast Intent Receiver**。比如：当电话响铃时，有短信时。**Broadcast Intent Receiver** 它并不能够拿来显示 UI 画面，它必需利用 **NotificationManager** 来通知使用者它们感兴趣的事件发生了。

◆ **Broadcast Intent Receiver** 同样的可以在 **AndroidManifest.xml** 中声明，但你也可以用写 **Context.registerReceiver()** 程序的方式来注册你自己的 **Broadcast Intent Receiver**。你自己的程序并不会因为 **Broadcast Receiver** 被呼叫而被它执行起来，而是当 **BroadcastReceiver** 被触发时系统会依据需求来执行相应的程序。程序可以利用 **Context.sendBroadcast()** 来发出它们自己的 **IntentBroadcast** 给其它的程序。



Intent 与 Activity

◆ 而画面的切来切去则是由 **resolving Intent** 来实现的。当你想产生新的画面时，现行的 **Activity** 就使用 **startActivity(myIntent)**。然后系统就会根据所有已安装的程序所定义的 **Intent filter** 来看哪个程序是最适合 **myIntent**。当 **startActivity** 被呼叫时，**resolving Intents** 的处理过程是伴随而来的。**Resolving Intent** 提供我们两个好处：

- 让 **Activities** 可以很容易的利用 **Intent** 的方式去使用别的程序的功能。
- **Activities** 可以很容易的在任何情况下由新的 **Activity** 所取代。

添加新的 Activity

```

package zyf.Android.Study;
import .....
import android.content.Intent;
import android.net.Uri;
import android.view.View.OnClickListener;
public class AndroidStudy_TWO extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final EditText inName = (EditText) findViewById(R.id.name);
        final TextView result = (TextView) findViewById(R.id.result);
        Button button_Start_Browser = (Button) findViewById(R.id.submit_toNET);
        Button button_Login=(Button) findViewById(R.id.show_Login);
        Button button_showLoginName=
            (Button) findViewById(R.id.submit_toshowLoingName);
        button_Start_Browser.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                Uri myUri = Uri.parse("http://www.flashwing.net");
                Intent openBrowserIntent = new Intent(Intent.ACTION_VIEW,myUri);
                startActivity(openBrowserIntent);
            }
        });
        button_Login.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent openWelcomeActivityIntent=new Intent();
                openWelcomeActivityIntent.setClass(AndroidStudy_TWO.this,
                                                    Welcome.class);
                startActivity(openWelcomeActivityIntent);
            }
        });
        button_showLoginName.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                result.setText(inName.getText()+"欢迎您进入");
            }
        });
    }
}

```

android
developers



android

android
developers

Android开发教程&笔记



android

Android 应用开发 3

使用 Bundle 在 Activity 间传递数据

从源 Activity 中传递数据



//数据写入 Intent

```
Intent openWelcomeActivityIntent=new Intent();
Bundle myBundelForName=new Bundle();
myBundelForName.putString("Key_Name",inName.getText().toString());
myBundelForName.putString("Key_Age",inAge.getText().toString());
openWelcomeActivityIntent.putExtras(myBundelForName);
openWelcomeActivityIntent.setClass(AndroidBundel.this, Welcome.class);
startActivity(openWelcomeActivityIntent);
```

目标 Activity 中获取数据

//从 Intent 中获取数据


```
Bundle myBundelForGetName=this getIntent().getExtras();
String name=myBundelForGetName.getString("Key_Name");
myTextView_showName.setText("欢迎您进入: "+name);
```

使用 Bundle 在 Activity 间传递数据 2

从源请求 Activity 中通过一个 Intent 把一个服务请求传到目标 Activity 中

```
private Intent toNextIntent;//Intent 成员声明
toNextIntent=new Intent();//Intent 定义
toNextIntent.setClass(TwoActivityME3.this, SecondActivity3.class);
//设定开启的下一个Activity
startActivityForResult(toNextIntent, REQUEST_ASK);
//开启 Intent 时候，把请求码同时传递
```


在源请求 Activity 中等待 Intent 返回应答结果，通过重载 onActivityResult()方法



```
@Override
protected void onActivityResult(int requestCode,
                                int resultCode, Intent data) {
    // TODO Auto-generated method stub
    super.onActivityResult(requestCode, resultCode, data);
    if(requestCode==REQUEST_ASK){
        if(resultCode==RESULT_CANCELED){
            setTitle("Cancel****");
        }else if(resultCode==RESULT_OK){
            showBundle=data.getExtras();//从返回的Intent中获得Bundle
            Name=showBundle.getString("myName");//从bundle中获得相应数据
            text.setText("the name get from the second layout:\n"+Name);
        }
    }
}
```

- ☺ 第一个参数是你开启请求Intent时的对应请求码，可以自己定义。
- ☺ 第二个参数是目标Activity返回的验证结果码
- ☺ 第三个参数是目标Activity返回的Intent

目标 Activity 中发送请求结果代码，连同源 Activity 请求的数据一同绑定到 Bundle 中通过 Intent 传回源请求 Activity 中

```
backIntent=new Intent();
stringBundle=new Bundle();
stringBundle.putString("myName", Name);
backIntent.putExtras(stringBundle);
setResult(RESULT_OK, backIntent);//返回Activity结果码
finish();
```

Log 与 DDMS(查看 Log 等信息)

```
Log.v("TAG", "nextPage_Activity onStart()");//设置标签来跟踪程序
```


Activity 生命周期

Activity 状态

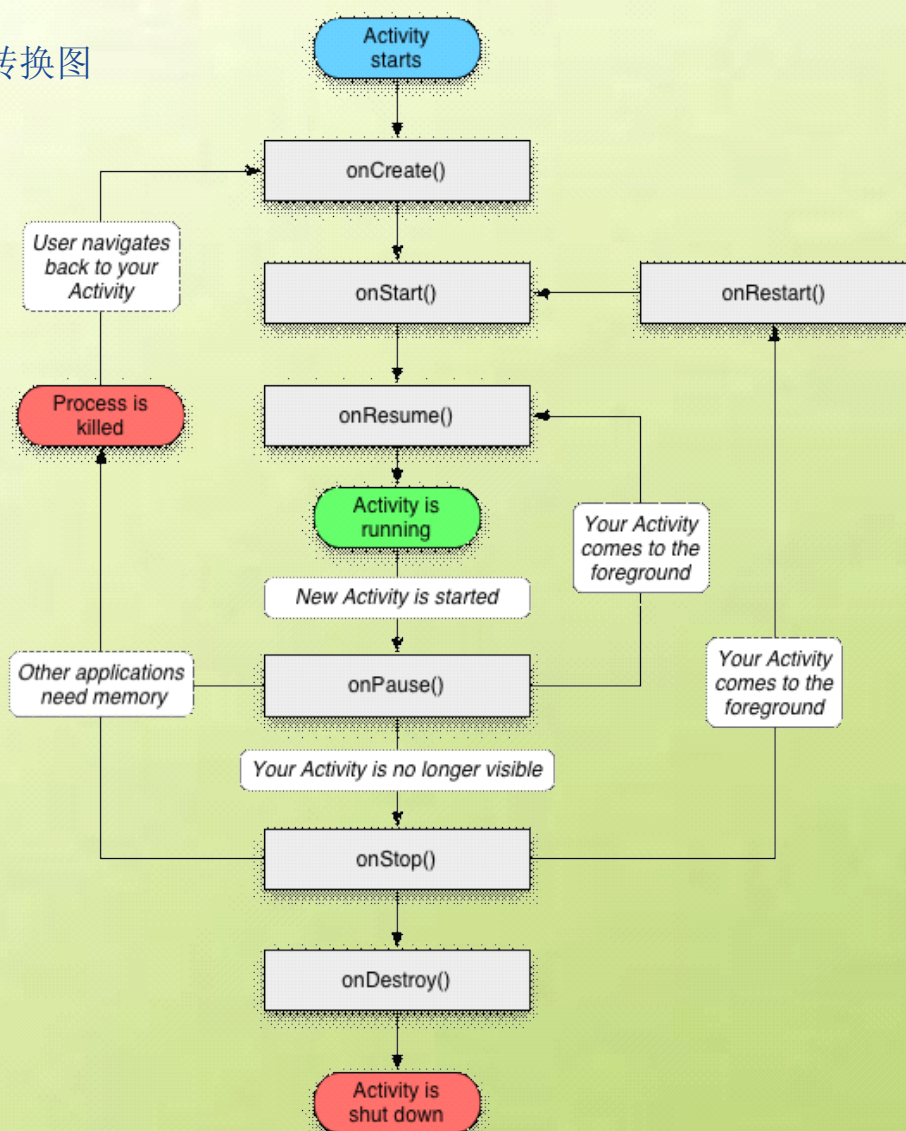
① 当一个 Activity 在屏幕的最上层时（对堆栈的最顶端），它就是属于 active 或者 running 的状态

② 如果一个 Activity 失去焦点（focus）但还看得到它的画面（比如：一个新的 Activity 画面并不是全屏或者它是一个半透明的情况），那失去焦点的 Activity 则处在 paused 的状态。像这个失去焦点的 Activity 它还是完全活着的，并没有消失。（活着的意思是指，Activity 自己本身所有的状态及数据都还是存在的，也跟窗口管理程序 window manager 保持联系着），像这种 paused 的 Activity，会在一种情况下消失，那就是当系统的内存不够用之时，系统会自动判断，八部重要的 Activity 移除。

③ 如果一个 Activity 被其它的 Activity 完全的遮盖住时，它仍然保有全部的状态及数据，但因为它已不再被使用者看见，所以它的画面是被隐藏起来的（画面不需要更新），当系统内存不足时，这种 stop 状态的 Activity 时最先被系统考虑拿下来释放内存的。

④ 当一个 Activity 处于 pause 或 stop 的状态时，系统可以要求 Activity 结束（finish）或直接移除（kill）它。当它需要再度呈现在使用者面前时，它必须要能完整的重新启动及回复先前的状态。

Activity 状态转换图



Android 应用开发 4 使用 Service

什么是服务（Service）

服务是运行在后台的一段代码。它可以运行在它自己的进程，也可以运行在其他应用程序的上下文（context）里面，这取决于自身的需要。其他的组件可以绑定到一个服务（Service）上面，通过远程过程调用（RPC）来调用这个方法。例如：媒体播放器的服务，当用户退出媒体选择用户界面，仍然希望音乐可以继续播放，这时就是由服务（Service）来保证当用户界面关闭时音乐继续播放的。



如何使用服务

- ✓ 第一种是通过调用 Context.startService() 启动，调用 Context.stopService() 结束，startService() 可以传递参数给 Service。
- ✓ 第二种方式是通过调用 Context.bindService() 启动，调用 Context.unbindService() 结束，还可以通过 ServiceConnection 访问 Service。二者可以混合使用，比如说我可以先 startService() 再 unbindService()。

Service 的生命周期

- ☹ startService() 后，即使调用 startService() 的进程结束了，Service 仍然存在，知道有进程调用 stopService()，或者 Service 自己自杀（stopSelf()）就没法了
- ☹ bindService() 后，Service 就和调用 bindService() 的进程同生共死，也就是说当调用 bindService() 的进程死了，那么它 bind 的 Service 也要跟着被结束，当然期间也可以调用 unbindService() 让 Service 结束
- ☹ 两种方式混合使用时，比如说你 startService() 了，我 bindService() 了，那么只有你 stopService() 了而且我也 unbindService() 了，这个 Service 才会被结束。

进程生命周期

- ♠ Android 系统将会尝试保留那些启动了或者时绑定了的的服务进程
- ♠ 如果该服务正在进程的 onCreate(), onStart() 或者 onDestroy() 这些方法中执行时，那么主进程将会成为一个前台进程，以确保此代码不会被停止
- ♠ 如果服务已经开始，那么它的主进程会就重要性而言低于所有可见的进程但高于不可见的进程，由于只有少数几个进程是用户可见的，所以只要不是内存特别低，该服务不会停止。
- ♠ 如果有多个客户端绑定了服务，只要客户端中的一个对于用户是可见的，即认为该服务可见

使用服务进行音乐播放

Manifest.xml
中的 Service
定义

```
<service android:name=".Music">  
    <intent-filter>  
        <action android:name="com.liangshan.wuyong.START_AUDIO_SERVICE" />  
        <category android:name="android.intent.category.default" />  
    </intent-filter>  
</service>
```

Service 子类
中的 Player

```
public void onStart(Intent intent, int startId) {  
    super.onStart(intent, startId);  
    player = MediaPlayer.create(this, R.raw.seven_days);  
    player.start();  
}  
  
public void onDestroy() {  
    super.onDestroy();  
    player.stop();  
}
```

Activity 中定
义的 Intent
开启相应的
Service

```
startService(new Intent("com.liangshan.wuyong.START_AUDIO_SERVICE"));  
  
stopService(new Intent("com.liangshan.wuyong.START_AUDIO_SERVICE"));
```



Android UI 布局

Activity

- ◆ Android 应用程序基本功能单元
- ◆ 本身没有任何屏幕存在

View 和 Viewgroup

- ◆ 表示在 Android 平台上的基本用户界面单元

Views

- ◆ android.view.View
 - 为指定的屏幕矩形区域存储布局和内容
 - 处理尺寸和布局，绘制，焦点改变，翻页，按键、手势
 - widget 基类

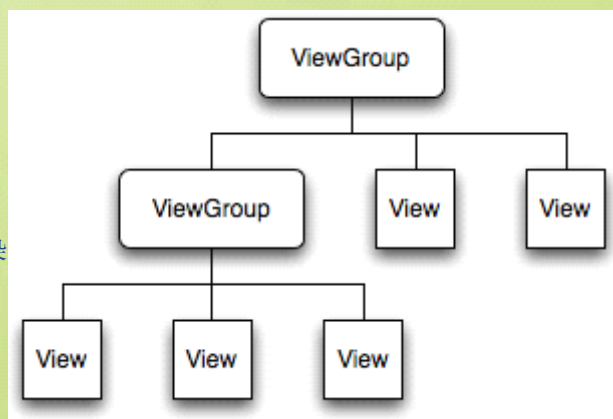
文本 TextView	输入框 EditText
输入法 InputMethod	活动方法 MovementMethod
按钮 Button	单选按钮 RadioButton
复选框 Checkbox	滚动视图 ScrollView

Viewgroups

- ◆ android.view.ViewGroup
 - 包含并管理下级系列的 Views 和其他 ViewGroup
 - 布局的基类

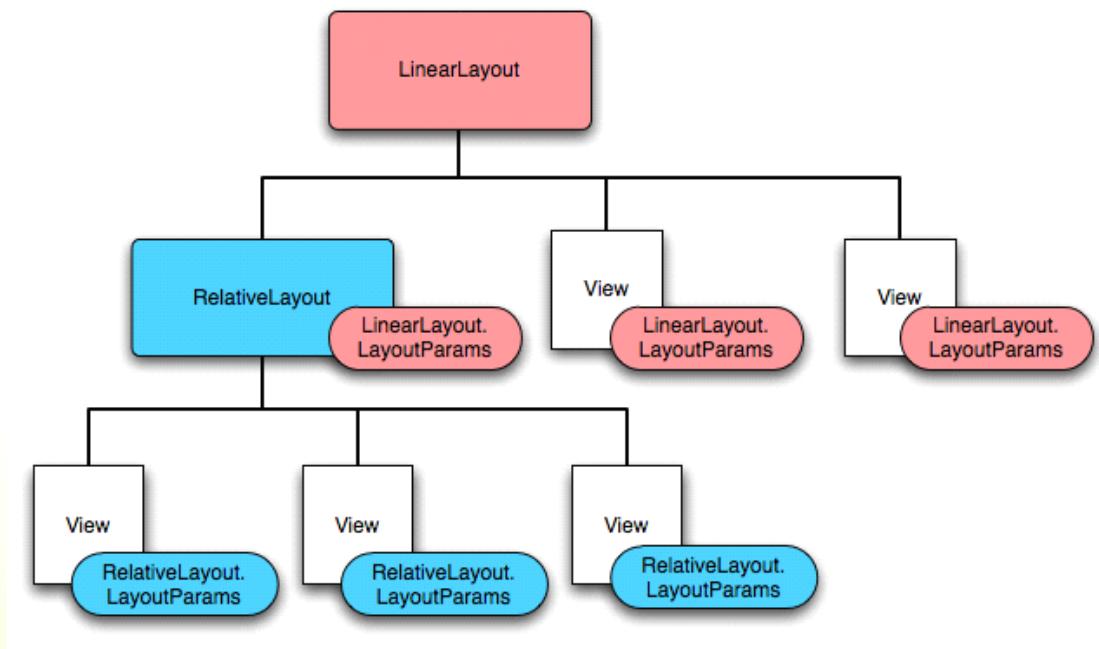
UI 树状结构

- ◆ Android 中的 Activity
 - 定义使用一个 view 和 iewgroup 的树状节点
- ◆ setContentView() 方法
 - 被 Activity 调用来把树状节点连接到屏幕渲染



LayoutParams (布局参数)

- ◆ 每一个 viewgroup 类使用一个继承于 ViewGroup.LayoutParams 的嵌套类
 - 包含定义了子节点 View 的尺寸和位置的属性类型



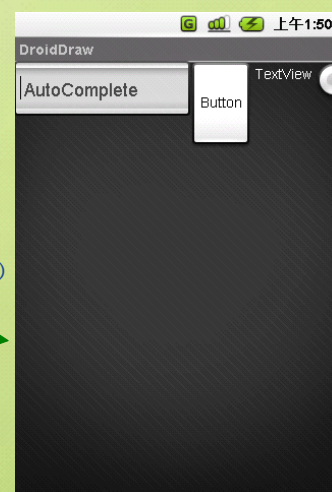
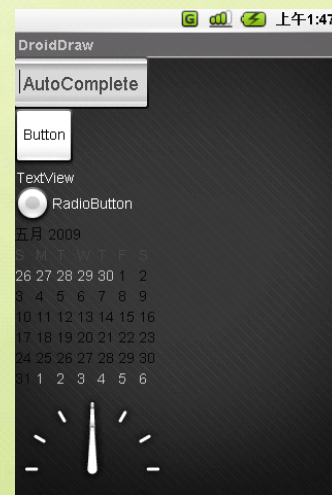
普通布局对象

FrameLayout

- ◆ 最简单的布局对象
- ◆ 在屏幕上故意保留的空白空间，你可以之后填充一个单独的对象
 - 例如：一个你要更换的图片
- ◆ 所有子元素都钉到屏幕的左上角
- ◆ 不能为子元素指定位置

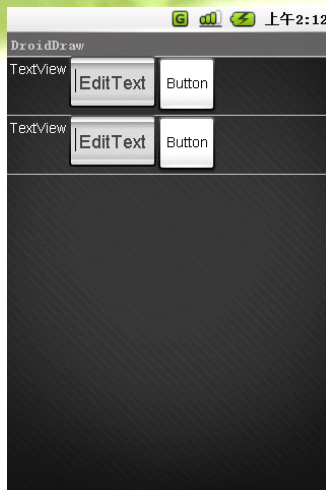
LinearLayout

- ◆ 在一个方向上(垂直或水平)对齐所有子元素
 - 所有子元素一个跟一个地堆放
 - ✓ 一个垂直列表每行将只有一个子元素(无论它们有多宽)
 - ✓ 一个水平列表只是一列的高度(最高子元素的高度来填充)



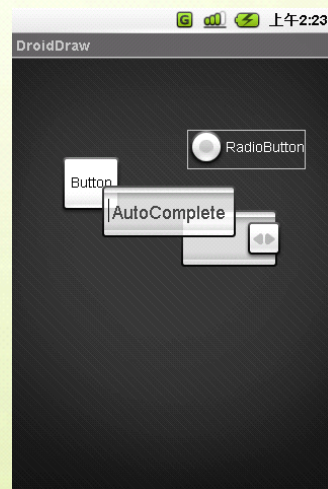
TableLayout

- ◆ 把子元素放入到行与列中
- ◆ 不显示行、列或是单元格边界线
- ◆ 单元格不能横跨行，如 HTML 中一样



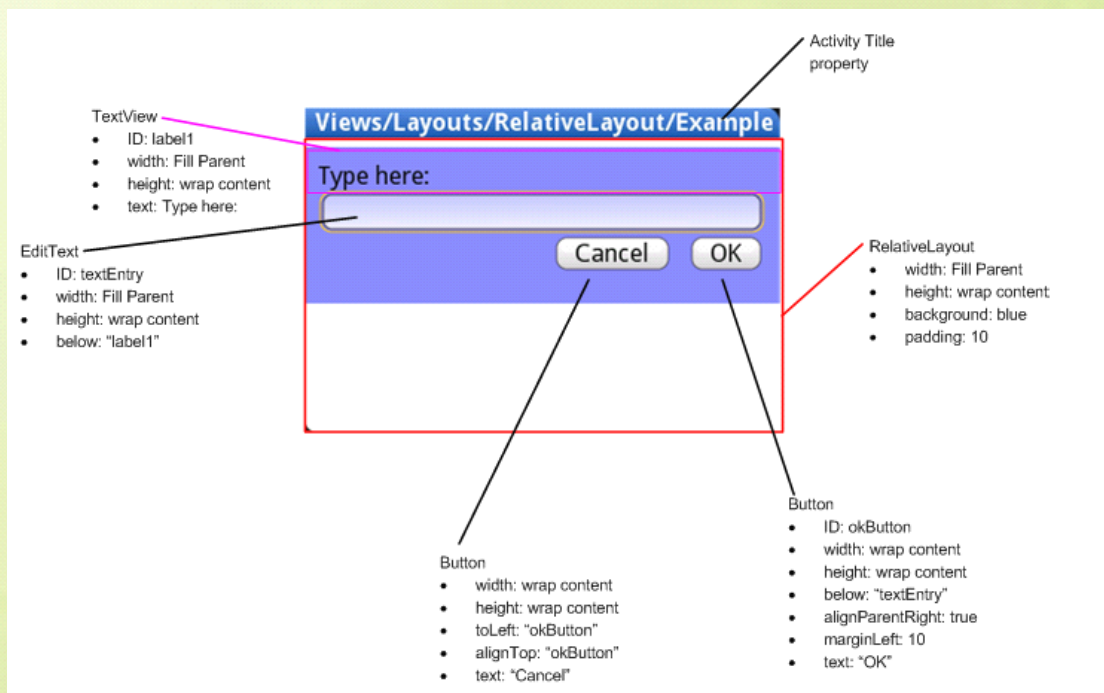
AbsoluteLayout

- ◆ 使子元素能够指明确切的 X / Y 坐标显示在屏幕上
 - (0,0)是左上角
 - 当你下移或右移时，坐标值增加
- ◆ 允许元素重叠(但是不推荐)
- ◆ 注意：
 - 一般建议不使用 AbsoluteLayout 除非你有很好的理由来使用它
 - 因为它相当严格并且在不同的设备显示中不能很好地工作



RelativeLayout

- ◆ 让子元素指定它们相对于其他元素的位置(通过 ID 来指定)或相对于父布局对象



AndroidManifest.xml 中修改程序布局的 Theme 主题



```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="zyf.GridViewTest"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon"
        android:theme="@android:style/Theme.Light"
        android:label="@string/app_name">
        <activity android:name=".GridViewTest"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
                    android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="2" />
</manifest>
```


android
developers



android

android
developers

Android开发教程&笔记



android

Android 基础 UI 编程 1

更改与显示文字标签

TextView 标签的使用

- ① 导入 TextView 包

```
import android.widget.TextView;
```

- ② 在 MainActivity.java 中声明一个 TextView

```
private TextView mTextView01;
```

- ③ 在 main.xml 中定义一个 TextView

```
<TextView android:text="TextView01"
          android:id="@+id/TextView01"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_x="61px"
          android:layout_y="69px">

</TextView>
```

- ④ 利用 findViewById() 方法获取 main.xml 中的 TextView

```
mTextView01 = (TextView) findViewById(R.id.TextView01);
```

- ⑤ 设置 TextView 标签内容

```
String str_2 = "欢迎来到Android 的TextView 世界...";
mTextView01.setText(str_2);
```

- ⑥ 设置文本超级链接

```
<TextView
    android:id="@+id/TextView02"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:autoLink="all"
    android:text="请访问Android 开发者:
                http://developer.android.com/index.html">

</TextView>
```

android.graphics.Color 实践——Color 颜色变幻

android.graphics.Color 包含颜色值

Color.BLACK	黑色
Color.BLUE	蓝色
Color.CYAN	青绿色
Color.DKGRAY	灰黑色
Color.GRAY	灰色
Color.GREEN	绿色
Color.LTGRAY	浅灰色
Color.MAGENTA	红紫色
Color.RED	红色
Color.TRANSPARENT	透明
Color.WHITE	白色
Color.YELLOW	黄色

编程实现颜色变幻

① 新建工程

② 修改 MainActivity.java 文件，添加 12 个 TextView 对象变量，一个 LinearLayout 对象变量、一个 WC 整数变量、一个 LinearLayout.LayoutParams 变量。

```
package zyf.ManyColorME;

/*导入要使用的包*/
import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.widget.LinearLayout;
import android.widget.TextView;

public class ManyColorME extends Activity {
    /** Called when the activity is first created. */
    /* 定义使用的对象 */
    private LinearLayout myLayout;
    private LinearLayout.LayoutParams layoutP;
    private int WC = LinearLayout.LayoutParams.WRAP_CONTENT;
    private TextView black_TV, blue_TV, cyan_TV, dkgray_TV,
        gray_TV, green_TV, ltgray_TV, magenta_TV, red_TV,
        transparent_TV, white_TV, yellow_TV;

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



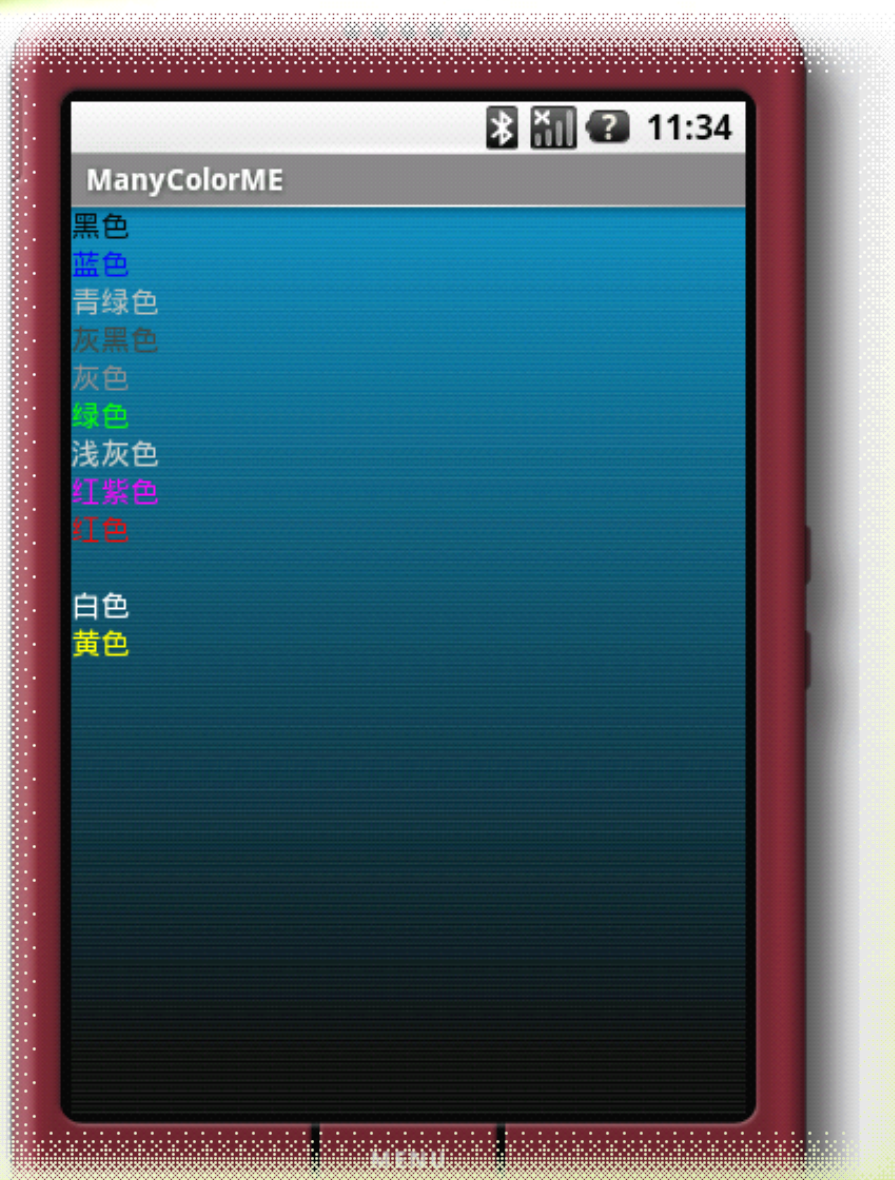

```
super.onCreate(savedInstanceState);
/* 实例化一个LinearLayout布局对象 */
myLayout = new LinearLayout(this);
/* 设置LinearLayout的布局为垂直布局 */
myLayout.setOrientation(LinearLayout.VERTICAL);
/* 设置LinearLayout布局背景图片 */
myLayout.setBackgroundResource(R.drawable.back);
/* 加载主屏布局 */
setContentView(myLayout);
/* 实例化一个LinearLayout布局参数，用来添加View */
layoutP = new LinearLayout.LayoutParams(WC, WC);
/* 构造实例化TextView对象 */
constructTextView();
/* 把TextView添加到LinearLayout布局中 */
addTextView();
/* 设置TextView文本颜色 */
setTextViewColor();
/* 设置TextView文本内容 */
setTextViewText();
}
/* 设置TextView文本内容 */
public void setTextViewText() {
    black_TV.setText("黑色");
    blue_TV.setText("蓝色");
    cyan_TV.setText("青绿色");
    dkgray_TV.setText("灰黑色");
    gray_TV.setText("灰色");
    green_TV.setText("绿色");
    ltgray_TV.setText("浅灰色");
    magenta_TV.setText("红紫色");
    red_TV.setText("红色");
    transparent_TV.setText("透明");
    white_TV.setText("白色");
    yellow_TV.setText("黄色");
}
/* 设置TextView文本颜色 */
public void setTextViewColor() {
    black_TV.setTextColor(Color.BLACK);
    blue_TV.setTextColor(Color.BLUE);
    dkgray_TV.setTextColor(Color.DKGRAY);
    gray_TV.setTextColor(Color.GRAY);
    green_TV.setTextColor(Color.GREEN);
    ltgray_TV.setTextColor(Color.LTGRAY);
    magenta_TV.setTextColor(Color.MAGENTA);
```

```
        red_TV.setTextColor(Color.RED);
        transparent_TV.setTextColor(Color.TRANSPARENT);
        white_TV.setTextColor(Color.WHITE);
        yellow_TV.setTextColor(Color.YELLOW);
    }

    /* 构造实例化TextView对象 */
    public void constructTextView() {
        black_TV = new TextView(this);
        blue_TV = new TextView(this);
        cyan_TV = new TextView(this);
        dkgray_TV = new TextView(this);
        gray_TV = new TextView(this);
        green_TV = new TextView(this);
        ltgray_TV = new TextView(this);
        magenta_TV = new TextView(this);
        red_TV = new TextView(this);
        transparent_TV = new TextView(this);
        white_TV = new TextView(this);
        yellow_TV = new TextView(this);
    }

    /* 把TextView添加到LinearLayout布局中 */
    public void addTextView() {
        myLayout.addView(black_TV, layoutP);
        myLayout.addView(blue_TV, layoutP);
        myLayout.addView(cyan_TV, layoutP);
        myLayout.addView(dkgray_TV, layoutP);
        myLayout.addView(gray_TV, layoutP);
        myLayout.addView(green_TV, layoutP);
        myLayout.addView(ltgray_TV, layoutP);
        myLayout.addView(magenta_TV, layoutP);
        myLayout.addView(red_TV, layoutP);
        myLayout.addView(transparent_TV, layoutP);
        myLayout.addView(white_TV, layoutP);
        myLayout.addView(yellow_TV, layoutP);
    }
}
```


③ 结果



android.graphics.Typeface 实践

字体风格 Typeface 种类

int Style 类型

BOLD	粗体
BOLD_ITALIC	粗斜体
ITALIC	斜体
NORMAL	普通字体

Typeface 类型

DEFAULT	默认字体
DEFAULT_BOLD	默认粗体
MONOSPACE	单间隔字体
SANS_SERIF	无衬线字体
SERIF	衬线字体

Typeface.create(Typeface family,int style)

创建一个混合型新的字体：有 4*5 中搭配

Typeface.setTypeface (Typeface tf, int style)

设置一个混合型字体：有 4*5 中搭配

Typeface.setTypeface(Typeface tf)

设置一个只有 Typeface 风格的字体：有五种形式

编程实现以上静态域字体

① 创建新工程

② 修改 mianActivity.java, 实现多种字体 TextView 显示



```
package zyf.TypefaceStudy;
/*导入要使用的包*/
import android.app.Activity;
import android.graphics.Color;
import android.graphics.Typeface;
import android.os.Bundle;
import android.view.ViewGroup;
import android.widget.LinearLayout;
import android.widget.TextView;
public class TypefaceStudy extends Activity {
    /** Called when the activity is first created. */
    /*
     * android.graphics.Typeface java.lang.Object
     * Typeface类指定一个字体的字体和固有风格.
     * 该类用于绘制, 与可选绘制设置一起使用,
     * 如textSize, textSkewX, textScaleX 当绘制(测量)时来指定如何显示文本.
     */
    /* 定义实例化一个 布局大小, 用来添加TextView */
    final int WRAP_CONTENT = ViewGroup.LayoutParams.WRAP_CONTENT;
    /* 定义TextView对象 */
    private TextView bold_TV, bold_italic_TV, default_TV,
        default_bold_TV, italic_TV, monospace_TV,
        normal_TV, sans_serif_TV, serif_TV;
    /* 定义LinearLayout布局对象 */
    private LinearLayout linearLayout;
    /* 定义LinearLayout布局参数对象 */
    private LinearLayout.LayoutParams linearLayouttParams;
    @Override
    public void onCreate(Bundle icle) {
        super.onCreate(icle);
        /* 定义实例化一个LinearLayout对象 */
        linearLayout = new LinearLayout(this);
        /* 设置LinearLayout布局为垂直布局 */
        linearLayout.setOrientation(LinearLayout.VERTICAL);
        /*设置布局背景图*/
        linearLayout.setBackgroundResource(R.drawable.back);
        /* 加载LinearLayout为主屏布局, 显示 */
        setContentView(linearLayout);
    }
}
```

```

/* 定义实例化一个LinearLayout布局参数 */
linearLayouttParams =
    new LinearLayout.LayoutParams(WRAP_CONTENT, WRAP_CONTENT);
constructTextView();
setTextSizeOf();
setTextViewText();
setStyleOfFont();
setFontColor();
toAddTextViewToLayout();
}

public void constructTextView() {
    /* 实例化TextView对象 */
    bold_TV = new TextView(this);
    bold_italic_TV = new TextView(this);
    default_TV = new TextView(this);
    default_bold_TV = new TextView(this);
    italic_TV = new TextView(this);
    monospace_TV = new TextView(this);
    normal_TV = new TextView(this);
    sans_serif_TV = new TextView(this);
    serif_TV = new TextView(this);
}

public void setTextSizeOf() {
    // 设置绘制的文本大小, 该值必须大于0
    bold_TV.setTextSize(24.0f);
    bold_italic_TV.setTextSize(24.0f);
    default_TV.setTextSize(24.0f);
    default_bold_TV.setTextSize(24.0f);
    italic_TV.setTextSize(24.0f);
    monospace_TV.setTextSize(24.0f);
    normal_TV.setTextSize(24.0f);
    sans_serif_TV.setTextSize(24.0f);
    serif_TV.setTextSize(24.0f);
}

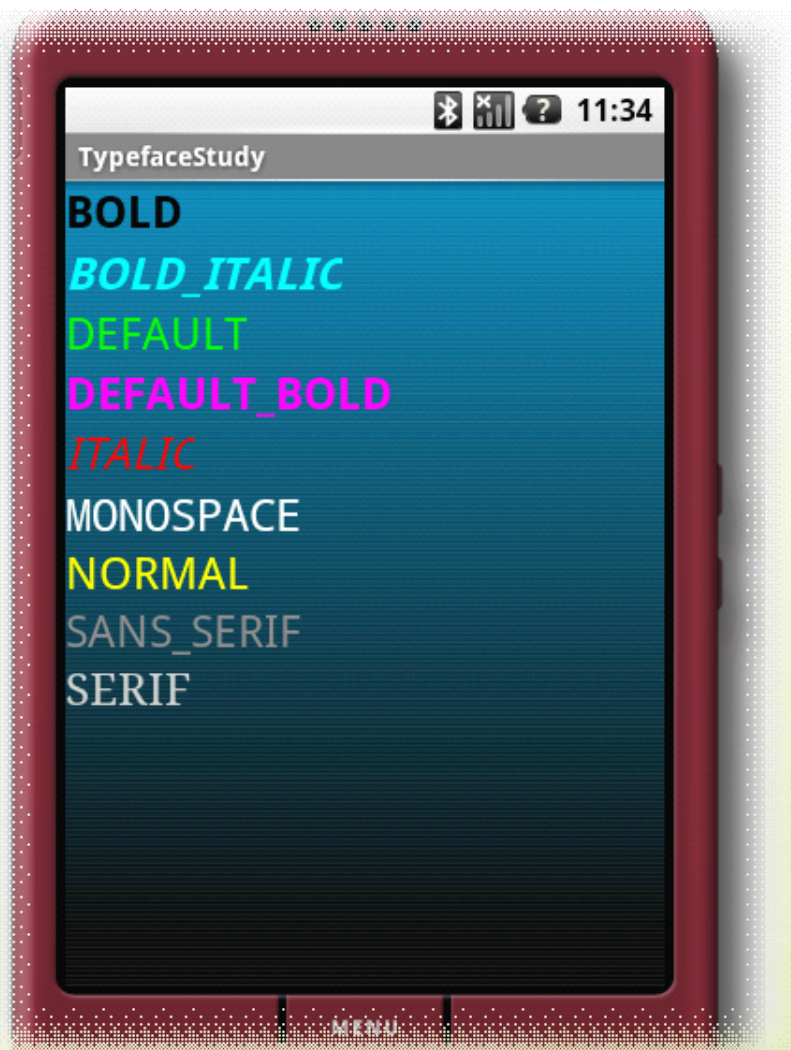
public void setTextViewText() {
    /* 设置文本 */
    bold_TV.setText("BOLD");
    bold_italic_TV.setText("BOLD_ITALIC");
    default_TV.setText("DEFAULT");
    default_bold_TV.setText("DEFAULT_BOLD");
    italic_TV.setText("ITALIC");
    monospace_TV.setText("MONOSPACE");
    normal_TV.setText("NORMAL");
    sans_serif_TV.setText("SANS_SERIF");
}

```




```
        serif_TV.setText("SERIF");
    }
    public void setStyleOfFont() {
        /* 设置字体风格 */
        bold_TV.setTypeface(null, Typeface.BOLD);
        bold_italic_TV.setTypeface(null, Typeface.BOLD_ITALIC);
        default_TV.setTypeface(Typeface.DEFAULT);
        default_bold_TV.setTypeface(Typeface.DEFAULT_BOLD);
        italic_TV.setTypeface(null, Typeface.ITALIC);
        monospace_TV.setTypeface(Typeface.MONOSPACE);
        normal_TV.setTypeface(null, Typeface.NORMAL);
        sans_serif_TV.setTypeface(Typeface.SANS_SERIF);
        serif_TV.setTypeface(Typeface.SERIF);
    }
    public void setFontColor() {
        /* 设置文本颜色 */
        bold_TV.setTextColor(Color.BLACK);
        bold_italic_TV.setTextColor(Color.CYAN);
        default_TV.setTextColor(Color.GREEN);
        default_bold_TV.setTextColor(Color.MAGENTA);
        italic_TV.setTextColor(Color.RED);
        monospace_TV.setTextColor(Color.WHITE);
        normal_TV.setTextColor(Color.YELLOW);
        sans_serif_TV.setTextColor(Color.GRAY);
        serif_TV.setTextColor(Color.LTGRAY);
    }
    public void toAddTextViewToLayout() {
        /* 把TextView加入LinearLayout布局中 */
        linearLayout.addView(bold_TV, linearLayouttParams);
        linearLayout.addView(bold_italic_TV, linearLayouttParams);
        linearLayout.addView(default_TV, linearLayouttParams);
        linearLayout.addView(default_bold_TV, linearLayouttParams);
        linearLayout.addView(italic_TV, linearLayouttParams);
        linearLayout.addView(monospace_TV, linearLayouttParams);
        linearLayout.addView(normal_TV, linearLayouttParams);
        linearLayout.addView(sans_serif_TV, linearLayouttParams);
        linearLayout.addView(serif_TV, linearLayouttParams);
    }
}
```

③ 结果



更改手机窗口画面底色

drawable 定义颜色常数的方法

① 编写 main 布局

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/widget0"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
    android:id="@+id/name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="账号"
    android:layout_x="61px"
    android:layout_y="69px"
>
</TextView>
<TextView
    android:id="@+id/password"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="密码"
    android:layout_x="61px"
    android:layout_y="158px"
>
</TextView>
<EditText
    android:id="@+id/name_in"
    android:layout_width="120dip"
    android:layout_height="wrap_content"
    android:textSize="18sp"
    android:layout_x="114px"
    android:layout_y="57px"
>
</EditText>
<EditText
    android:id="@+id/pwd_in"
    android:layout_width="120dip"
    android:layout_height="wrap_content"
```

```

        android:textSize="18sp"
        android:password="true"
        android:layout_x="112px"
        android:layout_y="142px"
    >
</EditText>
</AbsoluteLayout>

```

② 在 `values` 文件夹中定义一个 `drawable.xml` 文件 用来存放颜色值

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="white">#FFFFFF</color>
    <color name="darkgray">#938192</color>
    <color name="lightgreen">#7cd12e</color>
</resources>

```

③ 修改 `main.xml` 中的屏幕背景颜色和 `TextView` 的字体颜色

```

<AbsoluteLayout
    android:id="@+id/widget0"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@color/white"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
    android:id="@+id/name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="账号"
    android:textColor="@color/darkgray"
    android:layout_x="61px"
    android:layout_y="69px"
>
</TextView>
<TextView
    android:id="@+id/password"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="密码"
    android:textColor="@color/darkgray"
    android:layout_x="61px"
    android:layout_y="158px"
>

```


Android 编程基础

④ 在 MainActivity.java 代码中修改 TextView 背景颜色

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
    TextView text=(TextView)findViewById(R.id.name);  
    //由ID获得资源  
  
    Resources myColor=getBaseContext().getResources();  
    //getBaseContext() 获得基础Context  
    //getResources() 获得资源  
  
    Drawable color_M=myColor.getDrawable(R.color.lightgreen);  
    //由资源 myColor来获得Drawable R.color.lightgreen是颜色值的ID引用  
  
    text.setBackgroundDrawable(color_M);  
    //设置背景  
}
```

⑤ 结果:



代码中更改 **TextView** 文字颜色

① 创建工程

② 编写布局 main.xml



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
<TextView
    android:text="TextView01"
    android:id="@+id/TextView01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
</TextView>
<TextView
    android:text="这里使用Graphics颜色静态常量"
    android:id="@+id/TextView02"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
</TextView>
</LinearLayout>
```

③ 新加 drawable.xml，其中添加一个 white 颜色值

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="white">#ffffffff</color>
</resources>
```

④ 在代码中由 ID 获取 TextView

```
TextView text_A=(TextView)findViewById(R.id.TextView01);
TextView text_B=(TextView)findViewById(R.id.TextView02);
```

⑤ 获取 Resources 对象

```
Resources myColor_R=getBaseContext().getResources();
//getBaseContext() 获得基础 Context
//getResources() 从 Context 获取资源实例对象
```


⑥ 获取 Drawable 对象

```
Drawable myColor_D=myColor_R.getDrawable(R.color.white);
```

⑦ 设置文本背景颜色

```
text_A.setBackgroundDrawable(myColor_D);
```

⑧ 利用 android.graphics.Color 的颜色静态变量来改变文本颜色

```
text_A.setTextColor(android.graphics.Color.GREEN);
```

⑨ 利用 Color 的静态常量来设置文本颜色

```
text_B.setTextColor(Color.RED);
```




置换 TextView 文字

CharSequence 数据类型与 Resource ID 应用

① 创建新工程

② 修改 main.xml 布局，新增一个 TextView



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:text="@string/str_2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/myTextView2"
    />
</LinearLayout>
```

③ 在 string.xml 中添加字符串 str_2:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, Ex4_UI!</string>
    <string name="app_name">Ex4_UI</string>
    <string name="str_2">我是Resource里的“字符串”</string>
</resources>
```

④ 在 MainActivity.java 中 findViewById() 获取 TextView

```
TextView myText=(TextView) findViewById(R.id.myTextView2);
```

⑤ getString(R.string.str_2) 得到原字符串

```
CharSequence string2=getString(R.string.str_2);
```

⑥ 定义新字符串 str_3

```
String str_3="我是程序里调用 Resource 的";
```

⑦ 两个字符串连接并重新设置 TextView 内容


```
myText.setText(str_3+string2);
```


取得手机屏幕大小

DisplayMetrics 类取得画面宽高

① 创建新工程

② 修改 main.xml 布局，添加一个 TextView



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:text="TextView01"
    android:id="@+id/TextView01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"></TextView>
</LinearLayout>
```

③ 在代码中定义一个 DisplayMetrics 类对象

```
DisplayMetrics displaysMetrics=new DisplayMetrics();
//DisplayMetrics 一个描述普通显示信息的结构，例如显示大小、密度、字体尺寸
```

④ 获取手机窗口的 Display 来初始化 DisplayMetrics 对象

```
getWindowManager().getDefaultDisplay().getMetrics(displaysMetrics);
//getManager() 获取显示定制窗口的管理器。
//获取默认显示Display对象
//通过 Display 对象的数据来初始化一个 DisplayMetrics 对象
```

⑤ 得到屏幕宽高

```
String showSize="手机屏幕分辨率:  \n"+
    displaysMetrics.widthPixels+"*"+displaysMetrics.heightPixels;
```

⑥ 在 MainActivity.java 中 findViewById() 获取 TextView

```
TextView myShow=(TextView) findViewById(R.id.TextView01);
```

⑦ 显示屏幕分辨率信息

```
myShow.setText(showSize);
```

android
developers



android

android
developers

Android开发教程&笔记




android

Android 基础 UI 编程 2

标题、状态栏的隐藏

标题栏隐藏

在 Activity.setCurrentView();之前调用此方法



```
private void HideTitle() {  
    // TODO Auto-generated method stub  
    requestWindowFeature(Window.FEATURE_NO_TITLE);  
}
```

状态栏隐藏(全屏)

在 Activity.setCurrentView();之前调用此方法

```
private void HideStatusBar() {  
    // TODO Auto-generated method stub  
    //隐藏标题  
    requestWindowFeature(Window.FEATURE_NO_TITLE);  
    //定义全屏参数  
    int flag=WindowManager.LayoutParams.FLAG_FULLSCREEN;  
    //获得窗口对象  
    Window myWindow=this.getWindow();  
    //设置Flag标识  
    myWindow.setFlags(flag, flag);  
}
```


样式化的定型对象

Style 样式的定义

① 新建工程

② 定义一个 style.xml 存放样式



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="myStyle_Text1">
        <item name="android:textSize">25sp</item>
        <item name="android:textColor">#80FF00</item>
    </style>
    <style name="myStyle_Text2">
        <item name="android:textSize">18sp</item>
        <item name="android:textColor">#0C688E</item>
        <item name="android:fromAlpha">0.0</item>
        <item name="android:toAlpha">0.0</item>
    </style>
</resources>
```

③ 在 string.xml 中添加字符串

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="string_A">应用myStyle_Text1</string>
    <string name="string_B">应用myStyle_Text2</string>
</resources>
```

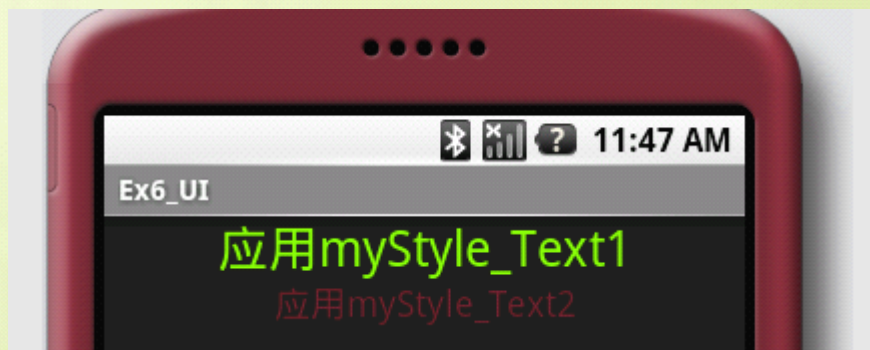
④ 修改布局 main.xml，添加两个 TextView

```
<TextView
    android:id="@+id/TextView01"
    android:layout_height="wrap_content"
    android:layout_width="fill_parent"
    android:gravity="center_vertical|center_horizontal"
    android:text="@string/string_A"></TextView>
<TextView
    android:id="@+id/TextView02"
    android:layout_height="wrap_content"
    android:layout_width="fill_parent"
    android:gravity="center_vertical|center_horizontal"
    android:text="@string/string_B"></TextView>
```

⑤ 加入 Style

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:id="@+id/TextView01"
        style="@style/myStyle_Text1"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:gravity="center_vertical|center_horizontal"
        android:text="@string/string_A"></TextView>
    <TextView
        android:id="@+id/TextView02"
        style="@style/myStyle_Text2"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:gravity="center_vertical|center_horizontal"
        android:text="@string/string_B"></TextView>
</LinearLayout>
```

⑥ 结果:



简易的按钮事件

Button 事件处理

① 创建新工程

② 修改 main.xml 布局，添加一个 TextView 和一个 Button



```
<?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/show_TextView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
    />
    <Button
        android:id="@+id/Click_Button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="点击"
    />
</LinearLayout>
```

③ 在 MainActivity.java 中 findViewById() 获取 TextView 和 Button 资源

```
show= (TextView) findViewById(R.id.show_TextView);
press=(Button) findViewById(R.id.Click_Button);
```

④ 给 Button 添加事件监听器 Button.OnClickListener()

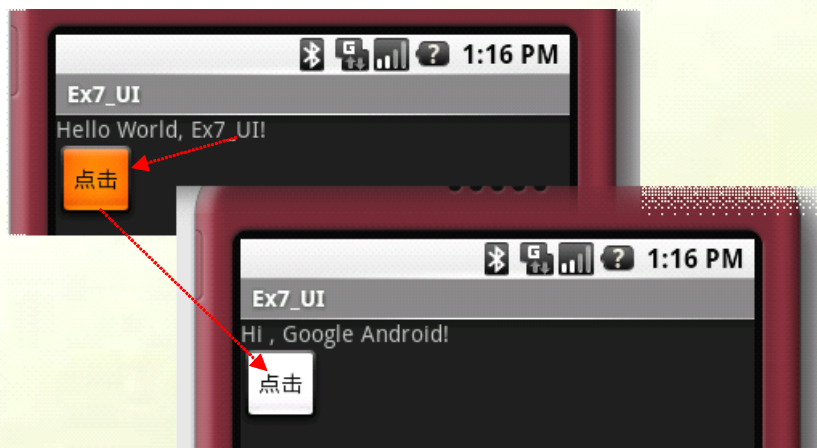
```
press.setOnClickListener(new Button.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
    }
});
```

Android 编程基础

⑤ 处理事件

```
press.setOnClickListener(new Button.OnClickListener(){  
  
    @Override  
    public void onClick(View v) {  
        // TODO Auto-generated method stub  
        show.setText("Hi , Google Android!");  
    }  
});
```

⑥ 结果:




手机页面的转换

setContentView 的应用

① 新建工程

② string 添加两个提示字符串



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="layout1">this is Layout 1</string>
    <string name="layout2">This is Layout 2</string>
    <string name="app_name">Ex8_UI</string>
</resources>
```

③ 新建 color.xml 保存两个颜色值

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="black">#000000</color>
    <color name="white">#FFFFFF</color>
</resources>
```

④ 修改 main.xml 布局, 添加一个 TextView 和一个 Button

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@color/black"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
    android:id="@+id/text1"
    android:textSize="24sp"
    android:layout_width="186px"
    android:layout_height="29px"
    android:layout_x="70px"
    android:layout_y="32px"
    android:text="@string/layout1"
></TextView>
<Button
    android:id="@+id/button1"
    android:layout_width="118px"
    android:layout_height="wrap_content"
    android:layout_x="100px"
```

```

        android:layout_y="82px"
        android:text="Go to Layout2"
    <</Button>
</AbsoluteLayout>

```

⑤ 新建 mylayout.xml 布局文件，并添加两个 View: TextView 和 Button

```

<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@color/white"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
    <TextView
        android:id="@+id/text2"
        android:textSize="24sp"
        android:layout_width="186px"
        android:layout_height="29px"
        android:layout_x="70px"
        android:layout_y="32px"
        android:textColor="@color/black"
        android:text="@string/layout2"
    >
    </TextView>
    <Button
        android:id="@+id/button2"
        android:layout_width="118px"
        android:layout_height="wrap_content"
        android:layout_x="100px"
        android:layout_y="82px"
        android:text="Go to Layout1"
    ></Button>
</AbsoluteLayout>

```

⑥ 编写 MainActivity.java

```

package zyf.Ex8_UI;
import android.app.Activity; /* import 相关class */
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
public class Ex8_UI extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {

```




```

super.onCreate(savedInstanceState);
/* 载入main.xml Layout */
setContentView(R.layout.main); // 默认启动布局
/* 以findViewById() 取得Button 对象, 并添加onClickListener */
Button b1 = (Button) findViewById(R.id.button1);
b1.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {
        jumpToLayout2(); // 调用跳转方法jumpToLayout2()
    }
});

}

/* method jumpToLayout2: 将layout 由main.xml 切换到mylayout.xml */
public void jumpToLayout2() {
    /* 将layout 改成mylayout.xml */
    setContentView(R.layout.mylayout);
    /* 以findViewById() 取得Button 对象, 并添加onClickListener */
    Button b2 = (Button) findViewById(R.id.button2);
    b2.setOnClickListener(new Button.OnClickListener() {
        public void onClick(View v) {
            jumpToLayout1(); // 调用跳转方法jumpToLayout1()
        }
    });
}

/* method jumpToLayout1: 将layout 由mylayout.xml 切换到main.xml */
public void jumpToLayout1() {
    /* 将layout 改成main.xml */
    setContentView(R.layout.main);
    /* 以findViewById() 取得Button 对象, 并添加onClickListener */
    Button b1 = (Button) findViewById(R.id.button1);
    b1.setOnClickListener(new Button.OnClickListener() {
        public void onClick(View v) {
            jumpToLayout2(); // 调用跳转方法jumpToLayout2()
        }
    });
}
}

```

⑦ 结果




调用另一个 Activity

Intent 对象的使用

① 新建工程

② 在 string.xml 中添加两个字符串



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, Ex9_UI!</string>
    <string name="app_name">Ex9_UI</string>
    <string name="act1">This is Activity 1!</string>
    <string name="act2">This is Activity 2!</string>
</resources>
```

③ 新建 color.xml 存放颜色值

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="black">#000000</color>
    <color name="white">#FFFFFF</color>
</resources>
```

④ 修改 main.xml 布局，添加一个 TextView 和一个 Button

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@color/black"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
    <TextView
        android:id="@+id/text1"
        android:textSize="24sp"
        android:layout_width="186px"
        android:layout_height="29px"
        android:layout_x="70px"
        android:layout_y="32px"
        android:text="@string/act1"
    ></TextView>
    <Button
        android:id="@+id/button1"
        android:layout_width="118px"
        android:layout_height="wrap_content"
```



```

    android:layout_x="100px"
    android:layout_y="82px"
    android:text="Go to Activity2"
<</Button>
</AbsoluteLayout>

```

⑤ 新建一个 secondlayout.xml 布局，并添加一个 TextView 和一个 Button

```

<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@color/white"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
    <TextView
        android:id="@+id/text2"
        android:textSize="24sp"
        android:layout_width="186px"
        android:layout_height="29px"
        android:layout_x="70px"
        android:layout_y="32px"
        android:textColor="@color/black"
        android:text="@string/act2"
    ></TextView>
    <Button
        android:id="@+id/button2"
        android:layout_width="118px"
        android:layout_height="wrap_content"
        android:layout_x="100px"
        android:layout_y="82px"
        android:text="Go to Activity1"
    ></Button>
</AbsoluteLayout>

```

⑥ 新建 SecondActivity.java 文件，添加内容

```

package zyf.Ex9_UI;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class SecondActivity extends Activity {

```



```

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    /* 载入mylayout.xml Layout */
    setContentView(R.layout.mylayout);
    /* 以findViewById() 取得Button 对象, 并添加onClickListener */
    Button b2 = (Button) findViewById(R.id.button2);
    b2.setOnClickListener(new Button.OnClickListener() {
        public void onClick(View v) {
            /* new 一个Intent 对象, 并指定要启动的class */
            Intent intent = new Intent();
            intent.setClass(SecondActivity.this, Ex9_UI.class);
            /* 调用一个新的Activity */
            startActivity(intent);
            /* 关闭原本的Activity */
            SecondActivity.this.finish();
        }
    });
}
}

```

⑦ 修改 MainActivity.java,添加代码

```

package zyf.Ex9_UI;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class Ex9_UI extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /* 载入main.xml Layout */
        setContentView(R.layout.main);
        /* 以findViewById() 取得Button 对象, 并添加onClickListener */
        Button b1 = (Button) findViewById(R.id.button1);
        b1.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                /* new 一个Intent 对象, 并指定要启动的class */
                Intent intent = new Intent();
            }
        });
    }
}

```



```

        intent.setClass(Ex9_UI.this, SecondActivity.class);
        /* 调用一个新的Activity */
        startActivity(intent);
        /* 关闭原本的Activity */
        Ex9_UI.this.finish();
    }
    });
}
}

```

⑧ 在 AndroidManifest.xml 文件中添加 SecondActivity

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="zyf.Ex9_UI"
    android:versionCode="1"
    android:versionName="1.0">
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name">
        <activity android:name=".Ex9_UI"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="SecondActivity"></activity>
    </application>
    <uses-sdk android:minSdkVersion="2" />
</manifest>

```

⑨ 结果




不同 Activity 之间的数据传递

Bundle 对象的实现

① 新建工程

② 修改 main.xml 布局，添加 UI 元素



```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/widget0"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <TextView
        android:id="@+id/showText"
        android:layout_width="wrap_content"
        android:layout_height="26px"
        android:text="计算你的标准体重!"
        android:textSize="25px"
        android:layout_x="65px"
        android:layout_y="21px">
    </TextView>
    <TextView
        android:id="@+id/text_Sex"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="性别:"
        android:layout_x="71px"
        android:layout_y="103px">
    </TextView>
    <TextView
        android:id="@+id/text_Height"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="身高:"
        android:layout_x="72px"
        android:layout_y="169px">
    </TextView>
    <RadioGroup
        android:id="@+id/radioGroup"
        android:layout_width="wrap_content"
        android:layout_height="37px"
        android:orientation="horizontal"
```



```

        android:layout_x="124px"
        android:layout_y="101px">
        <RadioButton
            android:id="@+id/Sex_Man"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="男">
        </RadioButton>
        <RadioButton
            android:id="@+id/Sex_Woman"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="女">
        </RadioButton>
    </RadioGroup>
    <EditText
        android:id="@+id/height_Edit"
        android:layout_width="123px"
        android:layout_height="wrap_content"
        android:text=""
        android:textSize="18sp"
        android:layout_x="124px"
        android:layout_y="160px">
    </EditText>
    <Button
        android:id="@+id/button_OK"
        android:layout_width="80px"
        android:layout_height="wrap_content"
        android:text="计算"
        android:layout_x="125px"
        android:layout_y="263px">
    </Button>
</AbsoluteLayout>

```

③ 新建 mylayout.xml，并添加 UI 元素

```

<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
    <TextView
        android:id="@+id/text1"

```

```

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_x="50px"
        android:layout_y="72px"
    ></TextView>
</AbsoluteLayout>

```

④ 新建一个 BMIActivity.java

```

package zyf.Ex10_UI;

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

public class BMIActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}

```

⑤ 在 AndroidManifest.xml 添加 Activity 定义

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="zyf.Ex10_UI"
    android:versionCode="1"
    android:versionName="1.0">
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name">
        <activity android:name=".Ex10_UI"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="BMIActivity"></activity>
    </application>
    <uses-sdk android:minSdkVersion="2" />
</manifest>

```


⑥ 修改 BMIActivity.java 内容

```
package zyf.Ex10_UI;

/* import 相关class */
import java.text.DecimalFormat;
import java.text.NumberFormat;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class BMIActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /* 加载main.xml Layout */
        setContentView(R.layout.mylayout);
        /* 取得Intent 中的Bundle 对象 */
        Bundle bunde = this getIntent().getExtras();
        /* 取得Bundle 对象中的数据 */
        String sex = bunde.getString("sex");
        double height = bunde.getDouble("height");
        /* 判断性别 */
        String sexText = "";
        if (sex.equals("M")) {
            sexText = "男性";
        } else {
            sexText = "女性";
        }
        /* 取得标准体重 */
        String weight = this.getWeight(sex, height);
        /* 设置输出文字 */
        TextView tv1 = (TextView) findViewById(R.id.text1);
        tv1.setText("你是一位" + sexText + "\n你的身高是" + height +
            "厘米\n你的标准体重是" + weight + "公斤");
    }

    /* 四舍五入的method */
    private String format(double num) {
        NumberFormat formatter = new DecimalFormat("0.00");
        String s = formatter.format(num);
        return s;
    }
}
```

```

/* 以findViewById()取得Button 对象, 并添加onClickListener */
private String getWeight(String sex, double height) {
    String weight = "";
    if (sex.equals("M")) {
        weight = format((height - 80) * 0.7);
    } else {
        weight = format((height - 70) * 0.6);
    }
    return weight;
}
}

```

⑦ 修改 MainActivity.java 内容

```

package zyf.Ex10_UI;

/* import 相关class */
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;

public class Ex10_UI extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /* 载入main.xml Layout */
        setContentView(R.layout.main);
        /* 以findViewById()取得Button 对象, 并添加onClickListener */
        Button ok = (Button) findViewById(R.id.button_OK);
        ok.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                /* 取得输入的身高 */
                EditText et = (EditText) findViewById(R.id.height_Edit);
                double height = Double.parseDouble(et.getText().toString());
                /* 取得选择的性别 */
                String sex = "";
                RadioButton rb1 = (RadioButton) findViewById(R.id.Sex_Man);
                if (rb1.isChecked()) {
                    sex = "M";
                } else {

```



```

        sex = "F";
    }
    /* new 一个Intent 对象, 并指定class */
    Intent intent = new Intent();
    intent.setClass(Ex10_UI.this, BMIActivity.class);
    /* new 一个Bundle对象, 并将要传递的数据传入 */
    Bundle bundle = new Bundle();
    bundle.putDouble("height", height);
    bundle.putString("sex", sex);
    /* 将Bundle 对象assign 给Intent */
    intent.putExtras(bundle);
    /* 调用Activity EX03_10_1 */
    startActivity(intent);
    }
});
}
}

```

⑧ 结果:




返回数据到前一个 Activity

startActivityForResult 方法

① 新建工程

② 修改 main.xml 布局, 添加 UI 元素



```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/widget0"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <TextView
        android:id="@+id/showText"
        android:layout_width="wrap_content"
        android:layout_height="26px"
        android:text="计算你的标准体重!"
        android:textSize="25px"
        android:layout_x="65px"
        android:layout_y="21px">
    </TextView>
    <TextView
        android:id="@+id/text_Sex"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="性别:"
        android:layout_x="71px"
        android:layout_y="103px">
    </TextView>
    <TextView
        android:id="@+id/text_Height"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="身高:"
        android:layout_x="72px"
        android:layout_y="169px">
    </TextView>
    <RadioGroup
        android:id="@+id/radioGroup"
        android:layout_width="wrap_content"
        android:layout_height="37px"
        android:orientation="horizontal"
```



```

        android:layout_x="124px"
        android:layout_y="101px">
        <RadioButton
            android:id="@+id/Sex_Man"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="男">
        </RadioButton>
        <RadioButton
            android:id="@+id/Sex_Woman"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="女">
        </RadioButton>
    </RadioGroup>
    <EditText
        android:id="@+id/height_Edit"
        android:layout_width="123px"
        android:layout_height="wrap_content"
        android:text=""
        android:numeric="decimal"
        android:textSize="18sp"
        android:layout_x="124px"
        android:layout_y="160px">
    </EditText>
    <Button
        android:id="@+id/button_OK"
        android:layout_width="80px"
        android:layout_height="wrap_content"
        android:text="计算"
        android:layout_x="125px"
        android:layout_y="263px">
    </Button>
</AbsoluteLayout>

```

③ 新建一个 mylayout.xml 布局，添加 UI 元素

```

<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
    android:id="@+id/text1"

```

```

    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:layout_x="50px"
    android:layout_y="72px"

```

```
></TextView>
```

```
<Button
```

```
    android:id="@+id/button_back"
```

```
    android:layout_width="100px"
```

```
    android:layout_height="48px"
```

```
    android:text="回上一页"
```

```
    android:layout_x="110px"
```

```
    android:layout_y="180px"
```

```
></Button>
```

```
</AbsoluteLayout>
```

上一例子中新添加

④ 新建一个 SecondActivity.java 的 Activity 子类

```

package zyf.Ex11_UI_A;

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

public class BMIActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}

```

⑤ 在 AndroidManifest.xml 中添加 SecondActivity 这个 Activity

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="zyf.Ex11_UI_A"
    android:versionCode="1"
    android:versionName="1.0">
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name">
        <activity android:name=".Ex11_UI_A"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

```



```

        </activity>
        <activity android:name="BMIActivity"></activity>
    </application>
    <uses-sdk android:minSdkVersion="2" />
</manifest>

```

必须在 AndroidManifest 中注册新的 Activity，否则程序出错

⑥ 修改 MainActivity.java 代码

```

package zyf.Ex11_UI_A;

import android.app.Activity; /* import 相关class */
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.Toast;

public class Ex11_UI_A extends Activity {
    protected int my_requestCode = 1550;
    private EditText edit_height;
    private RadioButton radiobutton_Man, radiobutton_Woman;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /* 载入main.xml Layout */
        setContentView(R.layout.main);
        /* 以findViewById() 取得Button 对象，并添加onClickListener */
        Button ok = (Button) findViewById(R.id.button_OK);
        edit_height = (EditText) findViewById(R.id.height_Edit);
        radiobutton_Man = (RadioButton) findViewById(R.id.Sex_Man);
        radiobutton_Woman = (RadioButton) findViewById(R.id.Sex_Woman);
        ok.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                try {
                    /* 取得输入的身高 */
                    double height = Double.parseDouble(edit_height.getText().toString());

                    /* 取得选择的性别 */
                    String sex = "";
                    if (radiobutton_Man.isChecked()) {
                        sex = "M";
                    } else {
                        sex = "F";
                    }
                }
            }
        });
    }
}

```



```

        /* new 一个Intent 对象, 并指定class */
        Intent intent = new Intent();
        intent.setClass(Ex11_UI_A.this, BMIActivity.class);
        /* new 一个Bundle对象, 并将要传递的数据传入 */
        Bundle bundle = new Bundle();
        bundle.putDouble("height", height);
        bundle.putString("sex", sex);
        /* 将Bundle 对象assign 给Intent */
        intent.putExtras(bundle);
        /* 调用Activity EX03_10_1 */
        startActivityForResult(intent, my_requestCode);
    } catch (Exception e) {
        // TODO: handle exception
        Toast.makeText(Ex11_UI_A.this,
            R.string.errorString, Toast.LENGTH_LONG).show();
    }
}
});
}

```

新重写方法, 等待返回结果

```

@Override
protected void onActivityResult(int requestCode, int resultCode,
    Intent data) {
    // TODO Auto-generated method stub
    super.onActivityResult(requestCode, resultCode, data);
    switch (resultCode) {
        case RESULT_OK:
            /* 取得来自Activity2 的数据, 并显示于画面上 */
            Bundle bunde = data.getExtras();
            String sex = bunde.getString("sex");
            double height = bunde.getDouble("height");
            edit_height.setText("" + height);
            if (sex.equals("M")) {
                radiobutton_Man.setChecked(true);
            } else {
                radiobutton_Woman.setChecked(true);
            }
            break;
        default:
            break;
    }
}
}

```


⑦ 修改 SecondActivity.java 代码

```
package zyf.Ex11_UI_A;
/* import 相关class */
import java.text.DecimalFormat;
import java.text.NumberFormat;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class BMIActivity extends Activity {
    private Intent intent;
    private Bundle bundle;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /* 加载main.xml Layout */
        setContentView(R.layout.mylayout);

        /* 取得Intent 中的Bundle 对象 */
        intent = this.getIntent();
        bundle = intent.getExtras();

        /* 取得Bundle 对象中的数据 */
        String sex = bundle.getString("sex");
        double height = bundle.getDouble("height");
        /* 判断性别 */
        String sexText = "";
        if (sex.equals("M")) {
            sexText = "男性";
        } else {
            sexText = "女性";
        }
        /* 取得标准体重 */
        String weight = this.getWeight(sex, height);
        /* 设置输出文字 */
        TextView tv1 = (TextView) findViewById(R.id.text1);
        tv1.setText("你是一位" + sexText + "\n你的身高是" + height +
            "厘米\n你的标准体重是"+ weight + "公斤");
    }
}
```

```
/* 以findViewById()取得Button 对象, 并添加onClickListener */
Button b1 = (Button) findViewById(R.id.button_back);
b1.setOnClickListener(new Button.OnClickListener() {
```

```
@Override
```

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

返回刚刚接收的 Intent

```
// TODO Auto-generated method stub
```

```
/* 返回result 回上一个activity */
```

```
BMIActivity.this.setResult(RESULT_OK, intent);
```

```
/* 结束这个activity */
```

```
BMIActivity.this.finish();
```

```
}
```

```
});
```

```
}
```

```
/* 四舍五入的method */
```

```
private String format(double num) {
```

```
    NumberFormat formatter = new DecimalFormat("0.00");
```

```
    String s = formatter.format(num);
```

```
    return s;
```

```
}
```

```
/* 以findViewById()取得Button 对象, 并添加onClickListener */
```

```
private String getWeight(String sex, double height) {
```

```
    String weight = "";
```

```
    if (sex.equals("M")) {
```

```
        weight = format((height - 80) * 0.7);
```

```
    } else {
```

```
        weight = format((height - 70) * 0.6);
```

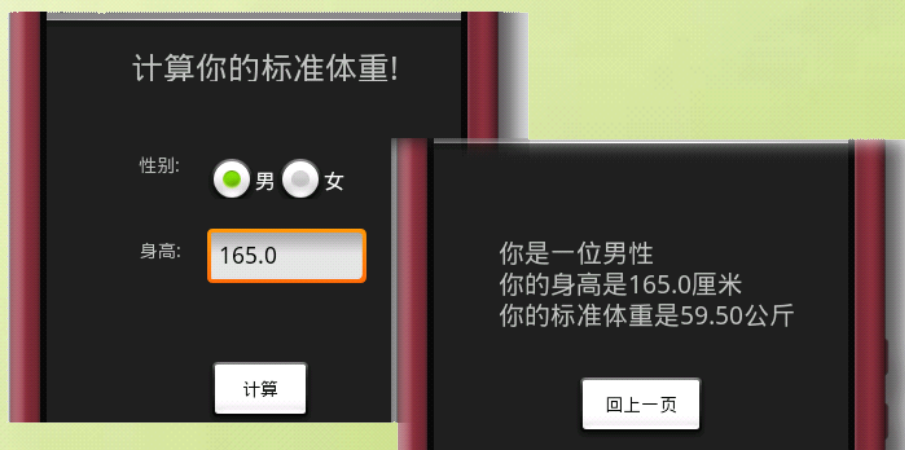
```
    }
```

```
    return weight;
```

```
}
```

```
}
```

⑧ 结果



android
developers



android

android
developers

Android开发教程&笔记



android


Android UI 编程基础 3

EditText 与 TextView 共舞

setOnKeyListener 事件

① 新建工程

② 在 main.xml 布局中添加 UI 元素：一个 EditText 和一个 TextView



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

③ 在 MainActivity.java 主 Activity 中修改代码

```
package zyf.EX_Ctrl_1;

import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

public class EX_Ctrl_1 extends Activity {
    private TextView mTextView01;
```

```
private EditText mEditText01;

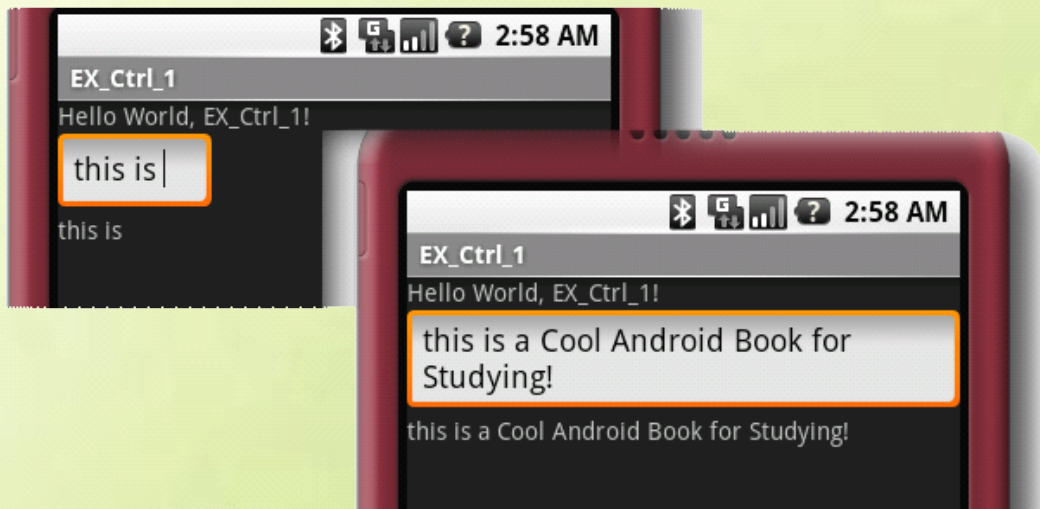
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    /* 取得TextView、EditText */
    mTextView01 = (TextView) findViewById(R.id.myTextView);
    mEditText01 = (EditText) findViewById(R.id.myEditText);

    /* 设置EditText 用OnKeyListener 事件来启动 */
    mEditText01.setOnKeyListener(new EditText.OnKeyListener() {

        @Override
        public boolean onKey(View v, int keyCode, KeyEvent event) {
            // TODO Auto-generated method stub
            mTextView01.setText(mEditText01.getText().toString());
            return false;
        }
    });
}
```

处理 EditText 键入事件

④ 结果

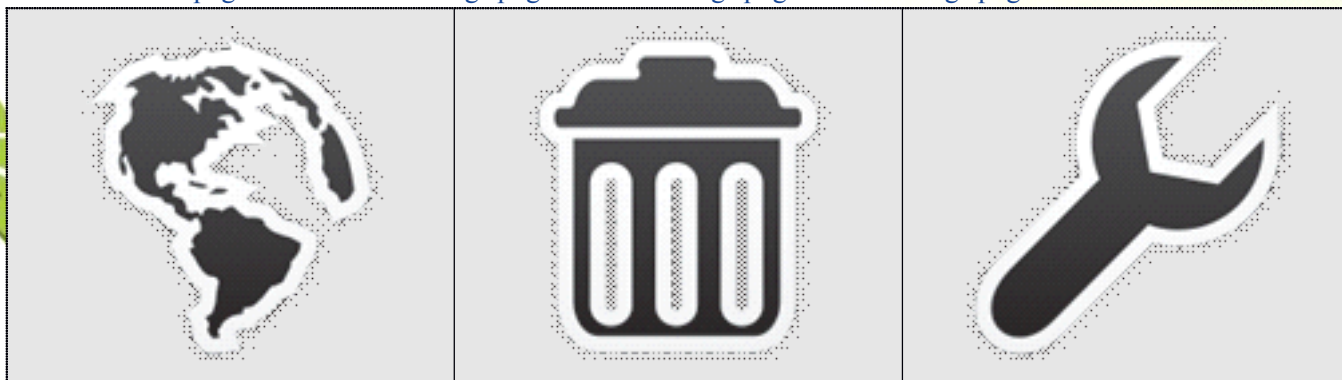


设计具有背景图的按钮

ImageButton 的焦点及事件处理

① 新建工程

② 准备 png 图片资源 clickimage.png、lostfocusimage.png、onfocusimage.png



③ 在 string.xml 中添加字符串

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">EX_Ctrl_2</string>
    <string name="unknown">图片按钮状态: 未知</string>
    <string name="onfocus">图片按钮状态: Got Focus</string>
    <string name="lostfocus">图片按钮状态: Lost Focus</string>
    <string name="onclick">图片按钮状态: Got Click</string>
    <string name="normalbutton">一般按钮</string>
</resources>
```

④ 修改 main.xml 布局, 添加布局元素

```
<?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"
    android:background="#EE559611">
    <TextView
        android:id="@+id/show_TextView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/unknown"
    />
    <ImageButton
        android:id="@+id/image_Button"
```

```

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
    />
    <Button
        android:id="@+id/normal_Button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/normalbutton"
    />
</LinearLayout>

```

⑤ 修改 MainActivity.java

```

package zyf.EX_Ctrl_2;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.View.OnFocusChangeListener;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.TextView;
public class EX_Ctrl_2 extends Activity {
    /** Called when the activity is first created. */
    /* 声明三个对象变量(图片按钮,按钮,与TextView) */
    private ImageButton mImageButton1;
    private Button mButton1;
    private TextView mTextView1;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        /* 通过findViewById 构造三个对象 */
        mImageButton1 = (ImageButton) findViewById(R.id.image_Button);
        mButton1 = (Button) findViewById(R.id.normal_Button);
        mTextView1 = (TextView) findViewById(R.id.show_TextView);
        /* 通过OnFocusChangeListener 来响应ImageButton 的onFous 事件 */
        mImageButton1.setOnFocusChangeListener(new OnFocusChangeListener() {
            public void onFocusChange(View arg0, boolean isFocused) {
                // TODO Auto-generated method stub
                /*若ImageButton 状态为onFocus 改变ImageButton 的图片
                并改变textView 的文字*/
                if (isFocused == true) {
                    mTextView1.setText(R.string.onfocus);
                    mImageButton1.setImageResource(R.drawable.onfocusimage);
                }
            }
        });
    }
}

```



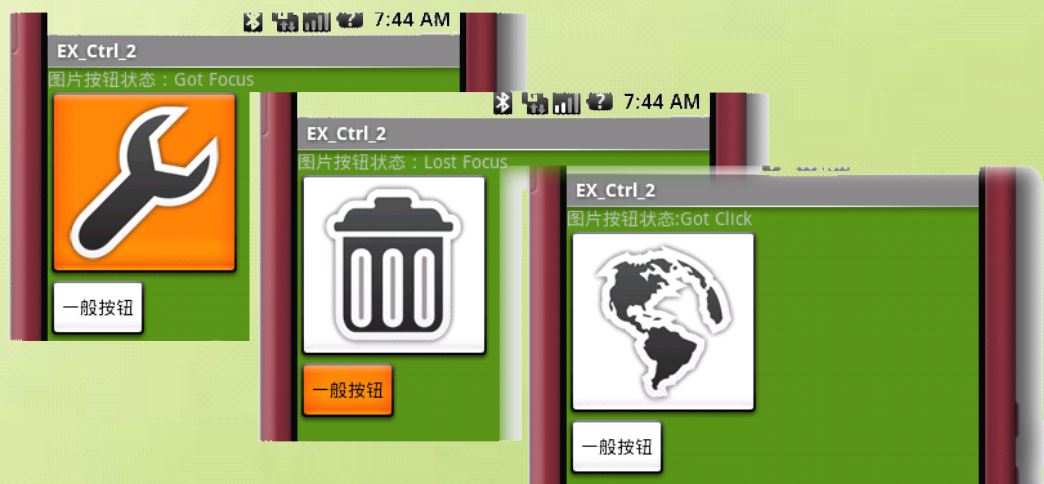
```

    }
    /*若ImageButton 状态为offFocus 改变ImageButton 的图片
    并改变textView 的文字*/
    else {
        mTextView1.setText(R.string.lostfocus);
        mImageButton1.setImageResource(R.drawable.lostfocusimage);
    }
}
});
/* 通过OnClickListener 来响应ImageButton 的onClick 事件 */
mImageButton1.setOnClickListener(new OnClickListener() {
    public void onClick(View v){
        // TODO Auto-generated method stub
        /*若ImageButton 状态为onClick 改变ImageButton 的图片
        并改变textView 的文字*/
        mTextView1.setText(R.string.onclick);
        mImageButton1.setImageResource(R.drawable.clickimage);
    }
});

/*通过OnClickListener 来响应Button 的onClick 事件*/
mButton1.setOnClickListener(new OnClickListener(){
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        /*若Button 状态为onClick 改变ImageButton 的图片
        * 并改变textView 的文字*/
        mTextView1.setText(R.string.lostfocus);
        mImageButton1.setImageResource(R.drawable.lostfocusimage);
    }
});
}
}
}

```

⑥ 结果



使用 XML 实现按钮改变焦点设置背景图

Button 的 android:background 属性

◆ 设置按钮背景图片:

- onFocus() 与 onClick() 事件处理
- Item 的 android:state_focused 和 android:state_pressed 属性

◆ 实现

① 新建工程

② 准备 png 背景图片 *defaultimage.png*、*onfocusimage.png*、*clickimage.png**clickimage**defaultimage.png**onfocusimage.png*③ 在 *main.xml* 布局中添加一个 *ImageButton* 和一个 *Button* (作对比)

```
<?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" android:background="#FF25CCDD">

    <ImageButton
        android:id="@+id/image_Button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

    <Button
        android:text="Button01"
        android:id="@+id/Button01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
</Button>
</LinearLayout>
```


Android 编程基础

④ 在 `drawable` 文件夹中添加一个 `advancedbutton.xml` 设置 `<selector>` 和 `<item>` 标签

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

获得焦点时

获得焦点并按下

失去焦点时

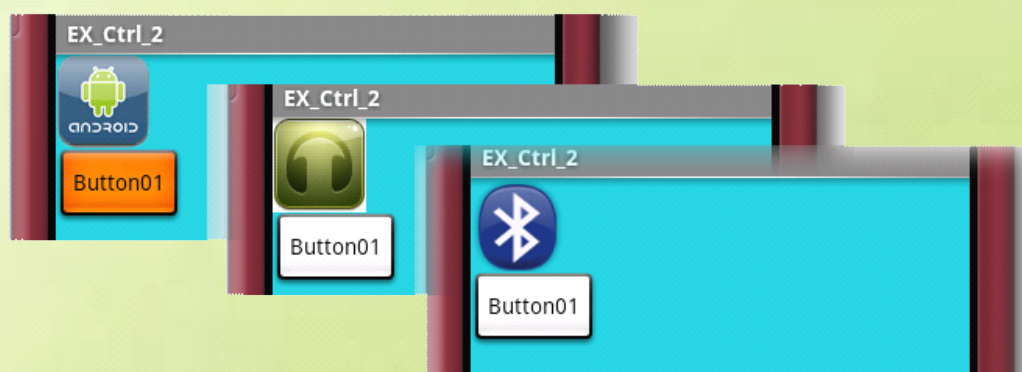
默认时

⑤ 设置 `ImageButton` 的 `android:background` 属性值

```
<ImageButton
    android:id="@+id/image_Button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/advancedbutton"
/>
```

利用 XML 来改变按钮背景图片

⑥ 结果




给圣诞老人的信息

Toast--Android 专属浮动小提示

① 新建工程

② 在 string.xml 中添加字符串



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">EX_Ctrl_3</string>
    <string name="dear">亲爱的圣诞老人: </string>
    <string name="sendmyWish">送出愿望</string>
    <string name="yourWish">你的愿望: </string>
    <string name="hasSend">已送达圣诞老人信箱!</string>
</resources>
```

③ 在 main.xml 布局中添加 UI 元素: 一个 TextView、一个 EditText 和一个 Button

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

    <EditText
        android:layout_height="wrap_content"
        android:id="@+id/EditText_Wish"
        android:layout_width="fill_parent"></EditText>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/Button_Send"
        android:text="@string/sendmyWish"
        android:layout_marginLeft="50px"></Button>

</LinearLayout>
```


④ 修改 MainActivity.java 文件

```
package zyf.EX_Ctrl_3;

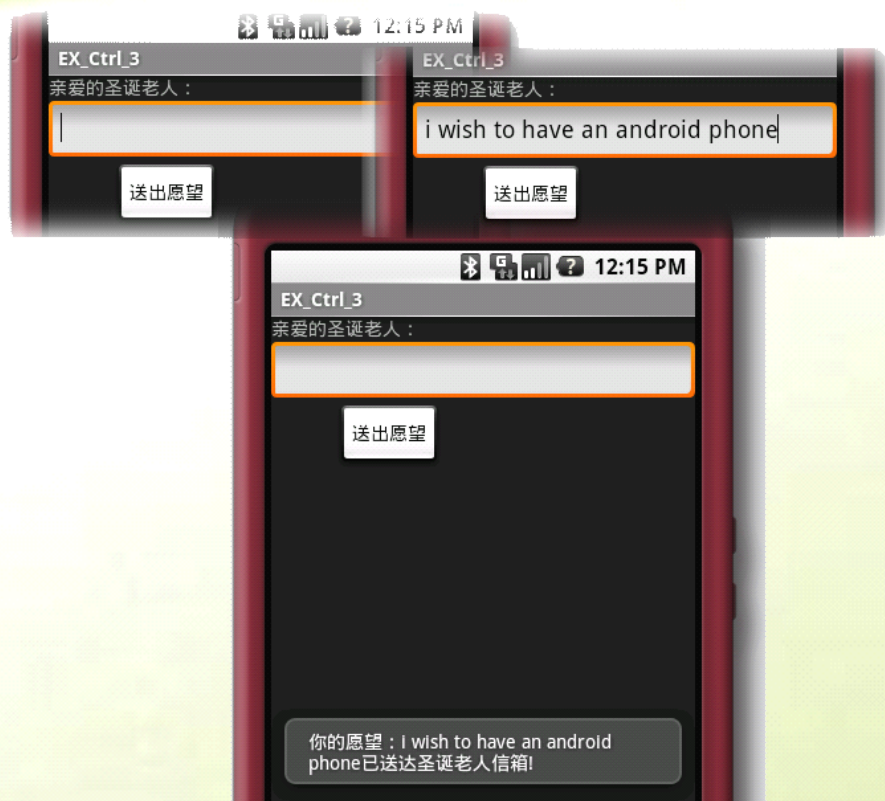
import android.app.Activity;
import android.os.Bundle;
import android.text.Editable;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class EX_Ctrl_3 extends Activity {
    /** Called when the activity is first created. */
    /*声明两个对象变量(按钮与编辑文字)*/
    private Button mButton;
    private EditText mEditText;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        /*通过findViewById()取得对象*/
        mButton=(Button)findViewById(R.id.Button_Send);
        mEditText=(EditText)findViewById(R.id.EditText_Wish);
        /*设置OnClickListener 给Button 对象聆听onClick 事件*/
        mButton.setOnClickListener(new Button.OnClickListener() {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                /*声明字符串变量并取得用户输入的EditText 字符串*/
                Editable Str;
                Str=mEditText.getText();
                /*使用CharSequence类getString()方法从XML中获取String*/
                CharSequence string2=getString(R.string.yourWish);
                CharSequence string3=getString(R.string.hasSend);
                /*使用系统标准的makeText()方式来产生Toast 信息*/

                Toast.makeText( EX_Ctrl_3.this,string2+Str.toString()+string3,Toast.LENGTH
_LONG).show();

                /*清空EditText*/
                mEditText.setText("");
            }
        });
    }
}
```

⑤ 结果



Toast—显示 View 的提示

Toast 显示一个 ImageView

① 新建工程

② 在 drawable 文件夹中添加一副 png 图片：argon.png



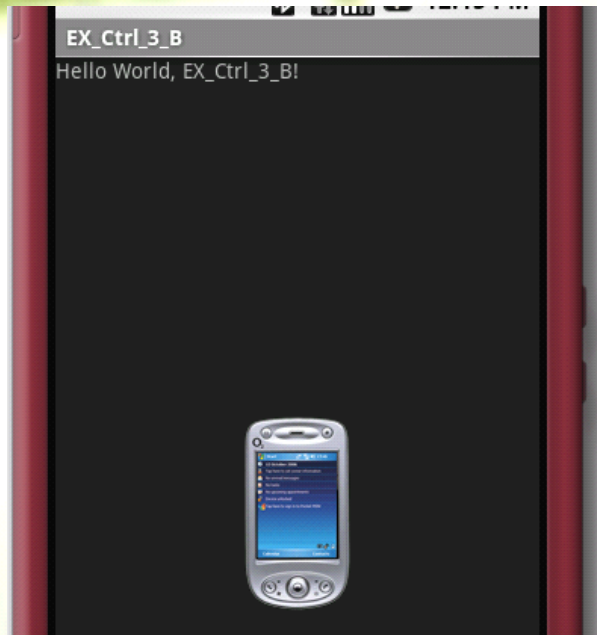
③ 修改 MainActivity.java 文件

```
package zyf.EX_Ctrl_3_B;

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

public class EX_Ctrl_3_B extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /*设置主屏布局*/
        setContentView(R.layout.main);
        /*创建新Toast对象*/
        Toast showImageToast=new Toast(this);
        /*创建新ImageView对象*/
        ImageView imageView=new ImageView(this);
        /*从资源中获取图片*/
        imageView.setImageResource(R.drawable.argon);
        /*设置Toast上的View--(ImageView)*/
        showImageToast.setView(imageView);
        /*设置Toast显示时间*/
        showImageToast.setDuration(Toast.LENGTH_LONG);
        /*显示Toast*/
        showImageToast.show();
    }
}
```

④ 结果

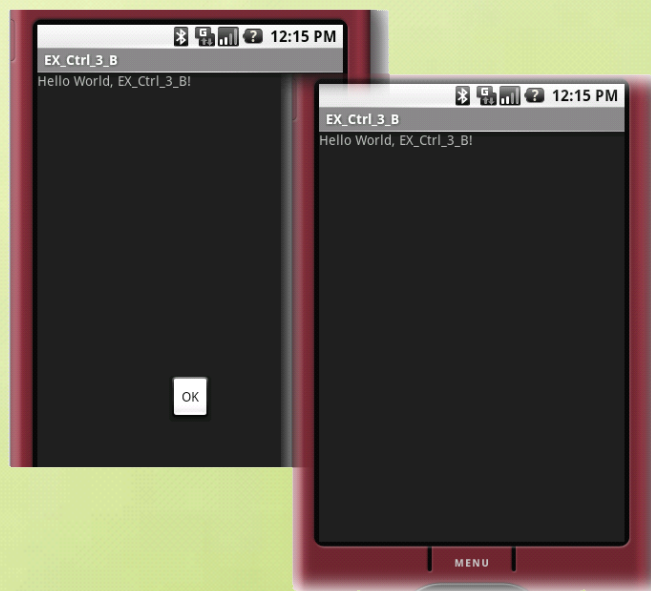


Toast 显示一个 Button

- ① 新建工程
- ② 修改 MainActivity.java 文件

```
package zyf.EX_Ctrl_3_B;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
public class EX_Ctrl_3_B extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /* 设置主屏布局 */
        setContentView(R.layout.main);
        /* 创建新Toast对象 */
        Toast showImageToast = new Toast(this);
        // /*新建Button对象*/
        Button button = new Button(this);
        button.setText("OK");
        /* 设置Toast上的View--(Button) */
        showImageToast.setView(button);
        /* 设置Toast显示时间 */
        showImageToast.setDuration(Toast.LENGTH_LONG);
        /* 显示Toast */
        showImageToast.show();
    }
}
```


- ③ 结果



Toast 显示一个 TextView

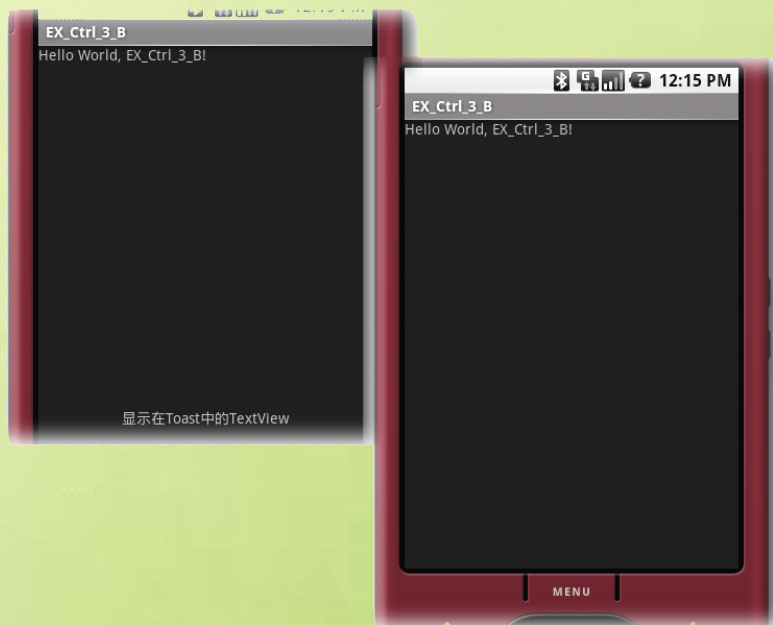
① 新建工程

② 修改 MainActivity.java 文件



```
package zyf.EX_Ctrl_3_B;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
import android.widget.Toast;
public class EX_Ctrl_3_B extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /* 设置主屏布局 */
        setContentView(R.layout.main);
        /* 创建新Toast对象 */
        Toast showImageToast = new Toast(this);
        /*新建TextView对象*/
        TextView text=new TextView(this);
        /*设置TextView内容*/
        text.setText("显示在Toast中的TextView");
        /* 设置Toast上的View-- (TextView) */
        showImageToast.setView(text);
        /* 设置Toast显示时间 */
        showImageToast.setDuration(Toast.LENGTH_LONG);
        /* 显示Toast */
        showImageToast.show();
    }
}
```

③ 结果



AlertDialog.Builder 提示对话框

① 新建工程

② 修改 MainActivity.java 文件

```
package zyf.EX_Ctrl_3_B;
import android.app.Activity;
import android.app.AlertDialog;
import android.os.Bundle;
public class EX_Ctrl_3_B extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /* 设置主屏布局 */
        setContentView(R.layout.main);
        /*新建一个AlertDialog.Builder对象*/
        AlertDialog.Builder my_ADIALOG = new AlertDialog.Builder(this);
        /*设置标题*/
        my_ADIALOG.setTitle("Android 提示");
        /*设置显示消息*/
        my_ADIALOG.setMessage("AlertDialog.Builder提示对话框消息!!");
        /*显示*/
        my_ADIALOG.show();
    }
}
```

③ 结果



同意条款

CheckBox 的 isChecked 属性

① 创建新工程

② 在 string.xml 中添加字符串

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Ex_Ctrl_4</string>
    <string name="advice">请勾选我同意</string>
    <string name="accept">您已接受同意!! </string>
    <string name="Notaccept">您未同意!! </string>
    <string name="allOK">全部许可</string>
</resources>
```

③ 修改 main.xml 布局, 添加 UI 元素

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <TextView
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:id="@+id/TextView_Guide"
        android:textSize="25px" android:text="@string/advice"></TextView>
    <TextView
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:id="@+id/TextView_youChoiceShow"
        android:textSize="25px"></TextView>
    <CheckBox
        android:layout_height="wrap_content"
        android:layout_marginTop="100px"
        android:layout_marginLeft="90px"
        android:id="@+id/CheckBox_Accept" android:layout_width="120px"></CheckBox>
    <Button
        android:layout_height="wrap_content"
        android:layout_marginLeft="90px"
        android:id="@+id/Button_OK"
        android:text="确定"
        android:layout_width="90px"></Button>
</LinearLayout>
```


④ 修改 MainActivity.java 文件

```
package zyf.Ex_Ctrl_4;

import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.TextView;

public class Ex_Ctrl_4 extends Activity {
    /** Called when the activity is first created. */
    private TextView showAdvice, yourChoiceshow;
    private CheckBox iAccept;
    private Button ok;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        /*findViewById() 从资源ID获取资源对象*/
        showAdvice=(TextView) findViewById(R.id.TextView_Guide);
        yourChoiceshow=(TextView) findViewById(R.id.TextView_youChoiceShow);
        iAccept=(CheckBox) findViewById(R.id.CheckBox_Accept);
        ok=(Button) findViewById(R.id.Button_OK);

        /*获取XML中字符串*/
        CharSequence titleString=getString(R.string.allOK);
        /*设置复选框标题*/
        iAccept.setHint(titleString);
        /*设置复选框标题字体颜色*/
        iAccept.setHintTextColor(Color.RED);
        /*将CheckBox设置成未选中*/
        iAccept.setChecked(false);
        /*将Button设置成不可选*/
        ok.setEnabled(false);
        iAccept.setOnClickListener(new CheckBox.OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                if(iAccept.isChecked()) {
```

```

        ok.setEnabled(true);
        yourChoiceshow.setText(R.string.accept);
    }else{
        ok.setEnabled(false);
        yourChoiceshow.setText(R.string.Notaccept);
    }
}

});
ok.setOnClickListener(new Button.OnClickListener(){

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        if(iAccept.isChecked()){
            showAdvice.setText(R.string.accept);
        }
    }
});
}
}

```

⑤ 结果




购物列表

多选项 CheckBox 的应用

① 新建工程

② 在 string.xml 中添加字符串



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Ex_Ctrl_5</string>
    <string name="shoopngList_TextView_text">网上购物商品清单</string>
    <string name="yourChocieWoods_text">您选择购买的商品: </string>
    <string name="woods_Text_MP4">纽曼MP4</string>
    <string name="woods_Text_musicCD">Beyond乐队CD</string>
    <string name="woods_Text_book">Android程序员指南</string>
</resources>
```

③ 修改 main.xml 布局, 添加 UI 元素

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:id="@+id/TextView_shoppingList"
        android:text="@string/shoopngList_TextView_text"
        android:textSize="25px"></TextView>
    <CheckBox
        android:layout_height="wrap_content"
        android:id="@+id/CheckBox_MP4"
        android:text="@string/woods_Text_MP4"
        android:layout_width="180px"></CheckBox>
    <CheckBox
        android:layout_height="wrap_content"
        android:text="@string/woods_Text_musicCD"
        android:id="@+id/CheckBox_musicCD"
        android:layout_width="180px"></CheckBox>
    <CheckBox
        android:layout_height="wrap_content"
        android:text="@string/woods_Text_book"
```

```

        android:id="@+id/CheckBox_book"
        android:layout_width="180px"></CheckBox>
<TextView
    android:layout_height="wrap_content"
    android:layout_width="fill_parent"
    android:id="@+id/TextView_yourChoice"
    android:text="@string/yourChocieWoods_text"
    android:textSize="20px"></TextView>
<TextView
    android:layout_height="wrap_content"
    android:id="@+id/TextView_yourWoodsList"
    android:layout_width="fill_parent"
    android:textSize="20px"></TextView>
</LinearLayout>

```

④ 修改 MainActivity.java, 添加逻辑判断

```

package zyf.Ex_Ctrl_5;

import android.app.Activity;
import android.os.Bundle;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.TextView;
import android.widget.CompoundButton.OnCheckedChangeListener;

public class Ex_Ctrl_5 extends Activity {
    /** Called when the activity is first created. */
    private TextView showyourChoice_TextView;
    private CheckBox mp4_CheckBox, musicCD_CheckBox, book_CheckBox;
    private String showinfo;

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

        /* findViewById() 从XML中获取资源对象 */
        showyourChoice_TextView = (TextView)
findViewById(R.id.TextView_yourWoodsList);
        mp4_CheckBox = (CheckBox) findViewById(R.id.CheckBox_MP4);
        musicCD_CheckBox = (CheckBox) findViewById(R.id.CheckBox_musicCD);
        book_CheckBox = (CheckBox) findViewById(R.id.CheckBox_book);

        /* 为三个CheckBox设置选择状态改变事件监听器 */
    }
}

```



```

mp4_CheckBox.setOnCheckedChangeListener (CheckedChangeListener);
musicCD_CheckBox.setOnCheckedChangeListener (CheckedChangeListener);
book_CheckBox.setOnCheckedChangeListener (CheckedChangeListener);

/* 从XML中获取显示信息String */
showinfo = getString(R.string.yourChocieWoods_text);
}

/* 内部接口实现 */
private OnCheckedChangeListener CheckedExceptionListener = new
OnCheckedChangeListener() {

    @Override
    public void onCheckedChanged(CompoundButton buttonView,
        boolean isChecked) {
        // TODO Auto-generated method stub
        /* 处理选中状态改变事件，动态显示选择结果 */
        if (mp4_CheckBox.isChecked()) {
            showinfo = getString(R.string.woods_Text_MP4) + "\n";
            showString();
        } else if (musicCD_CheckBox.isChecked()) {
            showinfo = getString(R.string.woods_Text_musicCD) + "\n";
            showString();
        } else if (book_CheckBox.isChecked()) {
            showinfo = getString(R.string.woods_Text_book) + "\n";
            showString();
        }
        if (mp4_CheckBox.isChecked() && musicCD_CheckBox.isChecked()) {
            showinfo = getString(R.string.woods_Text_MP4) + "\n"
                + getString(R.string.woods_Text_musicCD) + "\n";
            showString();
        } else if (mp4_CheckBox.isChecked() && book_CheckBox.isChecked()) {
            showinfo = getString(R.string.woods_Text_MP4) + "\n"
                + getString(R.string.woods_Text_book) + "\n";
            showString();
        } else if (musicCD_CheckBox.isChecked()
            && book_CheckBox.isChecked()) {
            showinfo = getString(R.string.woods_Text_musicCD) + "\n"
                + getString(R.string.woods_Text_book) + "\n";
            showString();
        }
        if (mp4_CheckBox.isChecked() && musicCD_CheckBox.isChecked()
            && book_CheckBox.isChecked()) {
            showinfo = getString(R.string.woods_Text_MP4) + "\n"

```

```

        + getString(R.string.woods_Text_musicCD) + "\n"
        + getString(R.string.woods_Text_book) + "\n";
    showString();
}
if (mp4_CheckBox.isChecked() == false
    && musicCD_CheckBox.isChecked() == false
    && book_CheckBox.isChecked() == false) {
    showyourChoice_TextView.setText("");
}
}

public void showString() {
    showyourChoice_TextView.setText(showinfo);
}
};
}

```

⑤ 结果



RadioButton 单选

RadioGroup 组与 onCheckedChanged 事件

① 新建工程

② string.xml 中添加字符串

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Ex_Ctrl_6</string>
    <string name="iam_Boy">帅哥</string>
    <string name="iamGirl">美女</string>
    <string name="ask">请问你是? ? </string>
</resources>
```

③ 修改 mian.xml 布局, 添加 UI 元素

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/ask"
        android:id="@+id/TextView_Ask_And_Show"
        android:textSize="25px"/>
    <RadioGroup
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/RadioGroup">
        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/RadioButton_Boy"
            android:text="@string/iam_Boy"></RadioButton>
        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/iamGirl"
            android:id="@+id/RadioButton_Girl"></RadioButton>
    </RadioGroup>
</LinearLayout>
```

④ 修改 MainActivity.java 文件

```
package zyf.Ex_Ctrl_6;

import android.app.Activity;
import android.os.Bundle;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;

public class Ex_Ctrl_6 extends Activity {
    /** Called when the activity is first created. */
    private TextView answer_TextView;
    private RadioButton boy_RadioButton, girl_RadioButton;
    private RadioGroup radioGroup;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        /* findViewById() 从XML中获取资源对象 */
        answer_TextView=(TextView) findViewById(R.id.TextView_Ask_And_Show);
        radioGroup=(RadioGroup) findViewById(R.id.RadioGroup);
        boy_RadioButton=(RadioButton) findViewById(R.id.RadioButton_Boy);
        girl_RadioButton=(RadioButton) findViewById(R.id.RadioButton_Girl);

        /*给单RadioGroup添加状态改变监听器*/
        radioGroup.setOnCheckedChangeListener(new
                                   RadioGroup.OnCheckedChangeListener() {

            @Override
            public void onCheckedChanged(RadioGroup group, int checkedId) {
                // TODO Auto-generated method stub
                if(boy_RadioButton.isChecked()){
                    answer_TextView.setText(R.string.iam_Boy);
                }else{
                    answer_TextView.setText(R.string.iamGirl);
                }
            }
        });
    }
}
```


⑤ 结果



RadioButton 猜猜看

猜猜我是??

① 新建项目

② 准备三张 png 图片

ic_menu_close_clear_cancel.png 	right.png 	wrong.png 
---	--	--

③ 在 string.xml 中添加要使用的字符串

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Ex_Ctrl_6</string>
    <string name="iam_Boy">帅哥</string>
    <string name="iamGirl">美女</string>
    <string name="ask">猜猜我是?? </string>
    <string name="answer_Q">回答</string>
    <string name="clean">清空</string>
    <string name="showAnswerTitle_YES">恭喜!</string>
    <string name="showAnswerTitle_NO">很遗憾!</string>
    <string name="about_dialog_ok">OK</string>
    <string name="right">您答对了!!</string>
    <string name="wrong">哈哈, 错误!</string>
    <string name="answerIS">正确答案是: </string>
</resources>
```

④ 在 main.xml 中添加 UI 元素

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/ask"
        android:id="@+id/TextView_Ask_And_Show"
        android:textSize="25px"/>
```




```
<RadioGroup
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/RadioGroup">
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/RadioButton_Boy"
        android:text="@string/iam_Boy">
    </RadioButton>
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/iamGirl"
        android:id="@+id/RadioButton_Girl">
    </RadioButton>
</RadioGroup>
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <Button
        android:id="@+id/answer_The_Q_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/answer_Q"
    ></Button>
    <Button
        android:id="@+id/clean_The_Q_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/clean"
    ></Button>
</LinearLayout>
</LinearLayout>
```

⑤ 修改 MainActivity.java 文件

```
package zyf.Ex_Ctrl_6;

import java.util.Random;
import android.app.Activity;
import android.app.AlertDialog;
import android.os.Bundle;
import android.view.Menu;
```

```
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.Toast;

public class Ex_Ctrl_6 extends Activity implements Button.OnClickListener {

    /** Called when the activity is first created. */
    private TextView answer_TextView;
    private RadioButton boy_RadioButton, girl_RadioButton;
    private RadioGroup radioGroup;
    private Button answer_Button, clean_Button;
    private String[] boy_girl;
    private AlertDialog.Builder showRightorNot;
    private MenuItem exit;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        /* 结果显示对话框 */
        showRightorNot = new AlertDialog.Builder(this);

        /* findViewById() 从XML中获取资源对象 */
        answer_TextView = (TextView) findViewById(R.id.TextView_Ask_And_Show);
        radioGroup = (RadioGroup) findViewById(R.id.RadioGroup);
        boy_RadioButton = (RadioButton) findViewById(R.id.RadioButton_Boy);
        girl_RadioButton = (RadioButton) findViewById(R.id.RadioButton_Girl);
        answer_Button = (Button) findViewById(R.id.answer_The_Q_button);
        clean_Button = (Button) findViewById(R.id.clean_The_Q_button);

        /* 答案数组 */
        boy_girl = new String[] { "Boy", "Girl", "Boy", "Girl",
                                   "Boy", "Girl", "Boy", "Girl",
                                   "Boy" };

        /* 按钮设置成不可选 */
        answer_Button.setEnabled(false);
        clean_Button.setEnabled(false);
    }
}
```



```

/* 给单RadioGroup添加状态改变监听器 */
radioGroup.setOnCheckedChangeListener(new
                                RadioGroup.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        // TODO Auto-generated method stub
        /* 判断显示 */
        if (boy_RadioButton.isChecked()) {
            answer_TextView.setText(R.string.iam_Boy);
        } else {
            answer_TextView.setText(R.string.iamGirl);
        }
    }
});

boy_RadioButton.setOnClickListener(new RadioGroup.OnClickListener() {

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        /* 按钮设置成可选 */
        answer_Button.setEnabled(true);
        clean_Button.setEnabled(true);
    }
});

girl_RadioButton.setOnClickListener(new RadioGroup.OnClickListener() {

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        /* 按钮设置成可选 */
        answer_Button.setEnabled(true);
        clean_Button.setEnabled(true);
    }
});

/* 设置按钮事件监听器 */
answer_Button.setOnClickListener(this);
clean_Button.setOnClickListener(this);
}

```

```
@Override
public void onClick(View v) {
    // TODO Auto-generated method stub
    if (v.getId() == R.id.answer_The_Q_button) {
        if (boy_RadioButton.isChecked()) {
            checkTheAnswer("Boy");
        } else if (girl_RadioButton.isChecked()) {
            checkTheAnswer("Girl");
        }
    }
    if (v.getId() == R.id.clean_The_Q_button) {
        /* 按钮设置成未选取 */
        boy_RadioButton.setChecked(false);
        girl_RadioButton.setChecked(false);

        /* 按钮设置成不可选 */
        answer_Button.setEnabled(false);
        clean_Button.setEnabled(false);
        answer_TextView.setText(R.string.ask);
    }
}

private void checkTheAnswer(String checkstring) {
    // TODO Auto-generated method stub
    /*检测提示*/
    Toast.makeText(this, "检测答案.....", Toast.LENGTH_SHORT).show();
    /*获取随机数*/
    Random random = new Random();
    int index = random.nextInt(9);

    if (boy_girl[index].equals(checkstring)) {
        /*回答正确，设置提示对话框，显示结果*/
        showRightorNot.setIcon(R.drawable.right);
        showRightorNot.setTitle(R.string.showAnswerTitle_YES);
        showRightorNot.setPositiveButton(R.string.about_dialog_ok, null);
        showRightorNot.setMessage(R.string.right).show();
        Toast.makeText(this, getString(R.string.answerIS)
            +boy_girl[index], Toast.LENGTH_LONG).show();
    } else {
        /*回答错误，设置提示对话框，显示结果*/
        showRightorNot.setIcon(R.drawable.wrong);
        showRightorNot.setTitle(R.string.showAnswerTitle_NO);
        showRightorNot.setPositiveButton(R.string.about_dialog_ok, null);
    }
}
```



```

        showRightorNot.setMessage(R.string.wrong).show();

        Toast.makeText(this, getString(R.string.answerIS) +
            boy_girl[index], Toast.LENGTH_LONG).show();
    }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // TODO Auto-generated method stub
    /*添加退出菜单*/
    exit=menu.add("Exit");
    /*设置退出菜单图片*/
    exit.setIcon(R.drawable.ic_menu_close_clear_cancel);
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // TODO Auto-generated method stub
    /*结束Activity*/
    finish();
    return super.onOptionsItemSelected(item);
}
}

```

⑥ 结果



android
developers



android

android
developers

Android开发教程&笔记



android

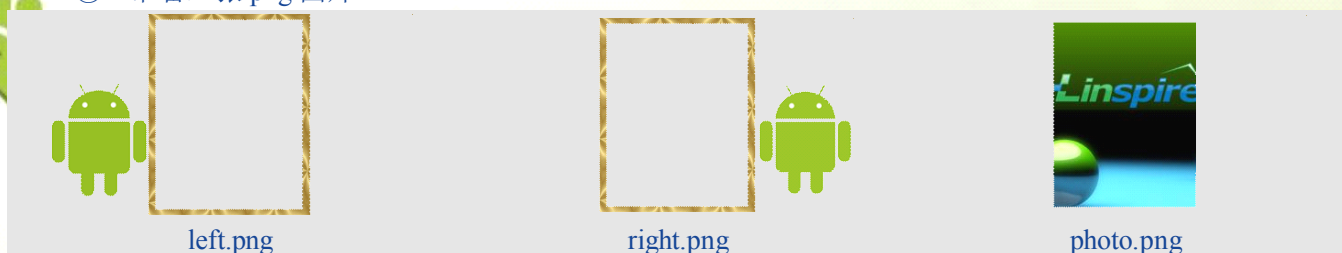
Android 基础 UI 编程 4

专业相框设计

ImageView 的堆叠应用

① 新建工程

② 准备三张 png 图片



③ 修改 main.xml 布局，添加 UI 元素

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/widget34"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
><!--创建第一个ImageView (第二层图片)-->
<ImageView
    android:id="@+id/myImageView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="0px"
    android:layout_y="36px"
/>
<!--创建第二个ImageView (第一层图片)-->
<ImageView
    android:id="@+id/myImageView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="0px"
    android:layout_y="36px"
/>
<!--创建第一个Button -->
<Button
```



```

        android:id="@+id/myButton1"
        android:layout_width="105px"
        android:layout_height="66px"
        android:text="pic1"
        android:layout_x="9px"
        android:layout_y="356px"
    />
<!--创建第二个Button -->
<Button
    android:id="@+id/myButton2"
    android:layout_width="105px"
    android:layout_height="66px"
    android:text="pic2"
    android:layout_x="179px"
    android:layout_y="356px"
/>
</AbsoluteLayout>

```

④ 修改 MainActivity.java

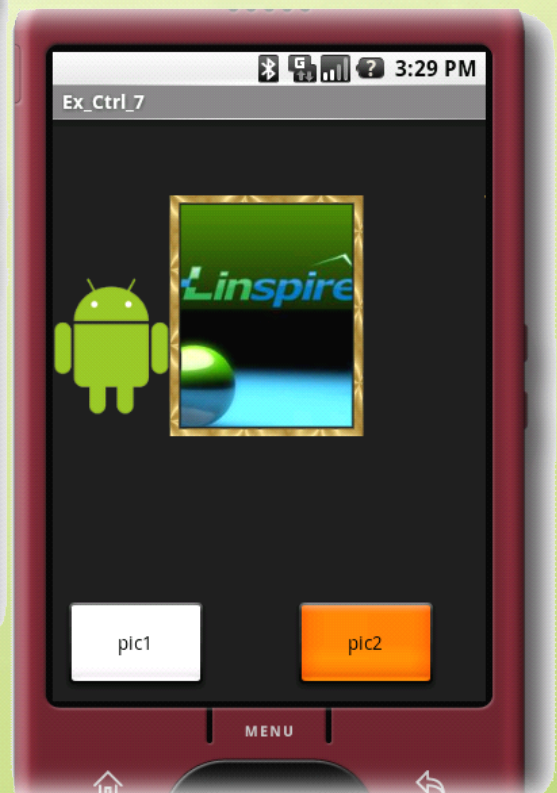
```

package zyf.Ex_Ctrl_7;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
public class Ex_Ctrl_7 extends Activity {
    /** Called when the activity is first created. */
    /* 声明Button、ImageView 对象 */
    private ImageView mImageView01;
    private ImageView mImageView02;
    private Button mButton01;
    private Button mButton02;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        /* 取得Button、ImageView 对象 */
        mImageView01 = (ImageView) findViewById(R.id.myImageView1);
        mImageView02 = (ImageView) findViewById(R.id.myImageView2);
        mButton01 = (Button) findViewById(R.id.myButton1);
        mButton02 = (Button) findViewById(R.id.myButton2);
        /* 设置ImageView 背景图 */
        mImageView01.setImageDrawable(getResources().getDrawable(
            R.drawable.right));
    }
}

```

```
mImageView02.setImageDrawable(getResources().getDrawable(
    R.drawable.photo));
/* 用OnClickListener 事件来启动 */
mButton01.setOnClickListener(new Button.OnClickListener() {
    @Override
    public void onClick(View v) {
        /* 当启动后, ImageView 立刻换背景图 */
        mImageView01.setImageDrawable(getResources().getDrawable(
            R.drawable.right));
    }
});
mButton02.setOnClickListener(new Button.OnClickListener() {
    @Override
    public void onClick(View v) {
        mImageView01.setImageDrawable(getResources().getDrawable(
            R.drawable.left));
    }
});
}
```

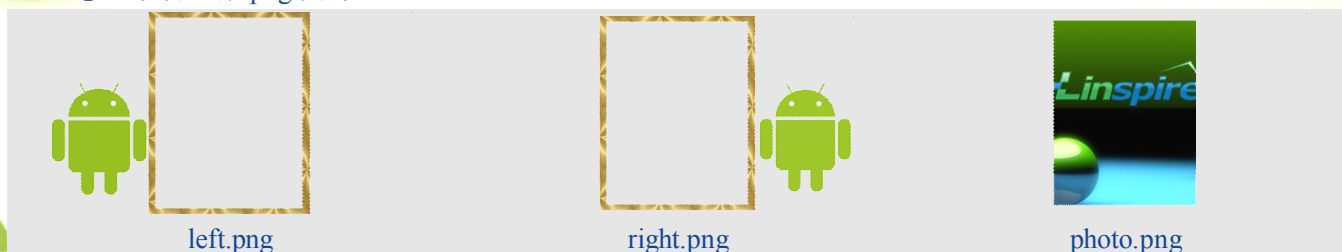
⑤ 结果



ImageButton 的堆叠应用

① 新建项目

② 准备三张 png 图片



③ 修改 main.xml 布局，添加 UI 元素

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <ImageButton
        android:id="@+id/myImageButton_Back"
        android:state_focused="true"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_x="0px"
        android:layout_y="36px"
    />
    <ImageButton
        android:id="@+id/myImageButton_Photo"
        android:state_focused="true"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_x="0px"
        android:layout_y="36px"
    />
</AbsoluteLayout>
```

设置成堆叠

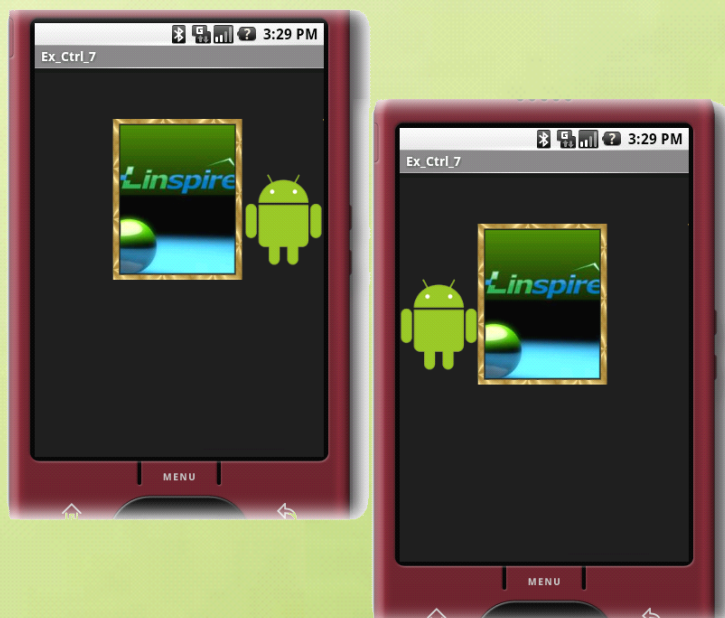
④ 修改 MainActivity.java

```
package zyf.Ex_Ctrl_7_B;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageButton;
```

```
public class Ex_Ctrl_7_B extends Activity {
    /** Called when the activity is first created. */
    /*声明 ImageButton*/
    private ImageButton back_Imagebutton,photo_Imagebutton;
    private boolean Tag=true;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        /*从XML中获取控件对象*/
        back_Imagebutton=(ImageButton)findViewById(R.id.myImageButton_Back);
        photo_Imagebutton=(ImageButton)findViewById(R.id.myImageButton_Photo);

        //设置默认的背景图片
        back_Imagebutton.setBackgroundResource(R.drawable.left);
        photo_Imagebutton.setBackgroundResource(R.drawable.photo);
        //给ImageButton设置事件监听器
        photo_Imagebutton.setOnClickListener(new ImageButton.OnClickListener() {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                Tag=!Tag;//更改背景图片
                if(Tag){
                    back_Imagebutton.setBackgroundResource(R.drawable.right);
                }else{
                    back_Imagebutton.setBackgroundResource(R.drawable.left);
                }
            }
        });
    }
}
```


⑤ 结果



自定义下拉菜单

Spinner 与 setDropDownViewResource

① 新建项目，在 res 目录下新建一个 anim 文件夹，存放动画效果用，在其中新建一个 my_anim.xml 文件



```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:fromXDelta="0"
        android:toXDelta="-100%p"
        android:duration="300">
    </translate><!--位置转换动画效果 -->
    <alpha
        android:fromAlpha="3.0"
        android:toAlpha="6.0"
        android:duration="300">
    </alpha><!--透明度转换动画效果 -->
</set>
```

② 在 res 目录下的 layout 文件夹中新建一个 myspinner_dropdown.xml 文件，用来存放下拉菜单弹出内容的布局

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/text1"
    android:layout_width="wrap_content"
    android:layout_height="24sp"
    android:singleLine="true"
    style="?android:attr/spinnerDropDownItemStyle"
/>
```

③ 修改 main.xml，添加 UI 元素

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:id="@+id/widget28"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android">
<TextView
    android:id="@+id/TextView_Show"
    android:layout_width="fill_parent"
```

```

    android:layout_height="wrap_content"
    android:text="你选择的是"
    android:textSize="25sp">
</TextView>
<Spinner
    android:id="@+id/spinner_City"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
</Spinner><!-- 定义一个下拉菜单 -->
</LinearLayout>

```

④ 定义一个下拉菜单

```

private Spinner mySpinner;
/* 以 findViewById() 取得对象 */
mySpinner = (Spinner) findViewById(R.id.spinner_City);

```

⑤ 定义一个字符串数组和一个 ArrayAdapter

```

private static final String[] countriesStr =
{ " 北京市 ", " 上海市 ", " 天津市 ", " 重庆市 " };
private ArrayAdapter<String> adapter;

adapter=new ArrayAdapter<String>(this,
                                android.R.layout.simple_spinner_item, countriesStr);

```

⑥ 给下拉菜单内容设置样式

```

/* myspinner_dropdown 为自定义下拉菜单样式定义在 res/layout 目录下 */
adapter.setDropDownViewResource(R.layout.myspinner_dropdown);

```

⑦ 给下拉菜单设置内容适配器

```

/* 将 ArrayAdapter 添加 Spinner 对象中 */
mySpinner.setAdapter(adapter);

```

⑧ 给下拉菜单添加动画

```

/* 取得Animation 定义在res/anim 目录下*/
myAnimation = AnimationUtils.loadAnimation(this, R.anim.my_anim);
/* 将myspinner 运行Animation */
mySpinner.startAnimation(myAnimation);

```

⑨ MainActivity.java 完整代码

```

package zyf.Ex_Ctrl_8;

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

```


9



```

        arg0.setVisibility(View.VISIBLE);
    }

    public void onNothingSelected(AdapterView<?> arg0) {
        // TODO Auto-generated method stub
        myTextView.setText("NONE");
    }
});
/*下拉菜单弹出的内容选项 触屏事件处理*/
mySpinner.setOnTouchListener(new Spinner.OnTouchListener() {

    public boolean onTouch(View v, MotionEvent event) {
        // TODO Auto-generated method stub
        /* 将mySpinner 运行Animation */
        v.startAnimation(myAnimation);
        /* 将mySpinner 隐藏*/
        v.setVisibility(View.INVISIBLE);
        return false;
    }
});
/*下拉菜单弹出的内容选项 焦点改变事件处理*/
mySpinner.setOnFocusChangeListener(new Spinner.OnFocusChangeListener() {

    public void onFocusChange(View v, boolean hasFocus) {
        // TODO Auto-generated method stub
    }
});
}
}

```

⑩ 结果



动态添加／删除的 Spinner 菜单

ArrayList 与 Widget 的依赖性

① 创建新工程

② 修改 mian.xml 布局，添加 UI 元素



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

<LinearLayout
    android:id="@+id/widget92"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
    android:id="@+id/TextView_Show_yourChoice"
    android:layout_width="190px"
    android:layout_height="wrap_content"
    android:text="TextView"
    android:textSize="25sp"
    android:layout_gravity="center_horizontal"
>
</TextView>
<EditText
    android:id="@+id/EditView_Input"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="18sp"
>
</EditText>
<Button
    android:id="@+id/Button_ADD"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="添加"
>
</Button>
<Button
    android:id="@+id/Button_DEL"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
```

```

        android:text="删除"
    >
</Button>
<Spinner
    android:id="@+id/Spinner_Slecte"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
>
</Spinner>
</LinearLayout>

```

③ 定义创建一个 List<T>

```

/*定义*/
private String[] cities;
private List<String> cityList;

/*初始化字符串数组*/
cities = new String[] { "Android", "BlackBerry", "J2ME", "Symbian",
    "Broncho", "LinuxMobile", "Palm", "WindwosMobile" };
/*初始化List实例 ArrayList的对象*/
cityList = new ArrayList<String>();
/*遍历，把字符串数组添加到ArrayList中*/
for (int i = 0; i < cities.length; i++) {
    cityList.add(cities[i]);
}

```

④ 定义创建一个 ArrayAdapter<String>

```

/*定义*/
private ArrayAdapter<String> arrayAdapter;
/* 这里应该使用List<> ,如果使用String[] 则会出错 */
/*初始化下拉菜单的内容适配器*/
arrayAdapter = new ArrayAdapter<String>(this,
    android.R.layout.simple_spinner_item, cityList);

```

⑤ 设置 ArrayAdapter<String>在下拉菜单中的显示布局

```

/*设置下拉菜单显示内容的风格*/
arrayAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_
    item);

```

⑥ 给下拉菜单添加内容 Adapter 适配器

```

/*给下拉菜单对象添加内容适配器*/
city_Spinner.setAdapter(arrayAdapter);

```


⑦ 修改该 MainActivity.java，实现动态添加和删除

```
package zyf.Ex_Ctrl_9ME;
/*使用的包导入*/
import java.util.ArrayList;
import java.util.List;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

public class Ex_Ctrl_9ME extends Activity implements Button.OnClickListener,
    Spinner.OnItemSelectedListener {
    /** Called when the activity is first created. */
    /*声明程序使用的对象*/
    private TextView show_yourChoice_TextView;
    private EditText input_City_EditText;
    private Button Add_Button, Del_Button;
    private Spinner city_Spinner;
    private ArrayAdapter<String> arrayAdapter;
    private String[] cities;
    private String addString;
    private List<String> cityList;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /*设置主屏显示布局为main.xml*/
        setContentView(R.layout.main);
        /*findViewById() 获取XML中的UI元素*/
        show_yourChoice_TextView =
            (TextView) findViewById(R.id.TextView_Show_yourChoice);
        input_City_EditText = (EditText) findViewById(R.id.EditView_Input);
        Add_Button = (Button) findViewById(R.id.Button_ADD);
        Del_Button = (Button) findViewById(R.id.Button_DEL);
        city_Spinner = (Spinner) findViewById(R.id.Spinner_Slecte);
        /*初始化字符串数组*/
        cities = new String[] { "Android", "BlackBerry", "J2ME",
```

```

        "Symbian", "Broncho", "LinuxMobile",
        "Palm", "WindwosMobile" };

    /*初始化List实例 ArrayList的对象*/
    cityList = new ArrayList<String>();
    /*遍历, 把字符串数组添加到ArrayList中*/
    for (int i = 0; i < cities.length; i++) {
        cityList.add(cities[i]);
    }
    /* 这里应该使用List<> ,如果使用String[] 则会出错 */
    /*初始化下拉菜单的内容适配器*/
    arrayAdapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_spinner_item, cityList);
    /*设置下拉菜单显示内容的风格*/
    arrayAdapter

.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    /*给下拉菜单对象添加内容适配器*/
    city_Spinner.setAdapter(arrayAdapter);
    /*默认启动时 文本标题显示*/
    show_yourChoice_TextView.setText(arrayAdapter.getItem(0));
    /*默认启动时 下拉菜单第一项被选中*/
    city_Spinner.setSelection(0);
    /*为按钮添加点击事件监听器*/
    Add_Button.setOnClickListener(this);
    Del_Button.setOnClickListener(this);
    /*为下拉菜单添加选项选中事件监听器*/
    city_Spinner.setOnItemSelectedListener(this);
}

@Override
public void onClick(View v) {
    // TODO Auto-generated method stub
    switch (v.getId()) {
        /*区别按钮来进行不同动作*/
        case R.id.Button_ADD:
            /*显示Toast提示*/
            Toast.makeText(this, "添加", Toast.LENGTH_SHORT).show();
            /*获得输入框中的将要添加的字符串*/
            addString = input_City_EditText.getText().toString();
            /*遍历, 比较是否和下拉菜单中内容重复*/
            for (int i = 0; i < arrayAdapter.getCount(); i++) {
                if (addString.equals(arrayAdapter.getItem(i))) {
                    return;
                }
                /*重复了, 则跳出*/
            }
        }
    }
}

```



```

    }
    /*如果添加字符串不为""*/
    if (!addString.equals("")) {
        /*添加进适配器中*/
        arrayAdapter.add(addString);
        /*获取刚刚添加进的字符串位置*/
        int post = arrayAdapter.getPosition(addString);
        /*设置刚刚添加进的下拉菜单内容被选中*/
        city_Spinner.setSelection(post);
        /*清空输入框*/
        input_City_EditText.setText("");
    }
    break;
case R.id.Button_DEL:

    if (city_Spinner.getSelectedItem() != null) {

        /* 删除mySpinner 的值 */
        arrayAdapter.remove(city_Spinner.getSelectedItem().toString());

        /* 将myEditText 清空 */
        input_City_EditText.setText("");
        if (arrayAdapter.getCount() == 0) {
            /* 将myTextView 清空 */
            Toast.makeText(this, "删完了", Toast.LENGTH_SHORT).show();
            show_yourChoice_TextView.setText("");
        }
    }
    break;
default:
    break;
}
}
/*下拉菜单选项被选中事件处理*/
@Override
public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
    long arg3) {
    // TODO Auto-generated method stub
    show_yourChoice_TextView.setText(arrayAdapter.getItem(arg2));
}
/*未被选中 事件处理*/
@Override
public void onNothingSelected(AdapterView<?> arg0) {
    // TODO Auto-generated method stub

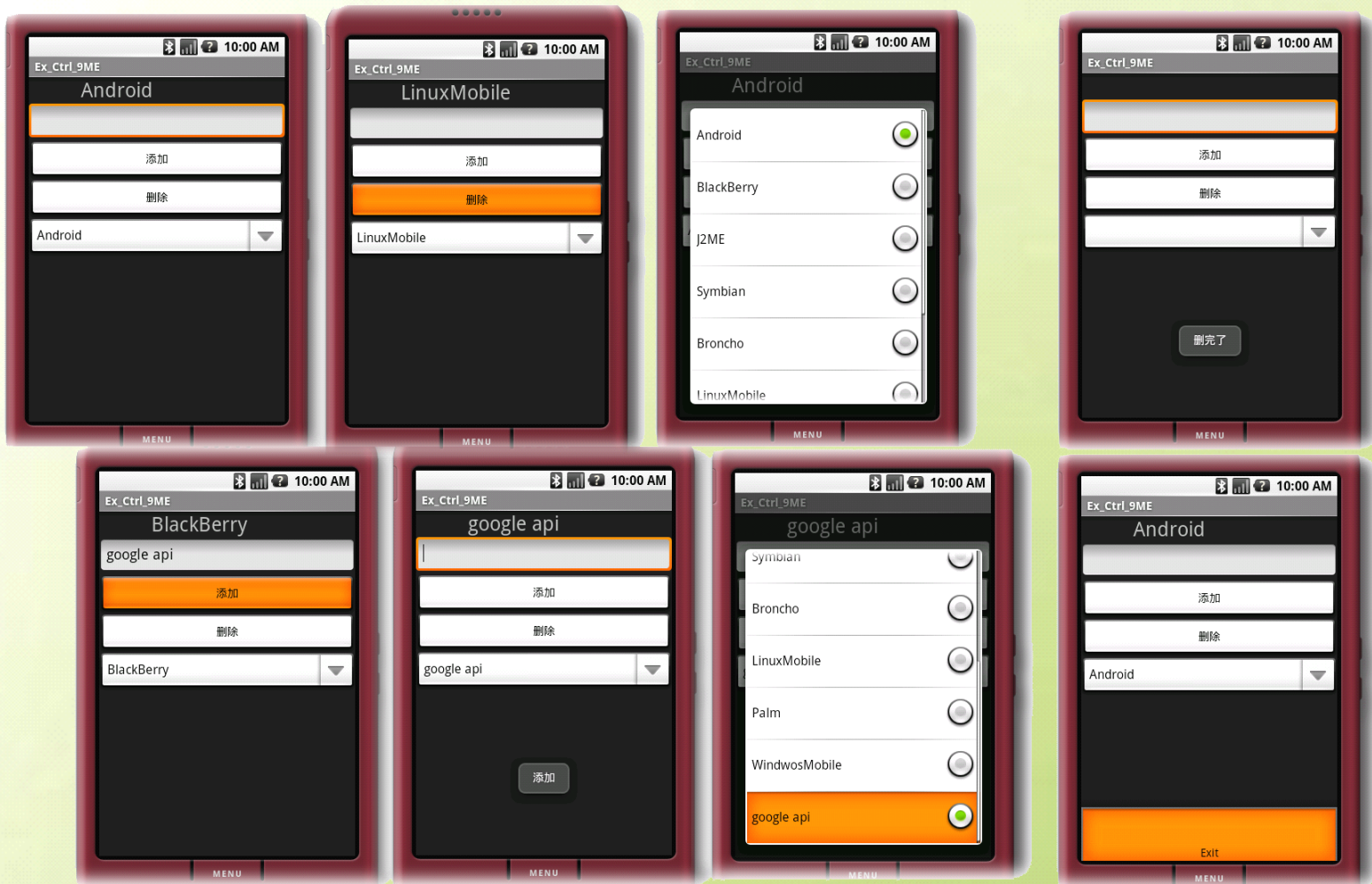
```

```

}
/*添加Menu菜单 进行退出操作*/
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // TODO Auto-generated method stub
    menu.add("Exit");
    return super.onCreateOptionsMenu(menu);
}
/*Menu菜单 退出*/
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // TODO Auto-generated method stub
    finish();
    return super.onOptionsItemSelected(item);
}
}

```

⑧ 结果




相簿浏览 Gallery

Gallery 与衍生 BaseAdapter 容器

① 新建项目

② 定义 layout 外部 resource 的 xml 文件，用来改变 layout 的背景



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="Gallery">
        <attr name="android:galleryItemBackground" />
    </declare-styleable>
    <!-- 定义layout 外部resource 的xml 文件，用来改变layout 的背景图。 -->
</resources>
```

③ 修改 main.xml 布局，添加一个 Gallery 和一个 ImageView

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/widget_absolutelayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
    <Gallery
        android:layout_width="fill_parent"
        android:layout_height="143px"
        android:layout_x="0px"
        android:layout_y="51px"
        android:id="@+id/Gallery_preView">
    </Gallery>
    <ImageView
        android:layout_width="239px"
        android:layout_height="218px"
        android:layout_x="38px"
        android:layout_y="184px"
        android:id="@+id/ImageView_photo">
    </ImageView>
</AbsoluteLayout>
```

④ 新建一个 myImageAdapter 类--Gallery 的适配器，它继承于 BaseAdapter 类。

```
package zyf.Ex_Ctrl_10ME;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
public class myImageAdapter extends BaseAdapter {
    @Override
    public int getCount() {
        // TODO Auto-generated method stub
        return 0;
    }
    @Override
    public Object getItem(int position) {
        // TODO Auto-generated method stub
        return null;
    }
    @Override
    public long getItemId(int position) {
        // TODO Auto-generated method stub
        return 0;
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        // TODO Auto-generated method stub
        return null;
    }
}
```

⑤ 修改 MainActivity.java，添加 Gallery 相关操作

```
package zyf.Ex_Ctrl_10ME;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.Gallery;
import android.widget.ImageView;
import android.widget.Toast;

public class Ex_Ctrl_10ME extends Activity {
    /** Called when the activity is first created. */
    /*定义要使用的对象*/
    private Gallery gallery;
    private ImageView imageview;
```




```
private myImageAdapter imageadapter;
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    imageadapter=new myImageAdapter(this);
    /* 通过findViewById 取得 资源对象*/
    gallery=(Gallery) findViewById(R.id.Gallery_preView);
    imageview=(ImageView) findViewById(R.id.ImageView_photo);
    /*给Gallery设置适配器 把Ex_Ctrl_10ME类传入参数*/
    gallery.setAdapter(imageadapter);
    /*设置Gallery的点击事件监听器*/
    gallery.setOnItemClickListener(new Gallery.OnItemClickListener(){
        @Override
        public void onItemClick(AdapterView<?> parent, View v, int position,
            long id) {
            // TODO Auto-generated method stub
            /*显示该图片是几号*/
            Toast.makeText(Ex_Ctrl_10ME.this,
                "这是图片: "+position+"号", Toast.LENGTH_SHORT).show();

            /*设置大图片*/
            imageview.setBackgroundResource(imageadapter.myImageIds[position]);
        }
    });
}
```

⑥ 修改 myImageAdapter.java 文件，实现相簿浏览效果

```
package zyf.Ex_Ctrl_10ME;

import android.content.Context;
import android.content.res.TypedArray;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageView;

public class myImageAdapter extends BaseAdapter{//自定义的类变量
    /*变量声明*/
    int mGalleryItemBackground;
    private Context context;//上下文
    /* 构建一Integer array 并取得预加载Drawable 的图片id */
```

```

public Integer[] myImageIds = { R.drawable.photo1, R.drawable.photo2,
    R.drawable.photo3, R.drawable.photo4, R.drawable.photo5,
    R.drawable.photo6, };

/*自定义的构造方法*/
public myImageAdapter(Context context) {
    // TODO Auto-generated constructor stub
    this.context=context;
    /*
     * 使用在res/values/attrs.xml 中的<declare-styleable>定义 的Gallery 属性.
     */
    TypedArray
typed_array=context.obtainStyledAttributes(R.styleable.Gallery);
    /* 取得Gallery 属性的Index id */

    mGalleryItemBackground=typed_array.getResourceId(R.styleable.Gallery_andro
id_galleryItemBackground, 0);
    /* 让对象的styleable 属性能够反复使用 */
    typed_array.recycle();
}
/* 重写的方法getCount,返回图片数目 */
@Override
public int getCount() {
    // TODO Auto-generated method stub
    return myImageIds.length;
}
/* 重写的方法getItemId,返回图像的数组id */
@Override
public Object getItem(int position) {
    // TODO Auto-generated method stub
    return position;
}
@Override
public long getItemId(int position) {
    // TODO Auto-generated method stub
    return position;
}
/* 重写的方法getView,返回一View 对象 */
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    // TODO Auto-generated method stub
    /* 产生ImageView 对象 */
    ImageView imageview = new ImageView(context);
    /* 设置图片给imageview 对象 */
    imageview.setImageResource(myImageIds[position]);

```



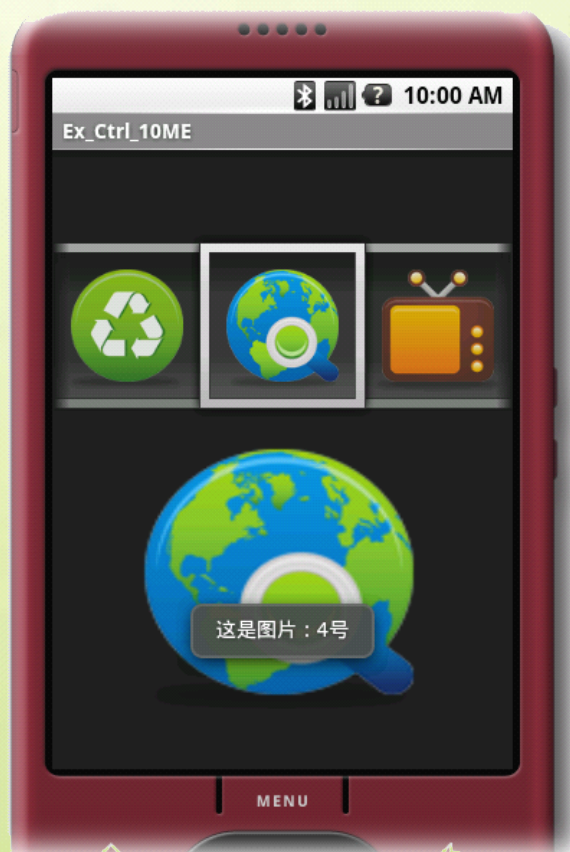
```

/* 重新设置图片的宽高 */
imageView.setScaleType(ImageView.ScaleType.FIT_XY);
/* 重新设置Layout 的宽高 */
imageView.setLayoutParams(new Gallery.LayoutParams(128, 128));
/* 设置Gallery 背景图 */
imageView.setBackgroundResource(mGalleryItemBackground);
/* 返回imageView 对象 */
return imageView;
}
}

```



⑦ 结果




文件搜索引擎 FileSearch

SDCard 中文件搜索与 File 类

① 创建新工程

② 在 string.xml 添加程序中要使用的字符串



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">myFileTest</string>
    <string name="showInput">输入关键字</string>
    <string name="toSearch">搜索</string>
    <string name="info">系统SDCard目录文件路径:\n</string>
    <string name="pleaseInput">请输入关键字!!</string>
    <string name="notFond">没有找到相关文件!!</string>
    <string name="pathError">读取路径出错!!</string>
</resources>
```

③ 修改 main.xml 布局, 添加两个 TextView、一个 EditText、一个 Button

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/widget0"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
<Button
    android:id="@+id/Button_Search"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="253px"
    android:layout_y="5px"
    android:text="@string/toSearch">
</Button>
<EditText
    android:id="@+id/input_KEY_EditText"
    android:layout_width="112px"
    android:layout_height="52px"
    android:textSize="18sp"
    android:layout_x="119px"
    android:layout_y="4px"
>
</EditText>
```



```

<TextView
    android:id="@+id/TextView_showIn"
    android:layout_width="103px"
    android:layout_height="29px"
    android:textSize="20sp"
    android:layout_x="5px"
    android:layout_y="16px"
    android:text="@string/showInput">
</TextView>
<TextView
    android:layout_width="fill_parent"
    android:layout_height="370px"
    android:layout_x="0px"
    android:layout_y="60px"
    android:id="@+id/TextView_Result">
</TextView>
</AbsoluteLayout>

```

④ 修改 MainActivity.java 文件，添加搜索功能

```

package zyf.myFileTest;
/*导入程序使用的包*/
import java.io.File;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class myFileTest extends Activity implements Button.OnClickListener {
    /** Called when the activity is first created. */
    /*定义程序要使用的类对象*/
    private File file;
    private String path;
    private String info;
    private String theKey_formInput;
    private TextView show_Result;
    private EditText input_SearchKey_Edit;
    private Button toSearch_Button;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}

```

```
/*设置主屏布局为main.xml*/
setContentView(R.layout.main);
/*通过findViewById() 获取XML中的UI对象*/
show_Result = (TextView) findViewById(R.id.TextView_Result);
input_SearchKey_Edit = (EditText) findViewById(R.id.input_KEY_EditText);
toSearch_Button = (Button) findViewById(R.id.Button_Search);
/*为搜索按钮添加点击事件监听器*/
toSearch_Button.setOnClickListener(this);
/*初始化一个Field 对象, 指定路径为/sdcard*/
file = new File("/sdcard");
/*从xml中获取字符串*/
info = getString(R.string.info);
}
/*按钮点击事件处理*/
public void onClick(View v) {
    // TODO Auto-generated method stub
    /*清空*/
    path = "";
    show_Result.setText("");
    /*取得输入框中的要查询的Key*/
    theKey_formInput = input_SearchKey_Edit.getText().toString();
    /*浏览文件*/
    BrowserFile(file);
}
/*浏览文件方法*/
public void BrowserFile(File file) {
    if (theKey_formInput.equals("")) {
        /*如果输入框没有输入点击搜索按钮, 提示输入*/
        Toast.makeText(this, getString(R.string.pleaseInput),
            Toast.LENGTH_SHORT).show();
    } else {
        /*开始搜索文件*/
        ToSearchFiles(file);
        /*搜索完毕后, 如果搜到结果为空, 提示没有找到*/
        if (show_Result.getText().equals("")) {
            Toast.makeText(this, getString(R.string.notFond),
                Toast.LENGTH_SHORT).show();
        }
    }
}
/*开始搜索文件方法*/
public void ToSearchFiles(File file) {
    /*定义一个File文件数组, 用来存放/sdcard目录下的文件或文件夹*/
    File[] the_Files = file.listFiles();
```



```

/*通过遍历所有文件和文件夹*/
for (File tempF : the_Files) {
    if (tempF.isDirectory()) {
        ToSearchFiles(tempF);
        /*如果是文件夹的话继续遍历搜索*/
    } else {
        try {
            /*是文件，进行比较，如果文件名称中包含输入搜索Key，则返回大于-1的值*/
            if (tempF.getName().indexOf(theKey_formInput) > -1) {
                /*获取符合条件文件的路径，进行累加*/
                path += "\n" + tempF.getPath();
                /*显示结果的TextView显示信息和搜索到的路径*/
                show_Result.setText(info + path);
            }
        } catch (Exception e) {
            // TODO: handle exception
            /*如果路径找不到，提示出错*/
            Toast.makeText(this, getString(R.string.pathError),
                Toast.LENGTH_SHORT).show();
        }
    }
}
}
}
}

```

⑤ 结果



按钮点击变换

ImageButton 选择特效

① 新建工程

② 修改 main.xml 布局, 添加一个 TextView、两个 ImageButton



```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/widget0"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
    android:id="@+id/showinform_TextView"
    android:layout_width="259px"
    android:layout_height="80px"
    android:text="没有按钮事件"
    android:textSize="20px"
    android:layout_x="32px"
    android:layout_y="33px"
>
</TextView>
<ImageButton
    android:id="@+id/ImageButton_A"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="129px"
    android:layout_y="140px"
    android:background="@drawable/ipod">
</ImageButton>
<ImageButton
    android:id="@+id/ImageButton_B"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="129px"
    android:layout_y="226px"
    android:background="@drawable/installer">
</ImageButton>
</AbsoluteLayout>
```


③ 修改 MainActivity.java 文件，实现选择效果

```
package zyf.Ex_Ctrl_12ME;

/*导入使用的包*/
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageButton;
import android.widget.TextView;

public class Ex_Ctrl_12ME extends Activity implements
    ImageButton.OnClickListener {
    /** Called when the activity is first created. */
    /* 定义使用的类对象 */
    private TextView showyourClick;
    private ImageButton button_A, button_B;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /* 载入主屏布局main.xml */
        setContentView(R.layout.main);
        /* 从XML中获取UI元素对象 */
        showyourClick = (TextView) findViewById(R.id.showinform_TextView);
        button_A = (ImageButton) findViewById(R.id.ImageButton_A);
        button_B = (ImageButton) findViewById(R.id.ImageButton_B);
        /* 给两个ImageButton添加点击事件监听器 */
        button_A.setOnClickListener(this);
        button_B.setOnClickListener(this);
    }

    /* button点击事件处理 */
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        switch (v.getId()) {
            case R.id.ImageButton_A: {
                /*button_A按钮被点击*/
                /*显示信息*/
                showyourClick.setText("你点击了: \nbutton_A");
                /*切换背景图片*/
                button_A.setBackgroundResource(R.drawable.android);
                button_B.setBackgroundResource(R.drawable.installer);
            }
        }
    }
}
```

```

        break;
    case R.id.ImageButton_B: {
        /*button_B按钮被点击*/
        /*显示信息*/
        showyourClick.setText("你点击了: \nbutton_B");
        /*切换背景图片*/
        button_B.setBackgroundResource(R.drawable.android);
        button_A.setBackgroundResource(R.drawable.ipod);
    }
        break;
    default:
        break;
    }
}

```

④ 结果



自动完成输入框自动提示功能的菜单

AutoCompleteTextView 与数组

① 新建工程

② 修改 main.xml 布局, 添加一个 TextView、一个 AutoCompleteTextView、一个 Button



```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/widget0"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
    android:id="@+id/TextView_InputShow"
    android:layout_width="228px"
    android:layout_height="47px"
    android:text="请输入 "
    android:textSize="25px"
    android:layout_x="42px"
    android:layout_y="37px"
>
</TextView>
<AutoCompleteTextView
    android:id="@+id/AutoCompleteTextView_input"
    android:layout_width="275px"
    android:layout_height="wrap_content"
    android:text=""
    android:textSize="18sp"
    android:layout_x="23px"
    android:layout_y="98px"
>
</AutoCompleteTextView>
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="127dip"
    android:text="清空"
    android:id="@+id/Button_clean"
    android:layout_y="150dip">
</Button>
</AbsoluteLayout>
```

③ 修改 MainActivity.java, 添加自动完成功能

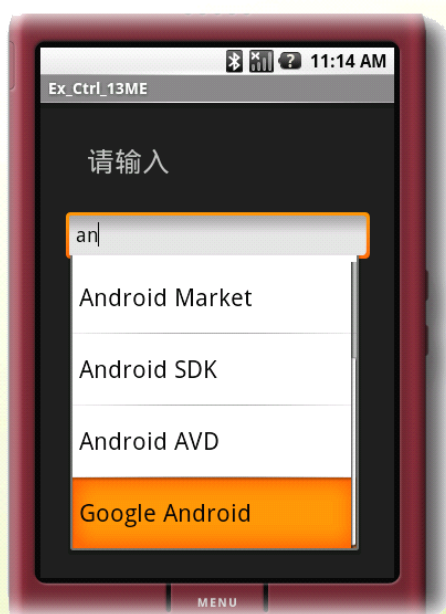
```
package zyf.Ex_Ctrl_13ME;
/*导入使用的包*/
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.TextView;
public class Ex_Ctrl_13ME extends Activity {
    /** Called when the activity is first created. */
    /*定义要使用的类对象*/
    private String[] normalString =
        new String[] {
            "Android", "Android Blog","Android Market", "Android SDK",
            "Android AVD","BlackBerry","BlackBerry JDE", "Symbian",
            "Symbian Carbide", "Java 2ME","Java FX", "Java 2EE",
            "Java 2SE", "Mobile", "Motorola", "Nokia", "Sun",
            "Nokia Symbian", "Nokia forum", "WindowsMobile", "Broncho",
            "Windows XP", "Google", "Google Android ", "Google 浏览器",
            "IBM", "MicroSoft", "Java", "C++", "C", "C#", "J#", "VB" };
    @SuppressWarnings("unused")
    private TextView show;
    private autoCompleteTextView autoTextView;
    private Button clean;
    private ArrayAdapter<String> arrayAdapter;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /*装入主屏布局main.xml*/
        setContentView(R.layout.main);
        /*从XML中获取UI元素对象*/
        show = (TextView) findViewById(R.id.TextView_InputShow);
        autoTextView =
            (AutoCompleteTextView) findViewById(R.id.AutoCompleteTextView_input);
        clean = (Button) findViewById(R.id.Button_clean);
        /*实现一个适配器对象, 用来给自动完成输入框添加自动装入的内容*/
        arrayAdapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_dropdown_item_1line, normalString);
        /*给自动完成输入框添加内容适配器*/
        autoTextView.setAdapter(arrayAdapter);
        /*给清空按钮添加点击事件处理监听器*/
```



```
clean.setOnClickListener(new Button.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        // TODO Auto-generated method stub  
        /*清空*/  
        autoTextView.setText("");  
    }  
});  
}
```



④ 结果



MultiAutoCompleteTextView 与数组

① 新建工程

② 修改 main.xml 布局，添加一个 TextView、一个 MultiAutoCompleteTextView、一个 Button



```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/widget0"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
<TextView
    android:id="@+id/TextView_InputShow"
    android:layout_width="228px"
    android:layout_height="47px"
    android:text="请输入 "
    android:textSize="25px"
    android:layout_x="42px"
    android:layout_y="37px"/>
<MultiAutoCompleteTextView
    android:layout_width="275px"
    android:layout_height="wrap_content"
    android:text=""
    android:id="@+id/MultiAutoCompleteTextView"
    android:layout_x="23px"
    android:layout_y="98px"/>
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="127dip"
    android:text="清空"
    android:id="@+id/Button_clean"
    android:layout_y="180dip"/>
</AbsoluteLayout>
```

③ MultiAutoCompleteTextView 是可以自动完成用户输入，能够添加多个输入，注意设置 Tokenizer

/*设置Tokenizer来确定用户输入文本的相关范围*/

```
myAutoCompleteTextView
    .setTokenizer(new MultiAutoCompleteTextView.CommaTokenizer());
```


④ 修改 MainActivity.java, 实现动态自动完成用户输入相关内容

```
package zyf.Ex_Ctrl_13_B;
/*导入使用的包*/
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.MultiAutoCompleteTextView;
public class Ex_Ctrl_13_B extends Activity {

    /** Called when the activity is first created. */
    /*定义要使用的类对象*/
    private String[] normalString =
        new String[] {
            "Android", "Android Blog", "Android Market", "Android SDK",
            "Android AVD", "BlackBerry", "BlackBerry JDE", "Symbian",
            "Symbian Carbide", "Java 2ME", "Java FX", "Java 2EE",
            "Java 2SE", "Mobile", "Motorola", "Nokia", "Sun",
            "Nokia Symbian", "Nokia forum", "WindowsMobile", "Broncho",
            "Windows XP", "Google", "Google Android ", "Google 浏览器",
            "IBM", "MicroSoft", "Java", "C++", "C", "C#", "J#", "VB" };

    private Button clean;
    private MultiAutoCompleteTextView myAutoCompleteTextView;
    private ArrayAdapter<String> adapter;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /* 装入主屏布局main.xml */
        setContentView(R.layout.main);
        /* 以findViewById() 从XML中获取UI元素对象 */
        myAutoCompleteTextView =
            (MultiAutoCompleteTextView) findViewById(R.id.MultiAutoCompleteTextView);
        clean = (Button) findViewById(R.id.Button_clean);
        /* new ArrayAdapter 对象并将normalString 字符串数组传入 */
        /* 实现一个适配器对象, 用来给自动完成输入框添加自动装入的内容 */
        adapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_dropdown_item_1line, normalString);
        /* 将ArrayAdapter 添加AutoCompleteTextView 对象中 */
        /* 给自动完成输入框添加内容适配器 */
        myAutoCompleteTextView.setAdapter(adapter);

        /*设置Tokenizer来确定用户输入文本的相关范围*/
        myAutoCompleteTextView
```

```

        .setTokenizer(new MultiAutoCompleteTextView.CommaTokenizer());
/* 给清空按钮添加点击事件处理监听器 */
clean.setOnClickListener(new Button.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        /* 清空 */
        myAutoCompleteTextView.setText("");
    }
});
}
}

```



⑤ 结果



模拟/数字/线程小时钟设计

AnalogClock 与 DigitalClock 的原理--线程时钟实现

① 新建工程

② 修改 man.xml 布局，添加一个 AnalogClock、一个 DigitalClock、一个 TextView



```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/widget0"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <TextView
        android:id="@+id/TextView_showTime"
        android:layout_width="200px"
        android:layout_height="39px"
        android:textSize="25px"
        android:layout_x="82px"
        android:layout_y="222px">
    </TextView>
    <AnalogClock
        android:id="@+id/Clock"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="82px"
        android:layout_y="34px">
    </AnalogClock>
    <DigitalClock
        android:id="@+id/DigitalClock01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="上午1:01"
        android:textSize="25px"
        android:layout_x="82px"
        android:layout_y="300px">
    </DigitalClock>
    <TextView
        android:id="@+id/widget46"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="模拟时钟"
        android:layout_x="11px"
```

```

        android:layout_y="46px">
</TextView>
<TextView
    android:id="@+id/widget47"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="线程时钟"
    android:layout_x="11px"
    android:layout_y="228px">
</TextView>
<TextView
    android:id="@+id/widget48"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="数字时钟"
    android:layout_x="11px"
    android:layout_y="308px">
</TextView>
</AbsoluteLayout>

```

- ③ 模拟时钟的实现不需要额外代码，只需要 UI 中添加，其自动显示时间

```

/*定义模拟时钟对象*/
AnalogClock myClock;
/*从XML中获取模拟时钟UI对象*/
myClock=(AnalogClock) findViewById(R.id.Clock);

```

- ④ 数字时钟的实现也不需要额外代码，只需要 UI 中添加，其自动显示时间

```

/*定义数字时钟对象*/
DigitalClock myDigClock;
/*从XML中获取数字时钟UI对象*/
myDigClock=(DigitalClock) findViewById(R.id.DigitalClock01);

```

- ⑤ 而使用线程实现的 TextView 时钟则需要线程 Thread、Handler（发送、处理消息）辅助实现

```

import android.os.Handler;
import android.os.Message;
public class Clock extends Activity implements Runnable{
    public Handler myHandler;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.gh);
        myHandler = new Handler() {
            @Override

```



```

        public void handleMessage(Message msg) {
            // TODO Auto-generated method stub
        }
    };
    Thread myT = new Thread(this);
    myT.start();
}
@Override
public void run() {
    // TODO Auto-generated method stub
}
}

```

⑥ 修改 `mianActivity.java`, 实现线程控制, 以及模拟/数字时钟的实现

```

package zyf.three.clock;

/*导入要使用的包*/
import java.util.Calendar;
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.widget.AnalogClock;
import android.widget.DigitalClock;
import android.widget.TextView;

public class Clock extends Activity implements Runnable {
    /* 定义要使用的类对象 */
    private TextView showTime;// 显示进程时钟的TextView
    AnalogClock myClock;// 模拟时钟
    DigitalClock myDigClock;// 数字时钟
    private final int msg_Key = 0x1234;// 发送的消息内容
    public Handler myHandler;// 发送、处理消息的类
    public Calendar myCalendar;// 日历类
    private int my_Hour, my_Minute, my_Second;// 时、分、秒
    private Thread myT;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        /* 导入主屏布局main.xml */
        setContentView(R.layout.gh);
        /* 从XML中获取模拟时钟UI对象 */
    }
}

```

```

myClock = (AnalogClock) findViewById(R.id.Clock);
/* 从XML中获取数字时钟UI对象 */
myDigClock = (DigitalClock) findViewById(R.id.DigitalClock01);
/* 从XML中获取TextView UI对象 */
showTime = (TextView) findViewById(R.id.TextView_showTime);
/* 通过Handler 来接收进程所传递的信息并更新TextView */
myHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        /* 这里是处理信息的方法 */
        // TODO Auto-generated method stub
        super.handleMessage(msg);
        switch (msg.what) {
            case msg_Key:
                /* 在这处理要TextView对象Show时间的事件 */
                showTime.setText(my_Hour + " : " + my_Minute + " : "
                    + my_Second);
                break;

            default:
                break;
        }
    }
};
/* 通过进程来持续取得系统时间 */
myT = new Thread(this);
myT.start();
}

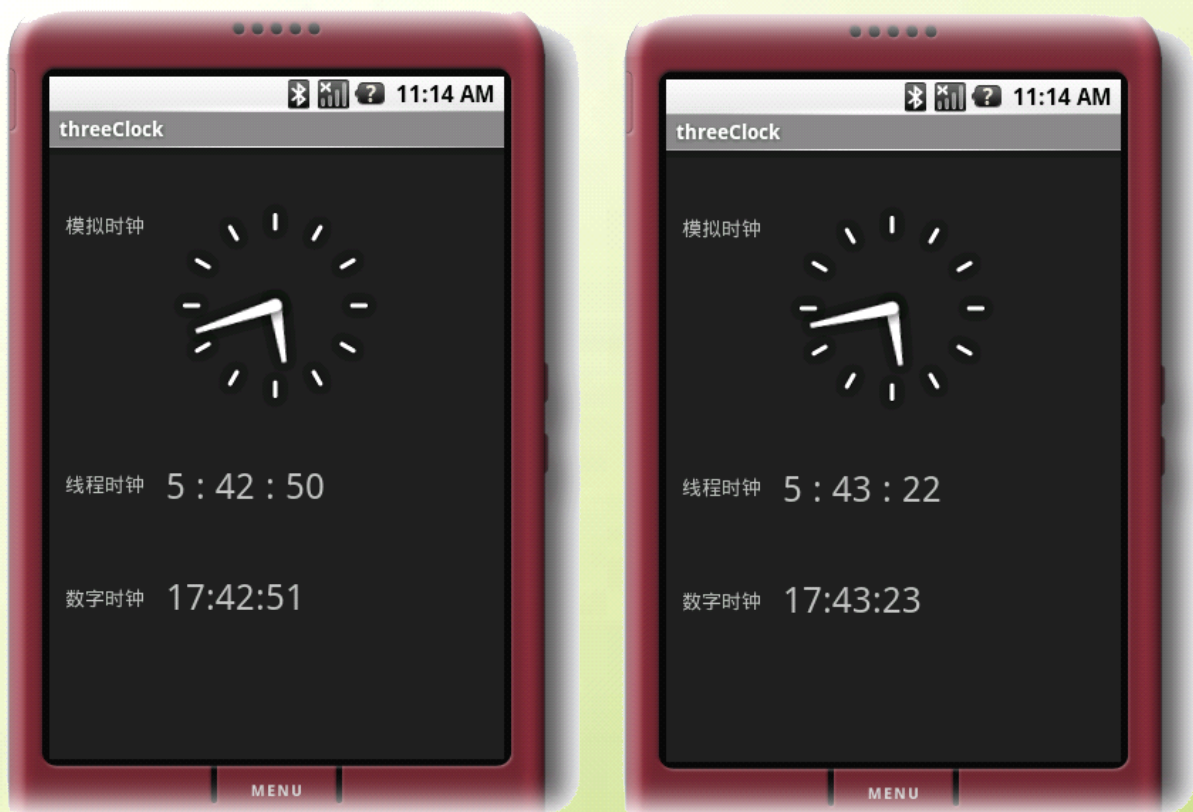
/* 实现一个Runnable接口，实例化一个进程对象， 用来持续取得系统时间 */
@Override
public void run() {
    // TODO Auto-generated method stub
    try {
        do {
            /* 取得系统时间 */
            long Time = System.currentTimeMillis();
            myCalendar = Calendar.getInstance();
            myCalendar.setTimeInMillis(Time);
            my_Hour = myCalendar.get(Calendar.HOUR);
            my_Minute = myCalendar.get(Calendar.MINUTE);
            my_Second = myCalendar.get(Calendar.SECOND);
            /* 让进程休息一秒 */
            Thread.sleep(1000);
        } while (true);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

```



```
        /* 重要关键程序:取得时间后发出信息给Handler */
        Message msg = new Message();
        msg.what = msg_Key;
        myHandler.sendMessage(msg);
        /* 重要关键程序:取得时间后发出信息给Handler */
    } while (myT.interrupted() == false);
    /* 当系统发出中断信息时停止本循环 */
} catch (InterruptedException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}
```

⑦ 结果



动态输入日期与时间

DatePicker 与 TimePicker 应用

① 新建工程

② 修改 main.xml 布局，添加一个 DatePicker、一个 TimePicker、一个 TextView



```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/widget0"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <DatePicker
        android:id="@+id/my_DatePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="10px"
        android:layout_y="10px">
    </DatePicker><!-- 日期设置器 -->
    <TimePicker
        android:id="@+id/my_TimePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="10px"
        android:layout_y="150px">
    </TimePicker><!-- 事件设置器 -->
    <TextView
        android:id="@+id/my_TextView"
        android:layout_width="228px"
        android:layout_height="29px"
        android:text="TextView"
        android:layout_x="10px"
        android:layout_y="300px">
    </TextView>
</AbsoluteLayout>
```

③ DatePicker 的初始化与日期改变事件的处理

```
/* 定义 程序用到的UI元素对象:日历设置器*/
DatePicker my_datePicker;

/* findViewById() 从XML中获取UI元素对象 */
my_datePicker = (DatePicker) findViewById(R.id.my_DatePicker);
```



```

/*为日历设置器添加点击事件监听器，处理设置日期事件*/
my_datePicker.init(my_Year, my_Month, my_Day,
    new DatePicker.OnDateChangeListener() {
        @Override
        public void onChanged(DatePicker view, int year,
            int monthOfYear, int dayOfMonth) {
            // TODO Auto-generated method stub
            /*日期改变事件处理*/
        }
    });

```

④ TimePicker 的初始化与时间改变事件的处理

```

/* 定义 程序用到的UI元素对象:时间设置器*/
TimePicker my_timePicker;
/* findViewById() 从XML中获取UI元素对象 */
my_timePicker = (TimePicker) findViewById(R.id.my_TimePicker);
/* 把时间设置成24小时制 */
my_timePicker.setIs24HourView(true);
/*为时间设置器添加点击事件监听器，处理设置时间事件*/
my_timePicker.setOnTimeChangeListener(new
    TimePicker.OnTimeChangeListener() {
        @Override
        public void onTimeChanged(TimePicker view, int hourOfDay,
            int minute) {
            // TODO Auto-generated method stub
            /*时间改变事件处理*/
        }
    });

```

⑤ 修改 MainActivity.java，添加动态修改时间并显示效果

```

package zyf.Ex_Ctrl_15ME;
/*导入要使用的包*/
import java.util.Calendar;
import java.util.Locale;
import android.app.Activity;
import android.os.Bundle;
import android.widget.DatePicker;
import android.widget.TextView;
import android.widget.TimePicker;
public class Ex_Ctrl_15ME extends Activity {
    /** Called when the activity is first created. */
    /* 定义时间变量：年、月、日、小时、分钟 */
    int my_Year;
    int my_Month;

```

```

int my_Day;
int my_Hour;
int my_Minute;
/* 定义 程序用到的UI元素对象:日历设置器、时间设置器、显示时间的TextView */
DatePicker my_datePicker;
TimePicker my_timePicker;
TextView showDate_Time;
/* 定义日历对象, 初始化时, 用来获取当前时间 */
Calendar my_Calendar;
@Override
public void onCreate(Bundle savedInstanceState) {
    /* 从Calendar抽象基类获得实例对象, 并设置成中国时区 */
    my_Calendar = Calendar.getInstance(Locale.CHINA);
    /* 从日历对象中获取当前的: 年、月、日、时、分 */
    my_Year = my_Calendar.get(Calendar.YEAR);
    my_Month = my_Calendar.get(Calendar.MONTH);
    my_Day = my_Calendar.get(Calendar.DAY_OF_MONTH);
    my_Hour = my_Calendar.get(Calendar.HOUR_OF_DAY);
    my_Minute = my_Calendar.get(Calendar.MINUTE);
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    /* findViewById() 从XML中获取UI元素对象 */
    my_datePicker = (DatePicker) findViewById(R.id.my_DatePicker);
    my_timePicker = (TimePicker) findViewById(R.id.my_TimePicker);
    showDate_Time = (TextView) findViewById(R.id.my_TextView);
    /* 把时间设置成24小时制 */
    my_timePicker.setIs24HourView(true);
    /* 显示时间 */
    loadDate_Time();
    /*为日历设置器添加点击事件监听器, 处理设置日期事件*/
    my_datePicker.init(my_Year, my_Month, my_Day,
        new DatePicker.OnDateChangedListener() {
        @Override
        public void onChanged(DatePicker view, int year,
            int monthOfYear, int dayOfMonth) {
            // TODO Auto-generated method stub
            /*把设置改动后的日期赋值给我的日期对象*/
            my_Year=year;
            my_Month=monthOfYear;
            my_Day=dayOfMonth;
            /* 动态显示修改后的日期 */
            loadDate_Time();
        }
    });
}

```



```

/*为时间设置器添加点击事件监听器，处理设置时间事件*/
my_timePicker.setOnTimeChangeListener(new
                                           TimePicker.OnTimeChangeListener(){
        @Override
        public void onTimeChanged(TimePicker view, int hourOfDay,
                                           int minute) {

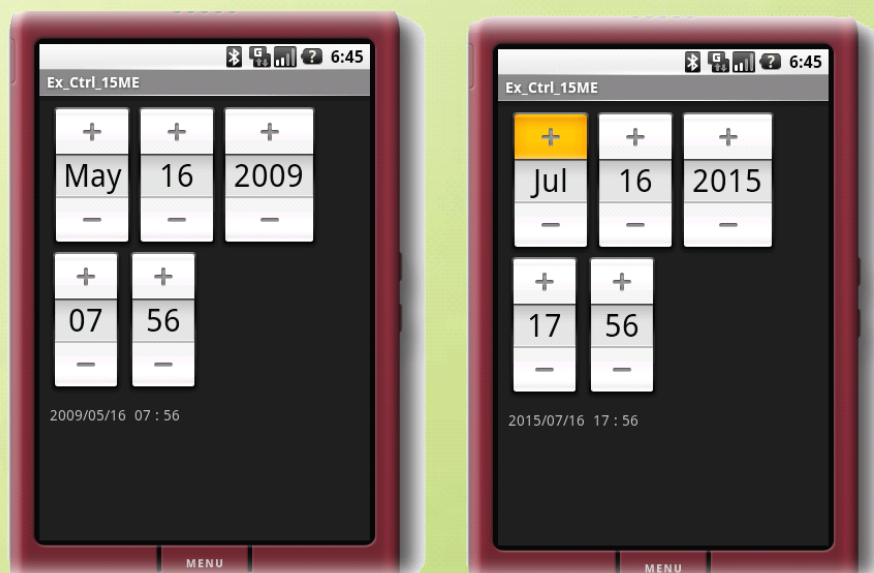
            /*把设置改动后的时间赋值给我的时间对象*/
            my_Hour=hourOfDay;
            my_Minute=minute;
            /* 动态显示修改后的时间 */
            loadDate_Time();
        }
    });
}

/* 设置显示日期时间的方法 */
private void loadDate_Time() {
    showDate_Time.setText(new StringBuffer()
        .append(my_Year).append("/")
        .append(FormatString(my_Month + 1))
        .append("/") .append(FormatString(my_Day))
        .append(" ") .append(FormatString(my_Hour))
        .append(" : ") .append(FormatString(my_Minute)));
}

/* 日期时间显示两位数的方法 */
private String FormatString(int x) {
    String s = Integer.toString(x);
    if (s.length() == 1) {
        s = "0" + s;
    }
    return s;
}
}

```

⑥ 结果



DatePickerDialog 与 TimePickerDialog 应用

① 新建工程

② 修改 main.xml 布局，添加两个按钮、一个 TextView



```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/widget0"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <Button
        android:id="@+id/show_DatePicker"
        android:layout_width="150px"
        android:layout_height="wrap_content"
        android:layout_x="10px"
        android:layout_y="10px" android:text="显示日期设置对话框"/>
    <Button
        android:id="@+id/show_TimePicker"
        android:layout_width="150px"
        android:layout_height="wrap_content"
        android:layout_x="10px"
        android:layout_y="70px" android:text="显示时间设置对话框"/>
    <TextView
        android:id="@+id/my_TextView"
        android:layout_width="228px"
        android:text="TextView"
        android:layout_x="10px"
        android:layout_y="180px"
        android:layout_height="45px"
        android:textSize="20px"/>
</AbsoluteLayout>
```

③ DatePickerDialog 的定义与初始化以及显示

```
/* 定义 程序用到的UI元素对象:日历设置器对话框 */
DatePickerDialog my_datePickerDialog;
/*构造一个DatePickerDialog对象, 第一个参数为Context、
 * 第二参数为日期修改事件处理监听器、后面为初始化的年月日*/
my_datePickerDialog=new DatePickerDialog (Ex_Ctrl_15_B.this,
                                           myDateSetListener, my_Year, my_Month, my_Day);
/*显示出日期设置对话框*/
my_datePickerDialog.show();
```


④ DatePickerDialog 的日期修改事件处理

```
/*日期改变设置事件监听器*/
private OnDateSetListener myDateSetListener=new OnDateSetListener(){
    @Override
    public void onDateSet(DatePicker view, int year, int monthOfYear,
        int dayOfMonth) {
        // TODO Auto-generated method stub
        /*日期改变设置事件处理*/
    }
};
```

⑤ TimePickerDialog 的定义与初始化以及显示

```
/* 定义 程序用到的UI元素对象:时间设置器对话框*/
TimePickerDialog my_timePickerDialog;
/*构造一个TimePickerDialog对象, 第一个参数为Context、
 *第二个参数为时间修改事件监听器、后面两个为初始化时间,
 *最后一个boolean类型设置是否为24小时制*/
my_timePickerDialog=new TimePickerDialog(Ex_Ctrl_15_B.this,
    myTimeSetListener, my_Hour, my_Minute, false);

/*显示出日期设置对话框*/
my_timePickerDialog.show();
```

⑥ TimePickerDialog 的时间修改事件处理

```
/*时间改变设置事件监听器*/
private OnTimeSetListener myTimeSetListener=new OnTimeSetListener(){
    @Override
    public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
        // TODO Auto-generated method stub
        /*时间改变设置事件处理*/
    }
};
```

⑦ 修改 MainActivity.java, 实现动态显示修改的日期、时间

```
package zyf.Ex_Ctrl_15_B;
/*导入使用的包*/
import java.util.Calendar;
import java.util.Locale;
import android.app.Activity;
import android.app.DatePickerDialog;
import android.app.TimePickerDialog;
import android.app.DatePickerDialog.OnDateSetListener;
import android.app.TimePickerDialog.OnTimeSetListener;
import android.os.Bundle;
import android.view.View;
```

```
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.TextView;
import android.widget.TimePicker;

public class Ex_Ctrl_15_B extends Activity implements Button.OnClickListener{
    /** Called when the activity is first created. */
    /* 定义时间变量：年、月、日、小时、分钟 */
    int my_Year;
    int my_Month;
    int my_Day;
    int my_Hour;
    int my_Minute;
    /* 定义 程序用到的UI元素对象：日历设置器对话框、时间设置器对话框、
       显示时间的TextView、按钮 */
    DatePickerDialog my_datePickerDialog;
    TimePickerDialog my_timePickerDialog;
    TextView showDate_Time;
    Button showDatePDIALOG;
    Button showTimePDIALOG;
    /* 定义日历对象，初始化时，用来获取当前时间 */
    Calendar my_Calendar;
    /*日期改变设置事件监听器*/
    private OnDateSetListener myDateSetListener=new OnDateSetListener() {
        @Override
        public void onDateSet(DatePicker view, int year,
                               int monthOfYear, int dayOfMonth) {
            // TODO Auto-generated method stub
            /*把设置修改后的日期赋值给我的年、月、日变量*/
            my_Year=year;
            my_Month=monthOfYear;
            my_Day=dayOfMonth;
            /*显示设置后的日期*/
            loadDate_Time();
        }
    };
    /*时间改变设置事件监听器*/
    private OnTimeSetListener myTimeSetListener=new OnTimeSetListener() {
        @Override
        public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
            // TODO Auto-generated method stub
            /*把设置修改后的时间赋值给我的时、分变量*/
            my_Hour=hourOfDay;
            my_Minute=minute;
        }
    };
}
```



```

        /*显示设置后的时间*/
        loadDate_Time();
    }
};

@Override
public void onCreate(Bundle savedInstanceState) {
    /* 从Calendar抽象基类获得实例对象，并设置成中国时区 */
    my_Calendar = Calendar.getInstance(Locale.CHINA);
    /* 从日历对象中获取当前的：年、月、日、时、分 */
    my_Year = my_Calendar.get(Calendar.YEAR);
    my_Month = my_Calendar.get(Calendar.MONTH);
    my_Day = my_Calendar.get(Calendar.DAY_OF_MONTH);
    my_Hour = my_Calendar.get(Calendar.HOUR_OF_DAY);
    my_Minute = my_Calendar.get(Calendar.MINUTE);
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    /* findViewById() 从XML中获取UI元素对象 */
    showDate_Time = (TextView) findViewById(R.id.my_TextView);
    showDatePDialog = (Button) findViewById(R.id.show_DatePicker);
    showTimePDialog = (Button) findViewById(R.id.show_TimePicker);
    /*显示当前时间*/
    loadDate_Time();
    /*为按钮添加点击事件监听器*/
    showDatePDialog.setOnClickListener(this);
    showTimePDialog.setOnClickListener(this);
}

/* 设置显示日期时间的方法 */
private void loadDate_Time() {
    // TODO Auto-generated method stub
    showDate_Time.setText(new StringBuffer()
        .append(my_Year).append("/")
        .append(FormatString(my_Month + 1))
        .append("/") .append(FormatString(my_Day))
        .append(" ") .append(FormatString(my_Hour))
        .append(" : ") .append(FormatString(my_Minute)));
}

/* 日期时间显示两位数的方法 */
private String FormatString(int x) {
    String s = Integer.toString(x);
    if (s.length() == 1) {
        s = "0" + s;
    }
    return s;
}

```

```

@Override
public void onClick(View v) {
    // TODO Auto-generated method stub
    switch (v.getId()) {
        case R.id.show_DatePicker:
            /*显示日期设置对话框的按钮点击事件处理*/{
                /*构造一个DatePickerDialog对象，第一个参数为Context、
                第二参数为日期修改事件处理监听器、后面为初始化的年月日*/
                my_datePickerDialog=new DatePickerDialog(Ex_Ctrl_15_B.this,
                    myDateSetListener, my_Year, my_Month, my_Day);
                /*显示出日期设置对话框*/
                my_datePickerDialog.show();
            }
            break;
        case R.id.show_TimePicker:
            /*显示时间设置对话框的按钮点击事件处理*/{
                /*构造一个TimePickerDialog对象，第一个参数为Context、第二个参数为时间修改
                事件监听器、后面两个为初始化时间，最后一个boolean类型设置是否为24小时制*/
                my_timePickerDialog=new TimePickerDialog(Ex_Ctrl_15_B.this,
                    myTimeSetListener, my_Hour, my_Minute, false);
                /*显示出日期设置对话框*/
                my_timePickerDialog.show();
            }
            break;
        default:
            break;
    }
}
}

```

⑧ 结果



GridView 宫格视图实践

BaseAdapter 与 GridView

- ① 新建工程
- ② 在 res/drawable 目录下添加名称为 a.png---p.png 的图片
- ③ 修改 main.xml 布局, 添加一个 GridView、一个 ImageView



```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/widget0"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <GridView
        android:id="@+id/grid"
        android:layout_width="fill_parent"
        android:padding="30dip"
        android:columnWidth="52px"
        android:layout_height="210px"
        android:numColumns="5">
        <!-- GridView设置为五列 边距为30pid-->
    </GridView>
    <ImageView
        android:id="@+id/ImageView_Big"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="95px"
        android:layout_y="250px">
    </ImageView>
</AbsoluteLayout>
```

- ④ GridView 的定义与实例化

```
/*定义类对象*/
private GridView my_gridview;
/*从xml中获取UI资源对象*/
my_gridview = (GridView) findViewById(R.id.grid);
```

⑤ GridView 的图像内容设置与 ImageAdapter

```
/*新建一个自定义的ImageAdapter*/  
myImageViewAdapter = new ImageAdapter(this);  
/*为GridView对象设置一个ImageAdapter*/  
my_gridview.setAdapter(myImageViewAdapter);
```

⑥ 内部类 ImageAdapter, 实现了 BaseAdapter

```
private class myImageAdapter extends BaseAdapter{  
    @Override  
    public int getCount() {  
        // TODO Auto-generated method stub  
        return 0;  
    }  
    @Override  
    public Object getItem(int position) {  
        // TODO Auto-generated method stub  
        return null;  
    }  
    @Override  
    public long getItemId(int position) {  
        // TODO Auto-generated method stub  
        return 0;  
    }  
    @Override  
    public View getView(int position,  
                        View convertView, ViewGroup parent) {  
        // TODO Auto-generated method stub  
        return null;  
    }  
}
```

⑦ GridView 的图片 Items 点击事件处理

```
/*为GridView添加图片Items点击事件监听器*/  
my_gridview.setOnItemClickListener(this);  
@Override  
public void onItemClick(AdapterView<?> arg0,  
                        View arg1, int arg2, long arg3) {  
    // TODO Auto-generated method stub  
    /*点击GridView中图片Items事件处理*/  
}
```


⑧ GridView 移动后选中图片 Items 的事件处理

```
/*为GridView添加图片Items移动选中事件监听器*/
my_gridview.setOnItemClickListener(this);
@Override
public void onItemClick(AdapterView<?> arg0,
                        View arg1, int arg2,long arg3) {
    // TODO Auto-generated method stub
    /*GridView中的图片移动焦点选中时事件处理*/
}
/*未选中GridView中的图片Items事件处理*/
@Override
public void onNothingSelected(AdapterView<?> arg0) {
    // TODO Auto-generated method stub
}
```

⑨ 修改 MainActivity.java 来实现图片点击和图片移动选中的效果

```
package zyf.GridViewTest;
/*导入要使用的包*/
import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.DialogInterface.OnClickListener;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;
public class GridViewTest extends Activity implements
                        GridView.OnItemClickListener,
                        GridView.OnItemSelectedListener {
    /** Called when the activity is first created. */
    /*定义类对象*/
    private GridView my_gridview;
    private ImageView big_imageView;
    private ImageAdapter myImageViewAdapter;
    /*内部类，实现一个图片适配器*/
    public class ImageAdapter extends BaseAdapter {
        /*myContext为上下文*/
        private Context myContext;
        /*GridView用来加载图片的ImageView*/
        private ImageView the_imageView;
```

```
// 这是图片资源ID的数组
private Integer[] mImageIds = {
    R.drawable.a, R.drawable.b, R.drawable.c, R.drawable.d,
    R.drawable.e, R.drawable.f, R.drawable.g, R.drawable.h,
    R.drawable.i, R.drawable.j, R.drawable.k, R.drawable.l,
    R.drawable.m, R.drawable.n, R.drawable.o, R.drawable.p
};

/*构造方法*/
public ImageAdapter(Context myContext) {
    // TODO Auto-generated constructor stub
    this.myContext = myContext;
    /*传入一个Context，本例中传入的是GridViewTest */
}

/*返回资源ID数组长度*/
@Override
public int getCount() {
    // TODO Auto-generated method stub
    return mImageIds.length;
}

/*得到Item*/
@Override
public Object getItem(int position) {
    // TODO Auto-generated method stub
    return position;
}

/*获取Items的ID*/
@Override
public long getItemId(int position) {
    // TODO Auto-generated method stub
    return position;
}

/*获取要显示的View对象*/
@Override
public View getView(int position,
                    View convertView, ViewGroup parent) {
    // TODO Auto-generated method stub
    /*创建一个ImageView*/
    the_imageView = new ImageView(myContext);
    // 设置图像源于资源ID。
    the_imageView.setImageResource(mImageIds[position]);
    /*使ImageView与边界适应*/
    the_imageView.setAdjustViewBounds(true);
}
```



```

        /*设置背景图片的风格*/
        the_imageView.setBackgroundResource(
                                android.R.drawable.picture_frame);

        /*返回带有多个图片ID的ImageView*/
        return the_imageView;
    }

    /*自定义获取对应位置的图片ID*/
    public Integer getcheckedImageIDPostion(int theindex) {
        return mImageIds[theindex];
    }
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    /*设置主屏布局*/
    setContentView(R.layout.main);
    /*从xml中获取UI资源对象*/
    my_gridview = (GridView) findViewById(R.id.grid);
    big_imageView =
        (ImageView) findViewById(R.id.ImageView_Big);
    /*新建一个自定义的ImageAdapter*/
    myImageViewAdapter = new ImageAdapter(this);
    /*为GridView对象设置一个ImageAdapter*/
    my_gridview.setAdapter(myImageViewAdapter);
    /*为GridView添加图片Items点击事件监听器*/
    my_gridview.setOnItemClickListener(this);
    /*为GridView添加图片Items移动选中事件监听器*/
    my_gridview.setOnItemSelectedListener(this);
}

@Override
public void onItemClick(AdapterView<?> arg0,
                        View arg1, int arg2, long arg3) {
    /*点击GridView中图片Items后显示一个AlertDialog提示框*/
    new AlertDialog.Builder(this)
        .setTitle("图片浏览")
        /*获得对应的图片并显示*/
        .setIcon(myImageViewAdapter.getcheckedImageIDPostion(arg2))
        /*添加一个按钮*/
        .setPositiveButton("返回", new OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog,
                                int which) {
            }
        })
        /*显示提示框*/

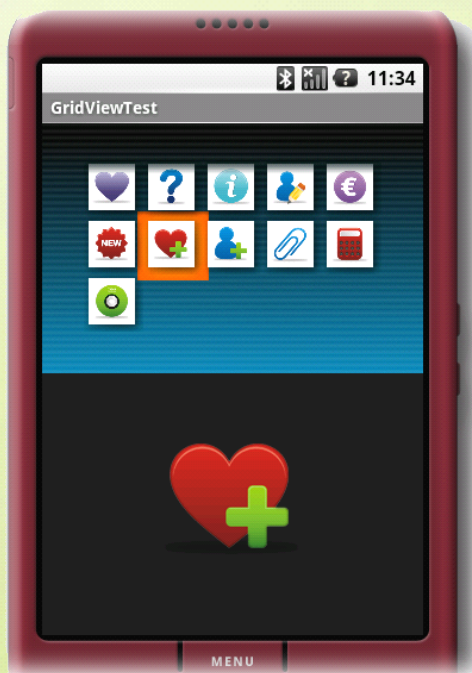
```

```

        }).show();
    }
    @Override
    public void onItemClick(AdapterView<?> arg0,
                            View arg1, int arg2, long arg3) {
        // TODO Auto-generated method stub
        /*GridView中的图片移动焦点选中时,
        *下面的大图ImageView显示相应的大图片*/
        big_imageView.setImageResource(myImageViewAdapter
            .getcheckedImageIDPostion(arg2));
    }
    /*未选中GridView中的图片Items事件处理*/
    @Override
    public void onNothingSelected(AdapterView<?> arg0) {
        // TODO Auto-generated method stub
    }
}

```

⑩ 结果



android
developers



android

android
developers

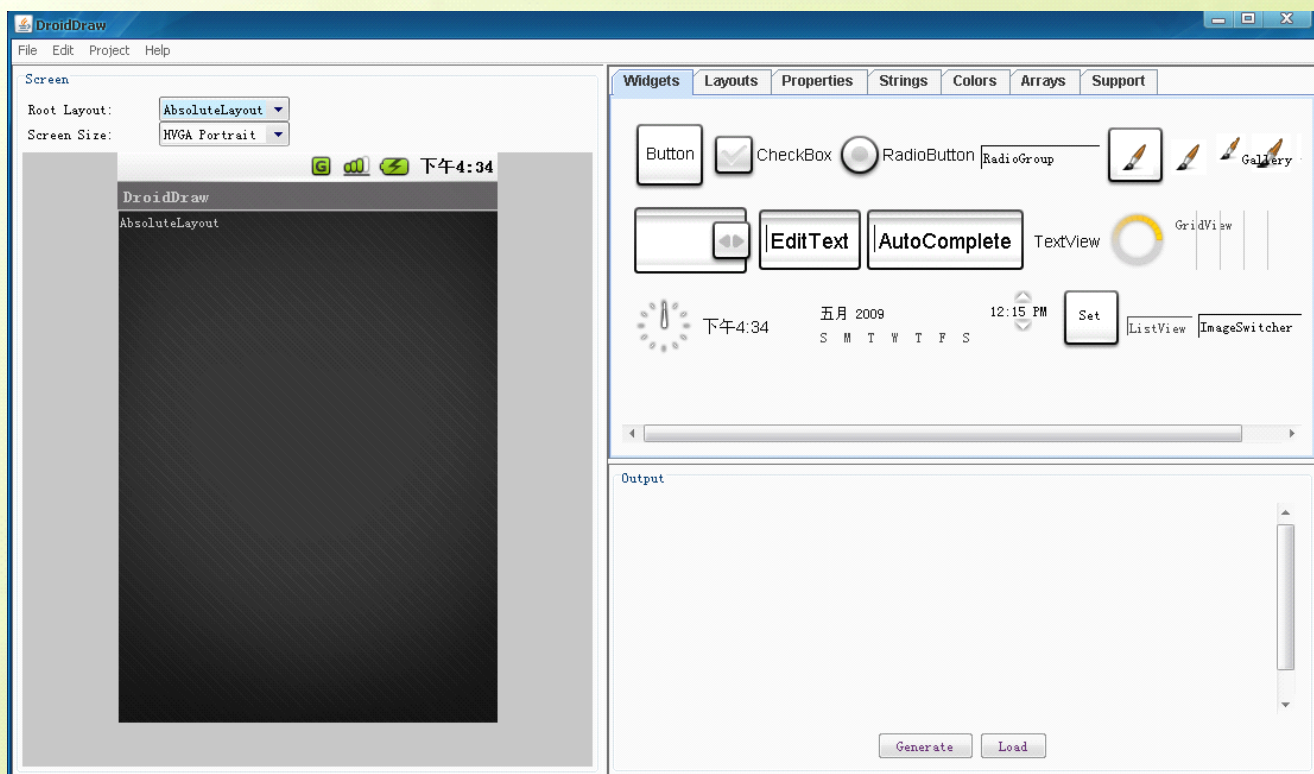
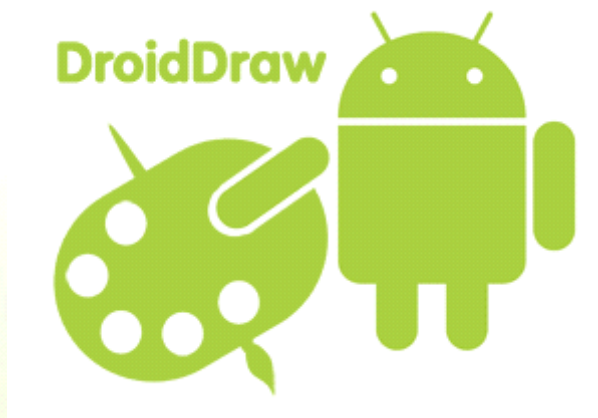
Android开发教程&笔记



android

GUI 可视化设计器——DroidDraw

DroidDraw 是一个基于 Java Swing 的 Android 界面设计器，可以通过它来生成复杂的 Android Layout XML 文件，Android 的 Layout 和 Swing Layout 中有很好的对应，设计器的代码编写起来比较容易。



AnDroidDraw

AnDroidDraw 是一个与 DroidDraw 集成的 Android 应用程序,它允许你从 DroidDraw 应用程序下载你的 GUIs,也允许你在一个 Android 设备上预览你的 GUIs。[下载 DroidDraw](#)

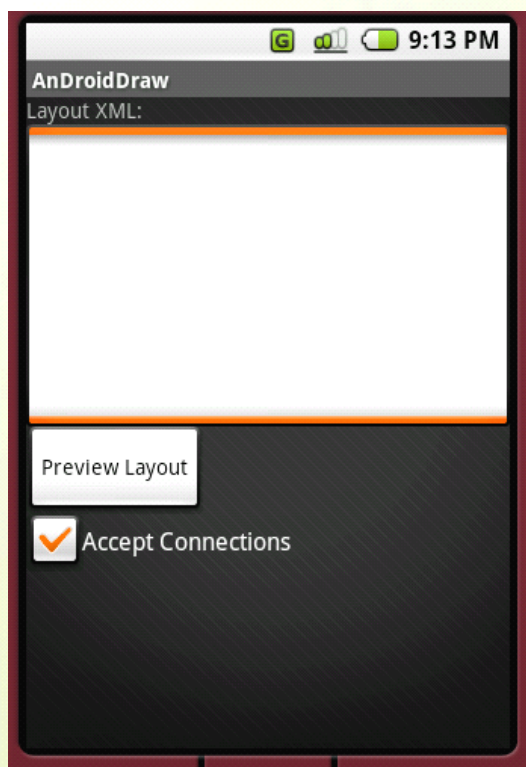
步骤零

- ◆ 下载 [AnDroidDraw.apk](#)
- ◆ 使用: `adb install AnDroidDraw.apk` 把它安装到你的 Android 设备上
- ◆ 安装一个端口转发规则: `adb forward tcp:6100 tcp:7100`



步骤一

在你的 Android 设备上运行 AnDroidDraw, 你应该看到像这样的:

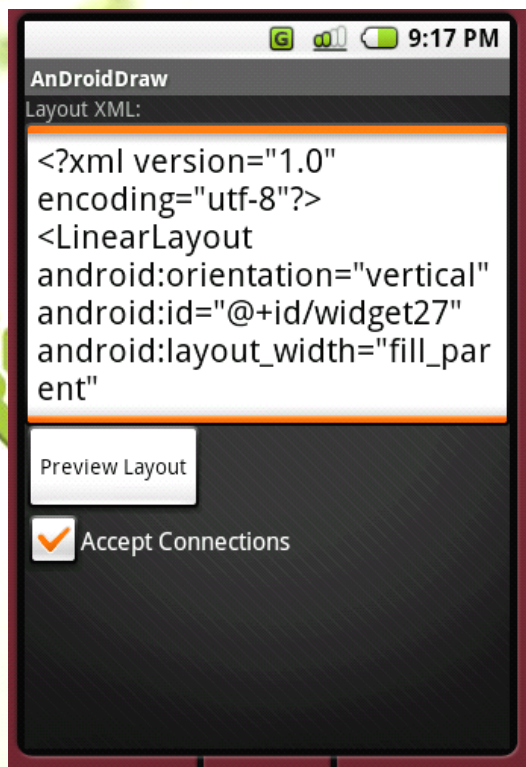


步骤二

在你的电脑上运行 DroidDraw, 并且创建一个 GUI, (获取更多关于创建 GUI 的信息, 请看[教程 1](#)、[教程 2](#)、[教程 3](#)。)接下来从 DroidDraw 菜单中选择"Project"->"Send GUI to Deviec"

步骤三

现在你应该在 Android 屏幕上看到你新创建的 GUI 的像这样的 xml:



步骤四

点击"Preview GUI"按钮来预览你的 GUI。

步骤五

当你结束时，点击向后的箭头，来返回到 AnDroidDraw 的主屏幕。

记住，如果你感兴趣，你可以在文本框中编辑该 XML 文件，并且再次点击"Preview GUI"来查看你的修改。然而，这些修改将不会返回到 DroidDraw。

步骤六

大功告成！?? /Comments/Bugs brendan.d.burns 在 gmail 上。

DroidDraw 教程一：Currency Converter

步骤零

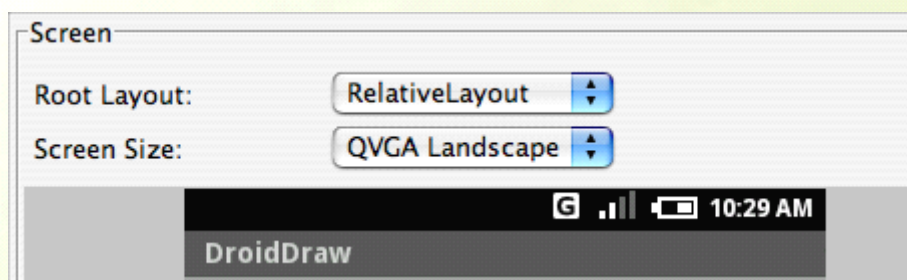
本教程将给你一个简短的介绍关于使用 DroidDraw 用户界面设计器来开发一个在 Android 上的 GUI 应用程序。本教程假设你已经下载并安装了 Android SDK。本教程也假设你对 GUI 编程概念和 Java 编程语言相当熟悉。

步骤一

登陆到 [DroidDraw UI Designer](#)

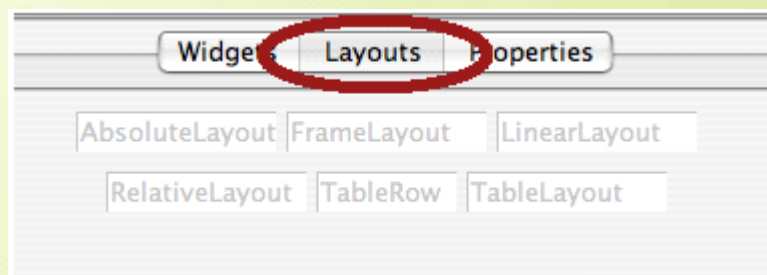
步骤二

设置根布局为 RelativeLayout(相对布局)



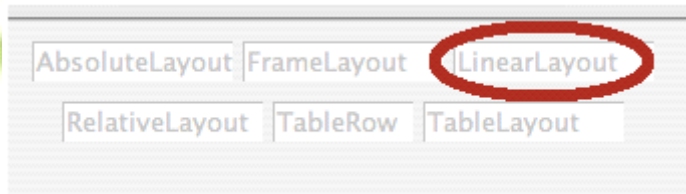
步骤三

选择"Layout"标签



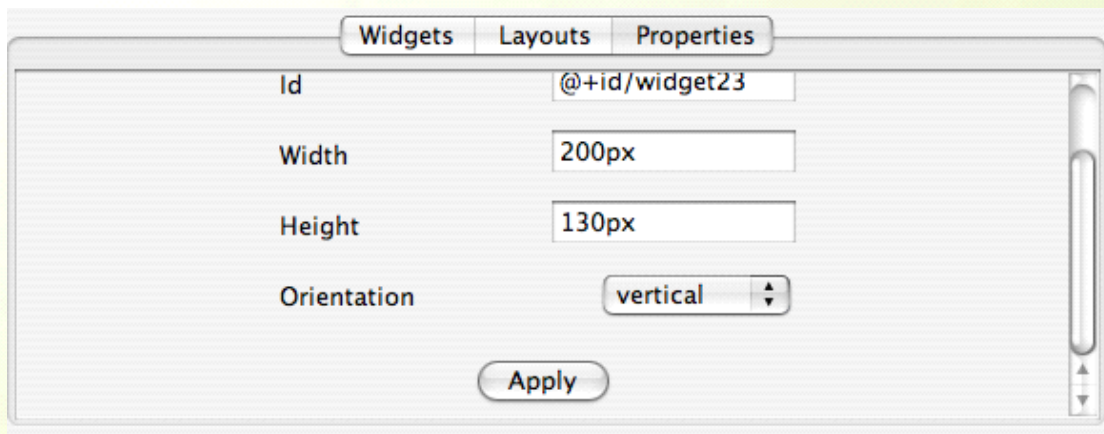
步骤四

从 Layouts 面板中把一个 LinearLayout 对象拖放到屏幕顶部中心位置



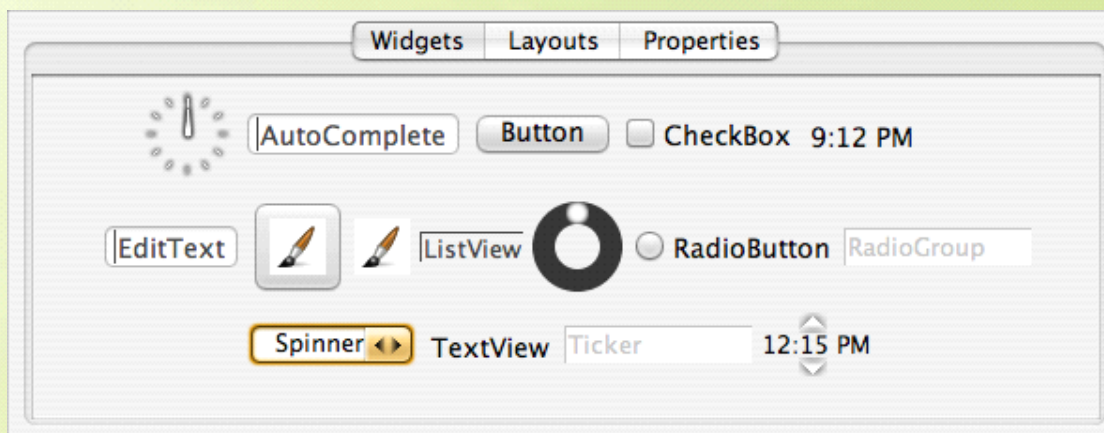
步骤五

选择该 LinearLayout 对象并点击属性 "Properties" 标签来开始编辑 layout 属性值。把宽度 "width" 改成 "200px", 高度 "height" 改成 "130px"。点击 "Apply" 来应用改变。



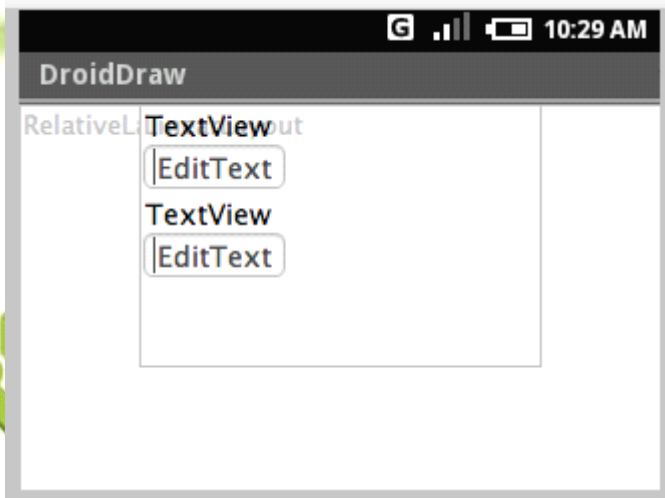
步骤六

转到 "Widgets" 标签



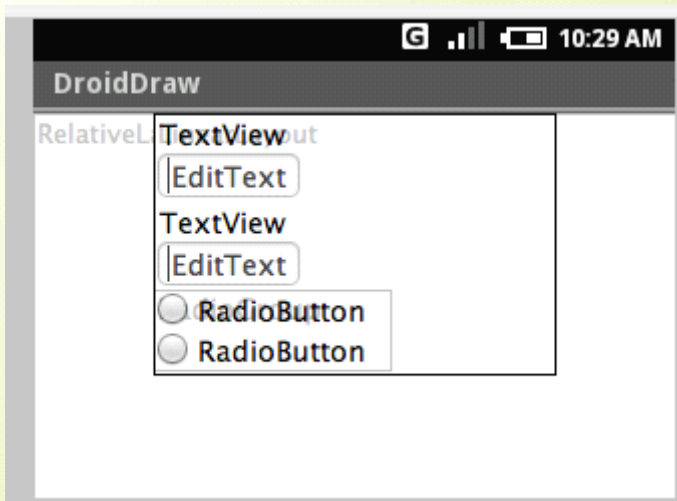
步骤七

把两个 TextView 对象和两个 EditText 对象交替地拖放到 LinearLayout 中



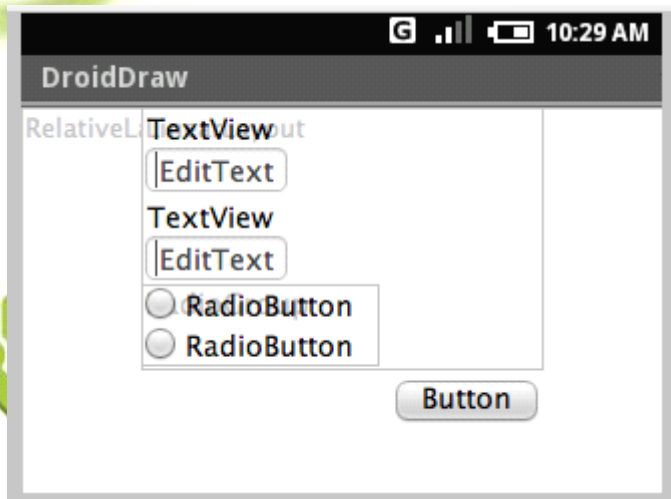
步骤八

把一个 RadioGroup 对象拖放进 LinearLayout 中。把两个 RadioButton 对象拖放到 RadioGroup 中。



步骤九

把一个 **Button** 对象拖放到根 **RelativeLayout** 中，它在 **LinearLayout** 对象下面。它应该和 **LinearLayout** 的右边对齐。



步骤十

编辑每个 **TextView** 对象的属性值。上面一个的文本设置成"Dollars"，并设置成"bold"字体样式。下面一个 **TextView** 的文本设置成"Euros"，并也设置成"bold"字体样式。

步骤十一

如以下内容编辑上面一个 **EditText** 的属性值：

- ◆ id 修改成: "@+id/dollars"
- ◆ 文本内容设置为空
- ◆ 宽度修改成"100px"

步骤十一半

在"Euros" **TextView** 下面的第二个 **EditText** 上重复步骤十一，但是把 id 设置为"@+id/euros"

步骤十二

编辑第一个 **RadioButton** 属性：文本设置为"Dollars to Euros",并把它 id 设置成"@+id/dtoe"

编辑第二个 **RadioButton** 属性：文本设置为"Euros to Dollars",并把它 id 设置成"@+id/etod"

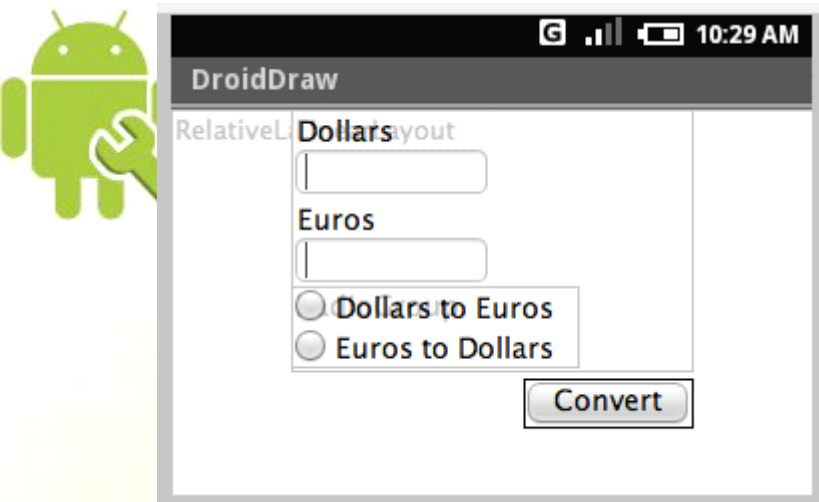
重要注意事项

你必须正确地获取 id，因为这是你在代码中如何获取搜索到该 UI 元素的方式。

步骤十三

编辑 Button 属性：文本修改为"Convert"、它的 id 设置成"@+id/convert"。

最终的 GUI 应该像这样：



步骤十四

点击"Generate"按钮来生成 XML 布局。该 xml 应像这样：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    android:id="@+id/widget30"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <LinearLayout
        android:id="@+id/widget31"
        android:layout_width="180px"
        android:layout_height="228px"
        android:orientation="vertical"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true">
        <TextView
            android:id="@+id/widget41"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```




```
        android:text="Dollars"
        android:textStyle="bold">
</TextView>
<EditText
    android:id="@+id/dollars"
    android:layout_width="100px"
    android:layout_height="wrap_content"
    android:textSize="18sp"></EditText>
<TextView
    android:id="@+id/widget43"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Euros"
    android:textStyle="bold"></TextView>
<EditText
    android:id="@+id/euros"
    android:layout_width="100px"
    android:layout_height="wrap_content"
    android:textSize="18sp"></EditText>
<RadioGroup
    android:id="@+id/widget45"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <RadioButton
        android:id="@+id/dtoe"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Dollars to Euros">
    </RadioButton>
    <RadioButton
        android:id="@+id/etod"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Euros to Dollars">
    </RadioButton>
</RadioGroup>
</LinearLayout>
<Button
    android:id="@+id/convert"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Convert"
    android:layout_below="@+id/widget31"
```

```

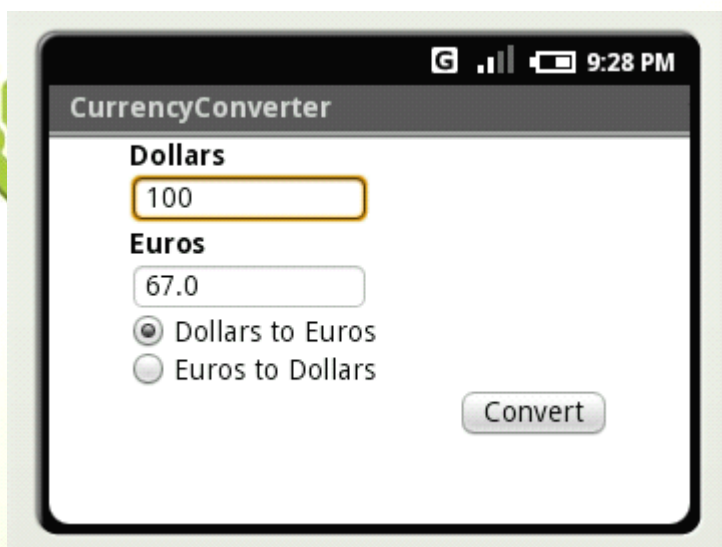
        android:layout_alignRight="@+id/widget31">
</Button>
</RelativeLayout>

```

步骤十五

在 Eclipse 中创建一个新的 Android 工程。从 DroidDraw 剪切该 XML 并粘贴替换到 res/layout/main.xml 的内容中。

到这里你就可以在 Android 中运行你的 GUI。它应该像这样：



步骤十六

最后一步是实际的代码货币转换。它不多，你可以用一下代码来查找到你的 GUI 元素：

```
this.findViewById(R.id.<id>);
```

下面是完整 CurrentConverter Activity 的代码：

```

package zyf.CurrentConverter;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.TextView;
public class CurrentConverter extends Activity
                                implements OnClickListener {
    TextView dollars;
    TextView euros;
    RadioButton dtoe;

```




```
RadioButton etod;
Button convert;
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    dollars = (TextView) this.findViewById(R.id.dollars);
    euros = (TextView) this.findViewById(R.id.euros);
    dtoe = (RadioButton) this.findViewById(R.id.dtoe);
    dtoe.setChecked(true);
    etod = (RadioButton) this.findViewById(R.id.etod);
    convert = (Button) this.findViewById(R.id.convert);
    convert.setOnClickListener(this);
}

public void onClick(View v) {
    if (dtoe.isChecked()) {
        convertDollarsToEuros();
    }
    if (etod.isChecked()) {
        convertEurosToDollars();
    }
}

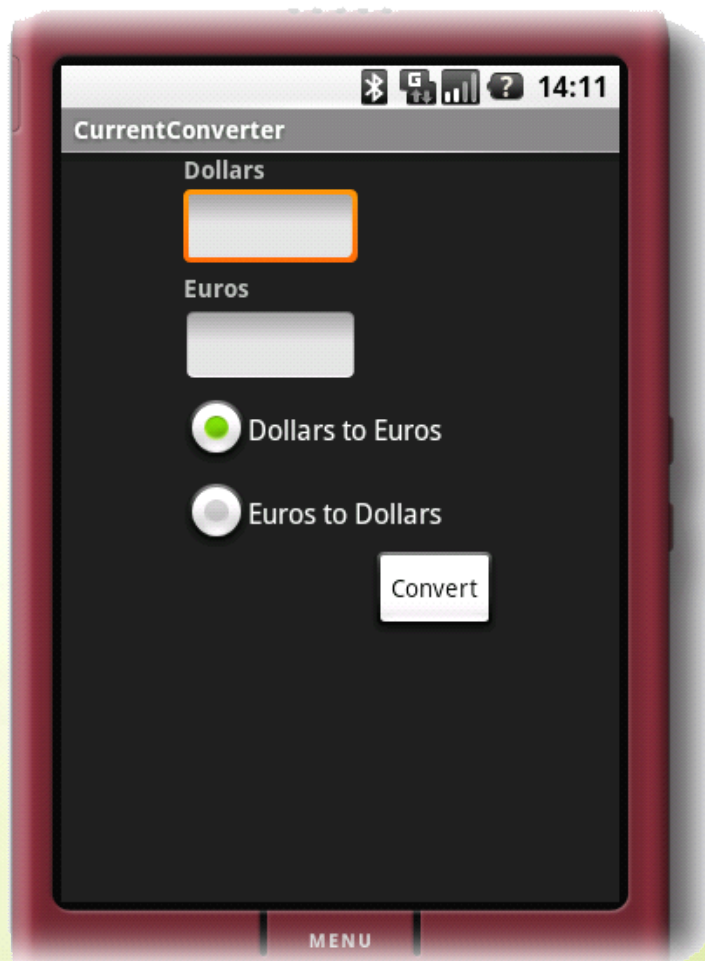
protected void convertDollarsToEuros() {
    double val =
        Double.parseDouble(dollars.getText().toString());
    // in a real app, we'd get this off the 'net
    euros.setText(Double.toString(val * 0.67));
}

protected void convertEurosToDollars() {
    double val = Double.parseDouble(euros.getText().toString());
    // in a real app, we'd get this off the 'net
    dollars.setText(Double.toString(val / 0.67));
}
}
```

步骤十七

嗯，就是这样。我希望你喜欢该教程。意见和问题邮件 [brendan.d.burns Gmail](mailto:brendan.d.burns@gmail.com) !

结果



DroidDraw 教程二: Table Layout

步骤零

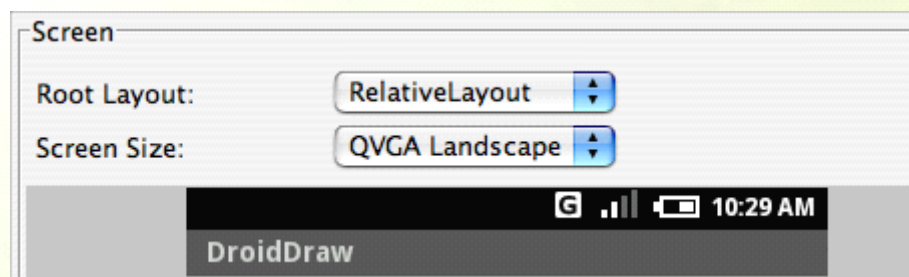
本教程描述如何创建一个从 DroidDraw 简单的输入和 TableLayout 布局。本教程假设你已经下载并安装了 Android SDK。本教程也假设你对 GUI 编程概念和 Java 编程语言相当熟悉。

步骤一

启动 [DroidDraw 用户界面设计器](#)

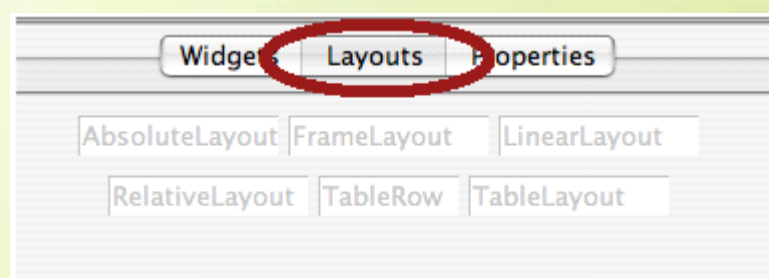
步骤二

根布局选择为 RelativeLayout 布局



步骤三

选择 "Layouts" 标签



步骤四

把一个 `TableLayout` 对象从 `Layouts` 面板中拖放到屏幕顶的中部。



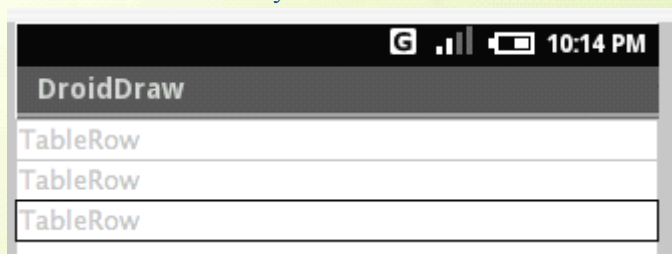
步骤五

双击 `TableLayout` 来修改它的属性。把它的宽度 `width` 改为 `fill_parent`



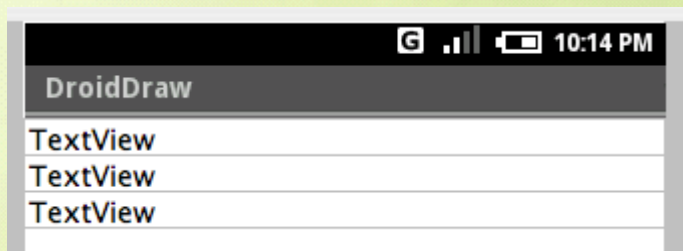
步骤六

把三个 `TableRow` 对象从 `Layouts` 面板中拖放到 `TableLayout` 对象中。当你拖放 `TableRow` 对象时，你应该从弹出菜单中选择 `TableLayout`。



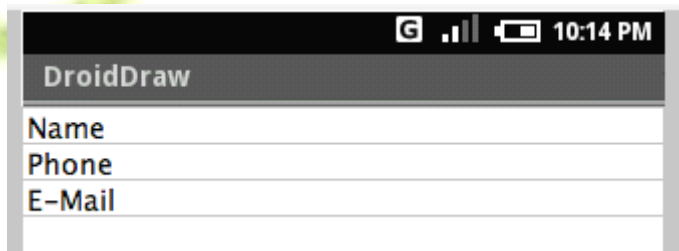
步骤七

每一个 `TableRow` 中拖放一个 `TextView`：



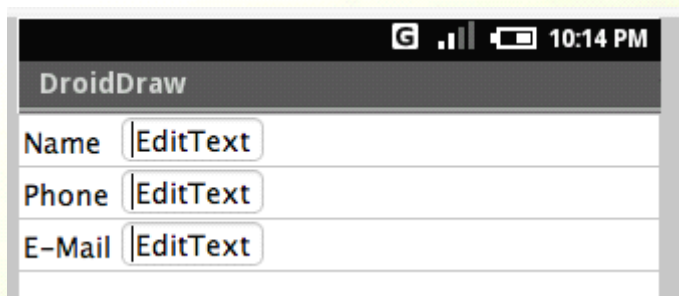
步骤八

双击每一个 TextView 来修改它的属性，修改显示文本如下图一样：



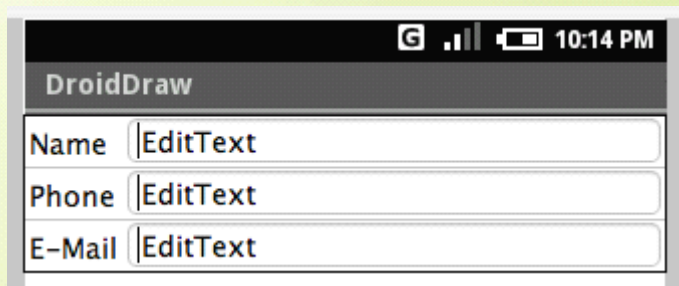
步骤九

每一个 TableRow 中拖放一个 EditText，放在存在的文本右边。



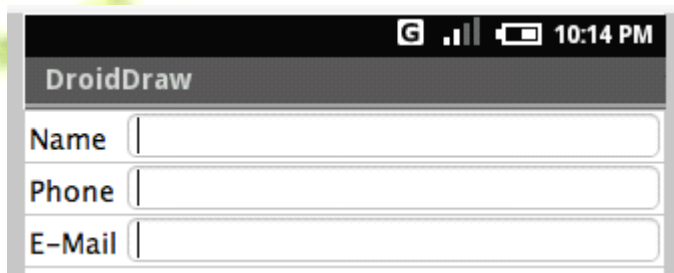
步骤十

选中 TableLayout，修改"Stretchable Column"(可扩展栏)属性值为 1，这将把所有的 EditText widget 扩展开来填满该 Table 表格。



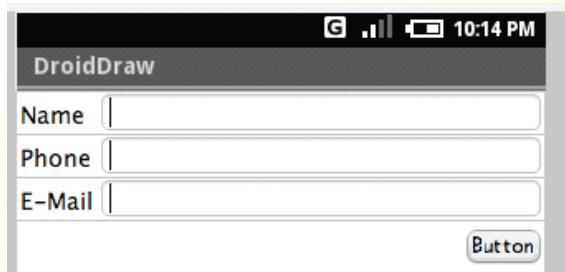
步骤十一

编辑每一个 EditText 的属性，让 Text 文本属性为""



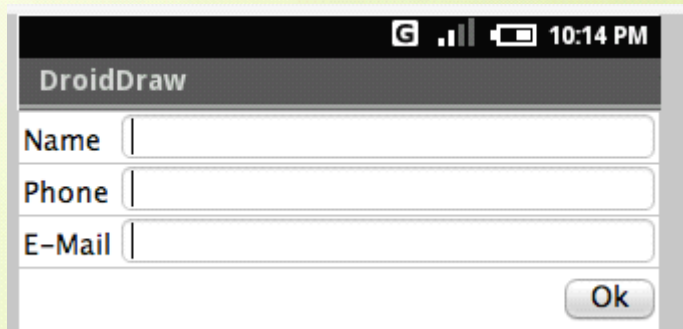
步骤十二

把一个 Button 拖放到 TableLayout 下面的右下角空白处。它应该在 TableLayout 的外面并和它右对齐。



步骤十三

修改该按钮的属性，文本设置为"OK"



步骤十四

点击"Generate"按钮来生成.xml 文件

步骤十五

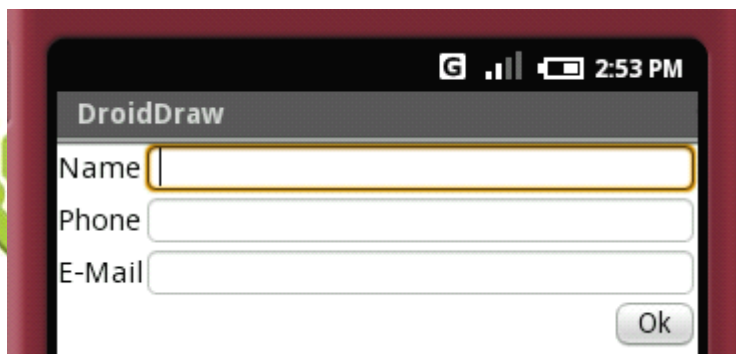
在 Eclipse 中，创建一个新的 Android 工程

步骤十六

用第十五步骤生成的 XML 来替换 res/layouts/mian.xml 文件内容。

步骤十七

运行你的新工程，你应该在 Android 中看到你的 GUI。它应该像这样：



完成！

完整 XML 文件

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    android:id="@+id/widget49"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <TableLayout
        android:id="@+id/widget54"
        android:layout_height="160px"
        android:orientation="vertical"
        android:stretchColumns="1"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" android:layout_width="fill_parent">
        <TableRow
            android:id="@+id/widget55"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal">
            <TextView
                android:id="@+id/widget58"
```

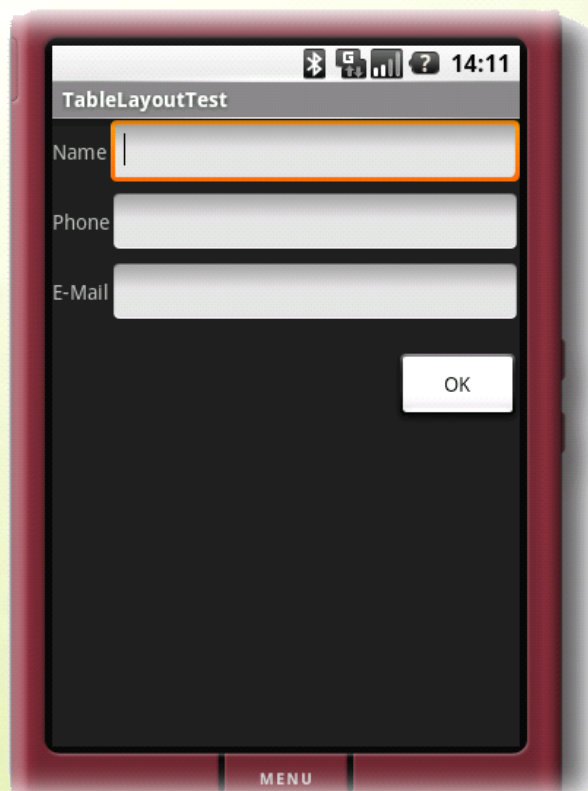
```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Name">
</TextView>
<EditText
    android:id="@+id/widget61"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="18sp">
</EditText>
</TableRow>
<TableRow
    android:id="@+id/widget56"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <TextView
        android:id="@+id/widget59"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Phone">
    </TextView>
    <EditText
        android:id="@+id/widget62"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="18sp">
    </EditText>
</TableRow>
<TableRow
    android:id="@+id/widget57"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <TextView
        android:id="@+id/widget60"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="E-Mail">
    </TextView>
    <EditText
        android:id="@+id/widget63"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```



```
        android:textSize="18sp">
    </EditText>
</TableRow>
</TableLayout>
<Button
    android:id="@+id/widget64"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button"
    android:layout_below="@+id/widget54"
    android:layout_alignRight="@+id/widget54">
</Button>
</RelativeLayout>
```



结果



DroidDraw 教程三:使用 ListView 和 array 资源

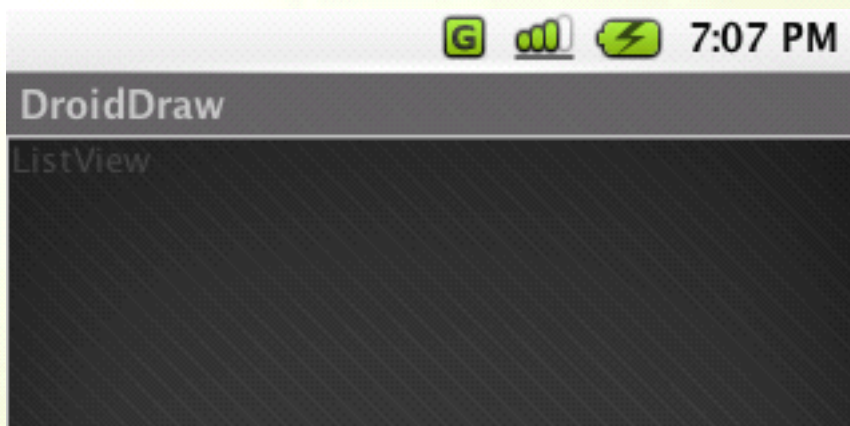
步骤零

在 Eclipse 新建一个工程

步骤一 - 创建初始化布局



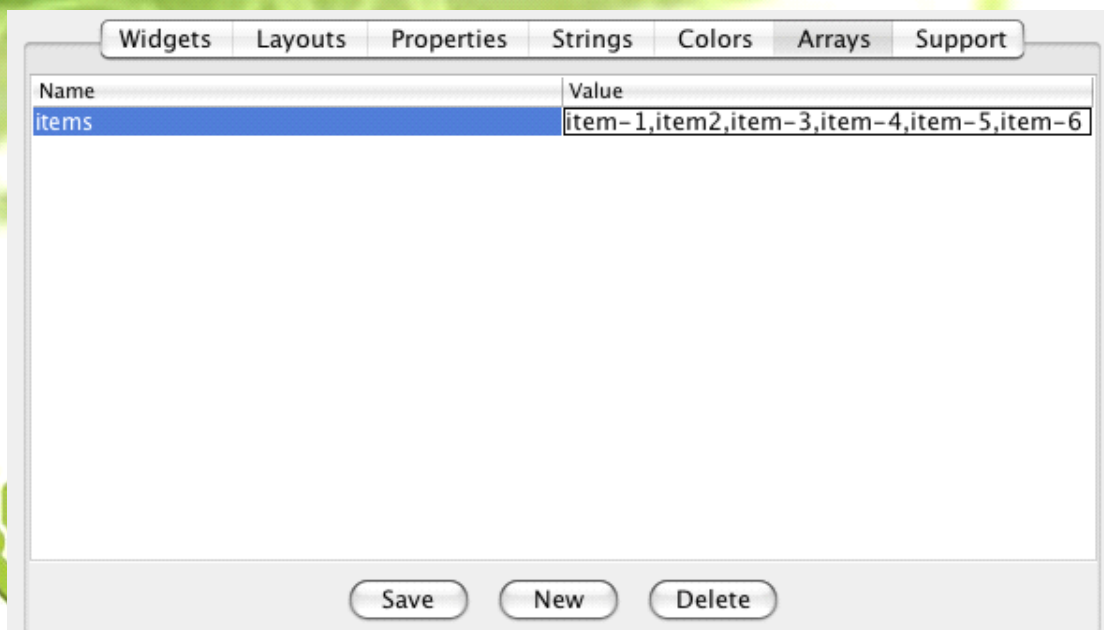
- ◆ 开启 DroidDraw 并创建一个新的 Layout
- ◆ 从 Widget 列表中拖放一个 ListView 放入该 Layout 中
- ◆ 双击该 ListView 编辑它的属性
- ◆ 把它的宽、高属性值改为"fill_parent"
- ◆ 点击"Apply"按钮



步骤二 - 创建一个 Array 数组资源

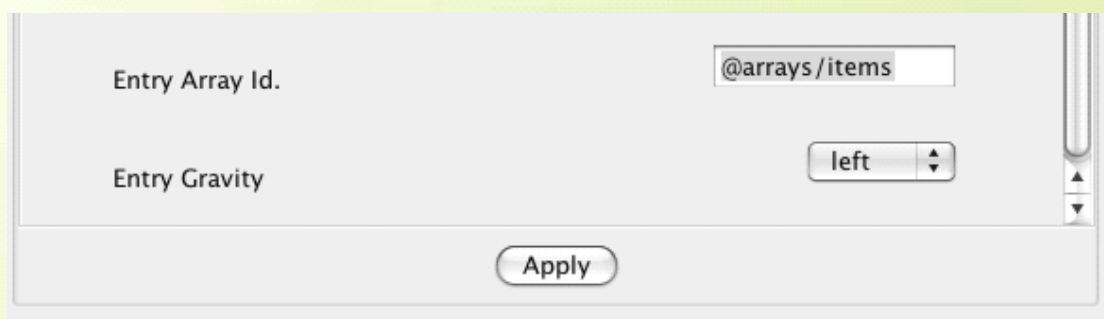
注意：这些使用说明是针对独立的 DroidDraw 可执行文件的。

- ◆ 点击 DroidDraw 中的"Arrays"标签
- ◆ 点击"New"按钮来添加一个新的 Array 数组
- ◆ 当提示名称时，使用"items"
- ◆ 对于数组值，使用","逗号来隔开列表的值
- ◆ 点击"Save"按钮并把该文件保存为 arrays.xml，保存在你工程 res/values 目录中



步骤三 - 让你的列表和数组连接

- ◆ 在你第一步创建的 ListView 上双击
- ◆ 修改"Entry Array Id"属性为"@+id/items"
- ◆ 点击"Apply"按钮
- ◆ 生成 Layout 布局文件并保存它为 main.xml，保存到你工程 res/layouts 目录中



步骤四 - 代码

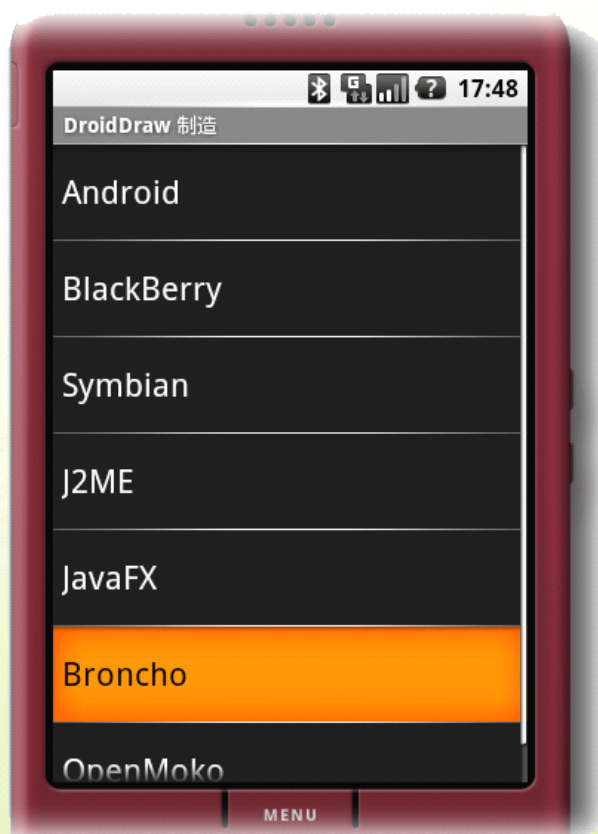
使用以下代码在你的 MainActivity.java 文件中：

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    /** Called when the activity is first created. */
    super.onCreate(savedInstanceState);
    this.setTitle("DroidDraw");
    setContentView(R.layout.main);
}
```

步骤五 - 完成

在 Android 模拟器中运行你的代码

结果



Android GUI Widget 可视化指导

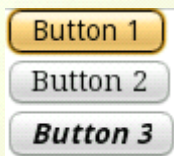
作为一个 Java Android 手机开发员、UI 设计者，为了让你的生活更简单。尝试用 [DroidDraw](#) 来高速开发你的用户界面。

AnalogClock



```
<AnalogClock
    android:id="@+id/clock1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
/>
```

Button



```
<Button android:id="@+id/button1"
    android:text="Label"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"/>
<Button
    android:id="@+id/button2"
    android:text="Label"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:typeface="serif"/>
<Button
    android:id="@+id/button3"
    android:text="Label"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:textStyle="bold|italic"/>
```

CheckBox

- ☒ Plain
- ☐ Serif
- ☐ **Bold**
- ☐ *Italic*

```
<CheckBox
    android:id="@+id/plain_cb"
    android:text="Plain"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
/>
<CheckBox
    android:id="@+id/serif_cb"
    android:text="Serif"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:typeface="serif"
/>
<CheckBox
    android:id="@+id/bold_cb"
    android:text="Bold"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textStyle="bold"
/>
<CheckBox
    android:id="@+id/italic_cb"
    android:text="Italic"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textStyle="italic"
/>
```


DatePicker

S	M	T	W	T	F	S
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

```
<DatePicker
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >
.....
// Required Java init code:
    DatePicker dp =
        (DatePicker) this.findViewById(R.id.widget27);

// for example init to 1/27/2008, no callback
dp.init(2008, 0, 27, Calendar.SUNDAY, null);
...
```

DigitalClock

9:26:00 pm

```
<DigitalClock
    android:id="@+id/digitalclock"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

EditText

EditText 1

(206)555-1212

••••••••••

```
<EditText
    android:id="@+id/edittext1"
    android:text="EditText 1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
/>
<EditText
    android:id="@+id/button2"
    android:text="(206)555-1212"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:typeface="serif"
    android:phoneNumber="true"
/>
<EditText
    android:id="@+id/password"
    android:text="SuperSecret"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textStyle="bold|italic"
    android:password="true"
/>
```

Gallery



```
<Gallery
    android:id="@+id/gallery"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```


ImageButton



```
<ImageButton
    android:id="@+id/imagebutton"
    android:src="@drawable/brush"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

ImageView



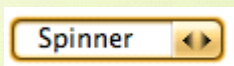
```
<ImageView
    android:id="@+id/imagebutton"
    android:src="@drawable/brush"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

ProgressBar



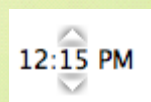
```
<ProgressBar
    android:id="@+id/progress"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

Spinner



```
<Spinner
    android:id="@+id/widget1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:drawSelectorOnTop="false"/>
```

TimePicker



```
<TimePicker
    android:id="@+id/widget3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

RadioButton



```

<RadioGroup
    android:id="@+id/widget1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical">

<RadioButton
    android:id="@+id/widget2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Plain"
    android:checked="false"
    android:layout_gravity="left"
    android:layout_weight="0">
</RadioButton>
<RadioButton
    android:id="@+id/widget3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Serif"
    android:checked="true"
    android:layout_gravity="left"
    android:layout_weight="0"
    android:typeface="serif">
</RadioButton>
<RadioButton
    android:id="@+id/widget25"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Bold"
    android:checked="false"
    android:layout_weight="0"
    android:layout_gravity="left"
    android:textStyle="bold">
</RadioButton>
<RadioButton
    android:id="@+id/widget24"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Bold & Italic"


```



```

        android:checked="false"
        android:layout_weight="0"
        android:layout_gravity="left"
        android:textStyle="bold_italic">
</RadioButton>
</RadioGroup>

```


TextView

Plain
 Serif
Bold
Italic

```

<TextView
    android:id="@+id/plain"
    android:text="Plain"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
<TextView
    android:id="@+id/serif"
    android:text="Serif"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:typeface="serif"/>
<TextView
    android:id="@+id/bold"
    android:text="Bold"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textStyle="bold"/>
<TextView
    android:id="@+id/italic"
    android:text="Italic"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textStyle="italic"/>

```

android
developers



android

android
developers

Android开发教程&笔记



android

文件存取编程基础

文件

- ◆ 文件可以用来存储比使用引用更大数量的数据
- ◆ Android 提供方法来读、写文件
- ◆ 只有本地文件可以被访问
- ◆ 优点：可以存储大容量的数据
- ◆ 缺点：文件更新或是格式改变可能会导致巨大的编程工作



文件操作

读文件

- `Context.openFileInput(String name)` 打开一个与应用程序联系的私有文件输入流
- 当文件不存在时抛出 `FileNotFoundException` 异常

```
FileInputStream in = this.openFileInput("test2.txt");//打开文件"test2.txt"
.....
in.close();//关闭输入流
```

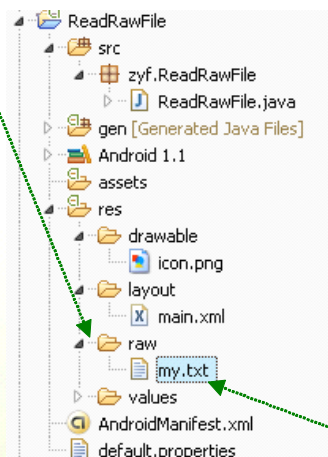
写文件

- `Context.openFileOutput(String name,int mode)` 开启一个与应用程序联系的私有文件输出流
- 当文件不存在时该文件将被创建
- 文件输出流可以在添加模式中打开，这意味新的数据将被添加到文件的末尾

```
FileOutputStream out = this.openFileOutput("test2.txt",MODE_APPEND);
//打开文件"test2.txt"进行写操作 、使用 MODE_APPEND 在添加模式中打开文件
.....
out.close();//关闭输出流
```


读取静态文件

- 要打开打包在应用程序中的静态文件，使用 `Resources.openRawResource(R.raw.mydatafile)`
- 该文件必须放在文件夹 `res/raw/` 中



```
InputStream in = this.getResources().openRawResource(R.raw.my);  
... //获得Context资源  
in.close(); //关闭输入流
```

文件存取示例

创建添加文件内容并保存，打开文件并显示内容

① 新建工程 FileWriteRead

② 修改 main.xml 布局，添加一个 EditText、一个 Button



```
<?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"
    >
    <EditText
        android:id="@+id/EditText_Txt"
        android:layout_height="350px"
        android:layout_width="fill_parent">
    </EditText>
    <Button
        android:layout_height="wrap_content"
        android:id="@+id/Button_Save"
        android:text="保存"
        android:layout_width="80px">
    </Button>
</LinearLayout>
```

③ 在 res/layout 中新建一个 open.xml 布局文件，添加一个 TextView、两个 Button("打开","清空")

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/openlayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <TextView
        android:id="@+id/TextView_showTxt"
        android:layout_width="314px"
        android:layout_height="373px"
        android:layout_x="3px"
        android:layout_y="3px">
    </TextView>
    <Button
        android:id="@+id/Button_openTxt"
```



```

    android:layout_width="80px"
    android:layout_height="wrap_content"
    android:text="打开"
    android:layout_x="2px"
    android:layout_y="378px">
</Button>
<Button
    android:id="@+id/Button_clean"
    android:layout_width="80px"
    android:layout_height="wrap_content"
    android:text="清空"
    android:layout_x="239px"
    android:layout_y="378px">
</Button>
</AbsoluteLayout>

```

④ 文件存储

```

/*定义IO对象*/
private String Text_of_input;
private OutputStream os;
Text_of_input = inputArt.getText().toString();
//得到用户输入字符
    try {
        os = this.openFileOutput("txtME", MODE_PRIVATE);
        //打开一个文件输出流。名称为txtME，模式为不覆盖
        os.write(Text_of_input.getBytes());
        //把内容写入文件
    } catch (FileNotFoundException e) {
        // TODO Auto-generated catch block
    } catch (IOException e) {
        // TODO Auto-generated catch block
    } finally {
        try {
            //关闭文件输出流
            os.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
        }
    }
}

```

⑤ 文件读取

```

/*定义IO对象*/
private String Text_of_output;
private InputStream is;

```



```
private byte[] b;
try {

    //打开一个文件输入流。名称为txtME
    is = this.openFileInput("txtME");
    //字节数组声明定义
    b = new byte[1024];
    //读取文件内容放入字节数组
    int length = is.read(b);
    //把字节数组转换成字符串
    Text_of_output = new String(b);
    //显示读取内容长度
    setTitle("文件字数: " + length);
    //显示读取的文件内容
    showmyText.setText(Text_of_output);
} catch (FileNotFoundException e) {
    // TODO Auto-generated catch block
} catch (IOException e) {
    // TODO Auto-generated catch block
} finally {
    try {
        //关闭文件输入流
        is.close();
    } catch (IOException e) {
        // TODO Auto-generated catch block
    }
}
```

⑥ 修改 mianActivity.java 文件，添加 menu 菜单与操作

```
package zyf.FileWrite;
/*导入要使用的包*/
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
```



```

public class FileWrite extends Activity implements Button.OnClickListener {
    /** Called when the activity is first created. */
    /*要使用的对象、变量声明*/
    /*保存部分*/
    private EditText inputArt; /*编辑框, 输入用户字符串*/
    private Button saveButton; /*按钮, 保存*/
    private String Text_of_input; /*字符串, 用户输入的字符串*/
    private OutputStream os; /*文件输出流, 保存文件流*/
    /*读取部分*/
    private TextView showmyText; /*TextView, 显示读取文件内容*/
    private Button openTxt, cleanTxt; /*按钮, 打开文件*/
    private String Text_of_output; /*字符串, 从文件中读取到得 字符串*/
    private InputStream is; /*文件输入流, 读取文件流*/
    private byte[] b; /*字节数组, 用来读取文件内容*/

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main); /*设置主屏布局*/
        UIinit("main"); /*初始化UI元素方法*/
        Logic("main"); /*添加事件逻辑方法*/
    }
    /*设置主屏*/
    private void setContentView(int layoutID) {
        // TODO Auto-generated method stub
        setContentView(layoutID); /*设置当前主屏布局*/
    }
    private void UIinit(String mainROopen) {
        /*初始化UI*/
        if (mainROopen.equals("main")) {
            inputArt = (EditText) findViewById(R.id.EditText_Txt);
            saveButton = (Button) findViewById(R.id.Button_Save);
        } else if (mainROopen.equals("open")) {
            showmyText = (TextView) findViewById(R.id.TextView_showTxt);
            openTxt = (Button) findViewById(R.id.Button_openTxt);
            cleanTxt = (Button) findViewById(R.id.Button_clean);
        }
    }
    private void Logic(String string) {
        // TODO Auto-generated method stub
        /*为按钮添加事件处理*/
        if (string.equals("main")) {
            saveButton.setOnClickListener(this);


```

```

    } else if (string.equals("open")) {
        openTxt.setOnClickListener(this);
        cleanTxt.setOnClickListener(this);
    }
}

@Override
public void onClick(View v) {
    /*根据ID判断按钮事件*/
    switch (v.getId()) {
        case R.id.Button_Save: {
            /*提示*/
            NoteDebug("文件保存");
            // TODO Auto-generated method stub
            /*获得用户输入的字符串*/
            Text_of_input = inputArt.getText().toString();
            try {
                /*打开文件输出流，名称txtME，以不覆盖模式打开*/
                os = this.openFileOutput("txtME", MODE_PRIVATE);
                /*把字符串转换成字节数组，写入文件中*/
                os.write(Text_of_input.getBytes());
            } catch (FileNotFoundException e) {
                /*文件未找到，异常*/
                // TODO Auto-generated catch block
                NoteDebug("文件关闭失败" + e);
            } catch (IOException e) {
                /*文件写入错误*/
                // TODO Auto-generated catch block
                NoteDebug("文件写入失败" + e);
            } finally {
                try {
                    /*关闭文件输出流*/
                    os.close();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    NoteDebug("文件关闭失败" + e);
                }
            }
            /*输入框清空*/
            inputArt.setText("");
        }
        break;
        case R.id.Button_openTxt: {
            NoteDebug("文件打开");

```

```
try {
    /*打开文件输入流，名称txtME*/
    is = this.openFileInput("txtME");
    /*初始化字节数组*/
    b = new byte[1024];
    /*从文件输入流中读取内容到字节数组中，返回内容长度*/
    int length = is.read(b);
    /*把字节数组转换成字符串*/
    Text_of_output = new String(b);
    /*设置标题，显示文件内容长度*/
    setTitle("文件字数: " + length);
    /*显示文件内容*/
    showmyText.setText(Text_of_output);
} catch (FileNotFoundException e) {
    /*文件未找到，异常*/
    // TODO Auto-generated catch block
    NoteDebug("文件打开失败" + e);
} catch (IOException e) {
    /*文件读取错误，异常*/
    // TODO Auto-generated catch block
    NoteDebug("文件读取失败" + e);
}

finally {
    try {
        /*关闭文件输入流*/
        is.close();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        NoteDebug("文件关闭失败"+e);
    }
}

break;
case R.id.Button_clean:{
    /*清空*/
    showmyText.setText("");
    NoteDebug("清空");
}

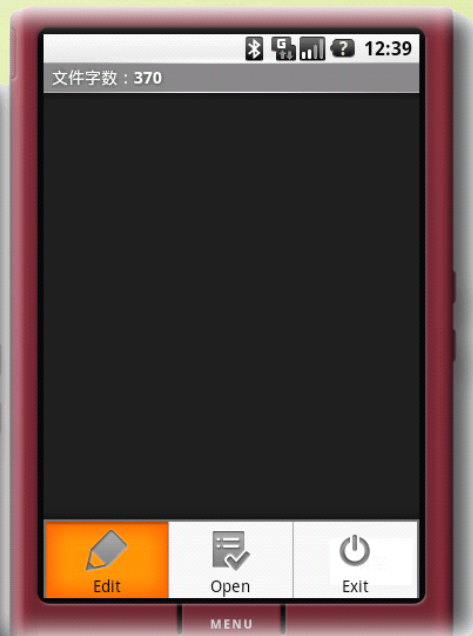
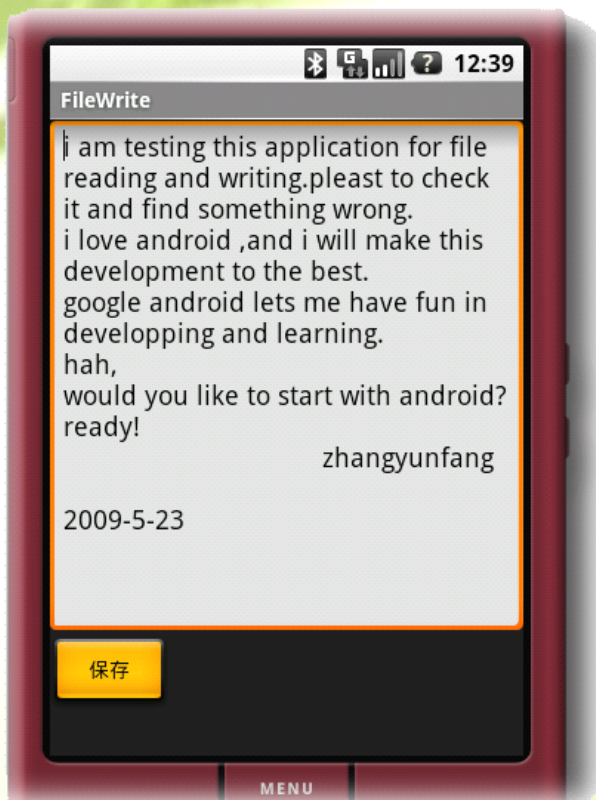
break;
default:
    break;
}
}
```

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // TODO Auto-generated method stub
    /*添加三个菜单项目，并设置图片*/
    menu.add(0, 1, 1, "Edit").setIcon(R.drawable.ic_menu_edit);
    menu.add(0, 2, 2, "Open").setIcon(R.drawable.ic_menu_agenda);
    menu.add(0, 3, 3, "Exit").setIcon(R.drawable.ic_lock_power_off);
    return super.onCreateOptionsMenu(menu);
}
```

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // TODO Auto-generated method stub
    switch (item.getItemId()) {
        case 1:
            /*显示main.xml为主屏布局*/
            setLayoutShow(R.layout.main);
            UIinit("main");
            Logic("main");
            NoteDebug("编辑文件Layout");
            break;
        case 2:
            /*显示open.xml为主屏布局*/
            setLayoutShow(R.layout.open);
            UIinit("open");
            Logic("open");
            NoteDebug("打开文件Layout");
            break;
        case 3:
            /*退出*/
            finish();
            NoteDebug("Byebye");
            break;
        default:
            break;
    }
    return super.onOptionsItemSelected(item);
}

private void NoteDebug(String showString){
    /*显示Toast提示*/
    Toast.makeText(this, showString, Toast.LENGTH_SHORT).show();
}
}
```


⑦ 结果



数据库编程基础

SQLite 数据库

- 在某些情况下，文件不是有效的
 - 如果多线程数据访问是相关的
 - 如果应用程序处理可能变化的复杂数据结构
 - 等等
- 因此，Android 带来了内置 SQLite 数据库支持
- 数据库对于创建它们的包套件是私有的
- 数据库不应该用来存贮文件
- 提示：在 SDK 中的 samples/NotePad 下可以找到关于如何使用数据库的例子
- SQLite 是一个轻量级的软件库
- 实现了一个完全适应严峻环境的数据库
 - 原子量性
 - 坚固性
 - 独立性
 - 耐久性
- 体积大小只用几千字节
- 一些 SQL 的指令只是部分支持，例如：ALTER、TABLE
- 参阅 <http://www.sqlite.org> 获取更多信息



创建数据库

- Context.createDatabase(String name,int version ,int mode,CursorFactory factory)创建一个新的数据库并返回一个 SQLiteDatabase 对象
- 假如数据库不能被创建，则抛出 FileNotFoundException 异常
- 新创建 SQLite 数据库方法

```

SQLiteDatabase mydataBase=SQLiteDatabase.create(new CursorFactory() {
    //创建一个数据库
    //工厂类，一个可选工厂类，当查询时调用来实例化一个光标
    @Override
    public Cursor newCursor(SQLiteDatabase db,
        SQLiteCursorDriver masterQuery, String editTable,
        SQLiteQuery query) {
        // TODO Auto-generated method stub
        return null;
    }

});

```



```

    SQLiteDatabase myDataBase=this.openOrCreateDatabase("myDataBase.db",
                                                    MODE_PRIVATE, new CursorFactory() {
//创建新的数据库，名称myDataBase，模式MODE_PRIVATE，鼠标工厂
//工厂类，一个可选工厂类，当查询时调用来实例化一个光标
        @Override
        public Cursor newCursor(SQLiteDatabase db,
                                SQLiteCursorDriver masterQuery, String editTable,
                                SQLiteQuery query) {
            // TODO Auto-generated method stub
            return null;
        }
    });

```

删除数据库

- Context.deleteDatabase(String name)删除指定名称的数据库
- 假如数据库成功删除则返回 true，失败则为 false(例如数据库不存在)

```

//删除指定名称的数据库
this.deleteDatabase("myDataBase.db");

```

打开数据库

- Context.openDatabase(String file,CursorFactory factory) 打开一个存在的数据库并返回一个 SQLiteDatabase 对象
- 如果数据库不存在则抛出 FileNotFoundException 异常

```

//创建一个名为：myDataBase的数据库，后缀为.db
SQLiteDatabase my_DataBase=this.openOrCreateDatabase("myDateBase.db",
                                                    MODE_PRIVATE, null);

my_DataBase.close(); //不要忘记关闭数据库

```

非查询 SQL 指令

- SQLiteDatabase.execSQL(String sql)可以用来执行非查询 SQL 指令，这些指令没有结果
- 包括：CREATE TABLE / DROP TABLE / INSERT 等等
- 例如：

① 创建一个名为"test"并带两个参数的表

```

//创建一个名为"test"并带两个参数的表
my_DataBase.execSQL("CREATE TABLE test (_id INTEGER PRIMARY KEY,
                                           someNumber INTERGER);");

```

② 在数据库中插入一个元组

//在数据库中插入一个元组

```
my_DataBase.execSQL("INSERT INTO test (_id,someNumber) values(1,8);");
```

③ 删除表

//删除表

```
my_DataBase.execSQL("DROP TABLE test");
```

查询 SQL 指令-游标 Cursors



- Android 使用游标(Cursors)来导航浏览查询结果
- 游标(Cursors)被 android.database.Cursor 对象来描述
- 一个游标(Cursors)是一个简单的指针，它从查询结果的一个元组跳到下一个元组(或是前一个或是第一个或是……)
- 游标(Cursors)在它定位位置的那一刻返回元组数据

表 test

	_id	someNumber
➡	1	8
	2	10
	3	2

//为了创建一个Cursor(游标)，必须执行一个查询，要么通过SQL使用rawQuery() 方法
//或是更精心设计的方法，像query() 方法

```
Cursor cur=my_DataBase.rawQuery("SELECT * FORM test", null);
```

```
if(cur!=null){ //游标不为空
```

```
    //返回给定名称的列的基于0开始的index，如果该属性列不存在则返回-1
```

```
    //通过它们的index来检索属性值
```

```
    int numColumn=cur.getColumnIndex("someNumber");
```

```
    if(cur.moveToFirst()){
```

```
        //cur.moveToFirst() 让游标指向第一行，如果游标指向第一行，则返回true
```

```
        do {
```

```
            int num=cur.getInt(numColumn); //获得当前行该属性的值
```

```
            /*Cursor提供了不同的方法来回索不同的数据类型
```

```
            例如getInt(int index)/getString(int index)等等*/
```

```
            /*做一些事情*/
```

```
        } while (cur.moveToNext());
```

```
        /*游标移动到下一行，如果游标已经通过了结果集中的最后，
```

```
        即没有行可以移动时，则返回false*/
```

```
        //其他可能移动的是 previous() 和first() 方法
```

```
    }
```

```
}
```


android
developers



android

android
developers

Android开发教程&笔记




android

应用

为程序添加 **Menu** 菜单

//创建 OptionsMenu **public boolean onCreateOptionsMenu(Menu menu)**



```
public boolean onCreateOptionsMenu(Menu menu) {  
    // TODO Auto-generated method stub  
    boolean result = super.onCreateOptionsMenu(menu);  
    menu.add(0, INSERT_ID_Play, 0, R.string.menu_toPlay);  
    menu.add(0, INSERT_ID_Stop, 0, R.string.menu_toStop);  
    return result;  
} //创建菜单
```

//处理选择事件 **public boolean onOptionsItemSelected(MenuItem item)**

```
public boolean onOptionsItemSelected(MenuItem item) {  
    // TODO Auto-generated method stub  
    if (item.getItemId() == INSERT_ID_Play) {  
        Play_Music();  
    }  
    if (item.getItemId() == INSERT_ID_Stop) {  
        Stop_Music();  
    }  
    return super.onOptionsItemSelected(item);  
}
```

应用 SDCard

- ① 打开 CMD
- ② 进入 C:\Documents and Settings\地狱怒兽\Local Settings\Application Data\Android\SDK-1.1 目录
- ③ 创建 sdcard 镜像 `mksdcard 256M ./sdcard.img`
- ④ 往 SDCard 中添加资源: `adb push zyf.mp3 /sdcard/zyf.mp3`
- ⑤ 往 SDCard 中获取资源: `adb pull /sdcard/mybaby.jpg C:\`
- ⑥ 重启模拟器后, 文件即在虚拟 SDCard 中



向模拟器安装 APK 软件包

- ① 打开 cmd
- ② 切换到 Android SDK tools 目录下
- ③ 把 APK 软件包复制到 Android SDK tools 目录下
- ④ `adb install Snake.pak`

删除模拟器中 APK 软件包

- ① 打开模拟器
- ② 打开 cmd
- ③ `adb shell`
- ④ `cd data/app`
- ⑤ `ls -l`
- ⑥ `rm 文件名.apk`

对话框的简单应用

AlertDialog.Builder

```
public class DialogDemo extends Activity implements OnClickListener{
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        AlertDialog.Builder myBuilder=new AlertDialog.Builder(this);
        myBuilder.setIcon(R.drawable.hermes);
        myBuilder.setTitle("我的对话框");
        myBuilder.setPositiveButton("退出", this);
        myBuilder.show();
    }
    public void onClick(DialogInterface dialog, int which) {
    }
}
```

Dialog

```
public class DialogDemo extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Dialog myDialog=new Dialog(this);
        myDialog.setTitle("this is a Dialog");
        myDialog.show();
    }
}
```

排版布局 Layout

元件名稱	說明
FrameLayout	單一物件的容器
AbsoluteLayout	以絕對座標排版的容器
LinearLayout	線性 (水平或垂直) 排版的容器
RelativeLayout	以相對座標 (相對於父元件或兄弟元件) 排版的容器
TableLayout	以表格方式排版的容器

TextView 中的超链接文本

```
<TextView
    android:id="@+id/linkText"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/linktext"
    android:autoLink="all"
>
</TextView>
```

关键点：设置了该属性则可以自动链接，否则没有链接



时间设置对话框 **DatePickerDialog** 的使用

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Button btn = (Button) findViewById(R.id.date);
    btn.setOnClickListener(this);
}

@Override
public void onClick(View v) { //普通按钮事件
    Calendar d = Calendar.getInstance(Locale.CHINA);
    //创建一个日历引用d, 通过静态方法getInstance() 从指定时区 Locale.CHINA 获得一个日期实例
    Date myDate=new Date();
    //创建一个Date实例
    d.setTime(myDate);
    //设置日历的时间, 把一个新建Date实例myDate传入
    int year=d.get(Calendar.YEAR);
    int month=d.get(Calendar.MONTH);
    int day=d.get(Calendar.DAY_OF_MONTH);
    //获得日历中的 year month day
    DatePickerDialog dlg=new DatePickerDialog(this, this, year, month, day);
    //新建一个DatePickerDialog 构造方法中
    // (设备上下文, OnDateSetListener时间设置监听器, 默认年, 默认月, 默认日)
    dlg.show();
    //让DatePickerDialog显示出来
}

@Override
public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth) {
    //DatePickerDialog 中按钮Set按下时自动调用
    TextView txt = (TextView) findViewById(R.id.text);
    //通过id获得TextView对象
    txt.setText(Integer.toString(year) + "-" +
                Integer.toString(monthOfYear) + "-" +
                Integer.toString(dayOfMonth));
    //设置text
}
```

ListView 的使用 1

① ListView 的声明、定义

```
ListView list=new ListView(this);
```

② 数组适配器的生命定义

```
String []name=new String[]{"Java","C++","C","C#","VB","XML",".NET","J#"};
ArrayAdapter<String> arrayadapter
=new ArrayAdapter<String>(this,
                        android.R.layout.simple_list_item_1,name);
```

③ 给 ListView 设置数组适配器

```
list.setAdapter(arrayadapter);
```

④ 设置 ListView 的元素被选中时的事件处理监听器

```
list.setOnItemClickListener(this);
```

⑤ 事件处理监听器方法

```
@Override
public void onItemClick(AdapterView<?> arg0,
                        View arg1, int arg2, long arg3) {
    // TODO Auto-generated method stub
    setTitle(GetData(arg2));
}
```

⑥ 让 ListView 中的内容变化，重新设置一个 ArrayAdapter 即可

```
String []score=new String[]{"93","76","84","85","93","84","88","78"};
ArrayAdapter<String> arratadapter2=new ArrayAdapter<String>(this,
                        android.R.layout.simple_expandable_list_item_1,score);
list.setAdapter(arratadapter2);
```


ListView 的使用 2

① ListView 在 XML 文件中声明

```
<ListView android:id="@+id/list"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
/>
```

② ListView 在 java 文件中获得

```
listview = (ListView) findViewById(R.id.list);
//从 list.xml 中获得 ListView 对象 list
```

③ 声明和定义、初始化一个 ArrayList

```
private ArrayList<Map<String, Object>> arrayList;
arrayList= new ArrayList<Map<String, Object>>();

Map<String, Object> map;
    map= new HashMap<String, Object>();
    map.put("System", "Linux");
    map.put("Ability", "Cool");
    arrayList.add(item);
```

④ 声明和定义、初始化一个 SimpleAdepter

```
SimpleAdapter adapter;
adapter= new SimpleAdapter(this, coll,
    android.R.layout.simple_list_item_1,
    new String[] { "System" }, new int[] { android.R.id.text1 });
```

⑤ ListView 设置适配器

```
listview.setAdapter(adapter); //设置适配器
```

⑥ 为 ListView 设置选项事件监听器

```
listview.setOnItemClickListener(listener);

OnItemClickListener listener = new OnItemClickListener() {
    public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
        long arg3) {
        textview.setTextColor(Color.YELLOW);
        textview.setText(coll.get(arg2).get("prod_type").toString());
    }
}; //事件处理接口实现
```



ListActivity 的使用

① 定义一个类 继承自 ListActivity

```
public class ListActivityOfMe extends ListActivity {}
```

② 声明和定义、初始化一个 ArrayList

```
private ArrayList<Map<String, Object>> arrayList;  
arrayList= new ArrayList<Map<String, Object>>();
```

```
Map<String, Object> map;  
map= new HashMap<String, Object>();  
map.put("System", "Linux");  
map.put("Ability", "Cool");  
arrayList.add(item);
```

③ 声明和定义、初始化一个 SimpleAdepter

```
SimpleAdapter adapter;  
adapter= new SimpleAdapter(this, arrayList,  
    android.R.layout.simple_list_item_1,  
    new String[] { "System" }, new int[] { android.R.id.text1 });
```

④ ListView 设置适配器

```
this.setAdapter(adapter); //设置适配器
```

⑤ 选项按键事件处理, 不需要.setOnItemClickListener(); 直接实现以下方法

```
protected void onItemClick(ListView l, View v, int position, long id)  
{  
    // TODO Auto-generated method stub  
    super.onItemClick(l, v, position, id);  
    actionTo(position);  
}
```


数据共享 SharedPreferences

- ① 数据请求端，线先开启另一个 Activity

```
Intent startNextLayout=new Intent();
startNextLayout.setClass(myEx.this,choiceGroupActivity.class);
startActivity(startNextLayout);
//开启另一个 Activity
```

- ② 数据选择端，设置好数据，利用实现了 SharedPreferences 接口的 Editor 子类添加并修改数据

```
Editor passfileEditor=getSharedPreferences("ITEM", 0).edit();
passfileEditor.putString("ChoiceKye", myContentString);
//设置数据 Key，并添加数据 myContentString 到共享区
```

获得实现了 SharedPreferences 接口的 Editor 对象

- ③ 数据选择端，存入数据，关闭 Activity。

```
passfileEditor.commit();//委托，存入数据
finish();//
```

- ④ 数据请求端，利用 SharedPreferences 接口来获取数据 Key，然后通过 getString(String Key, String Defaultvalue)来获得 Key 中数据

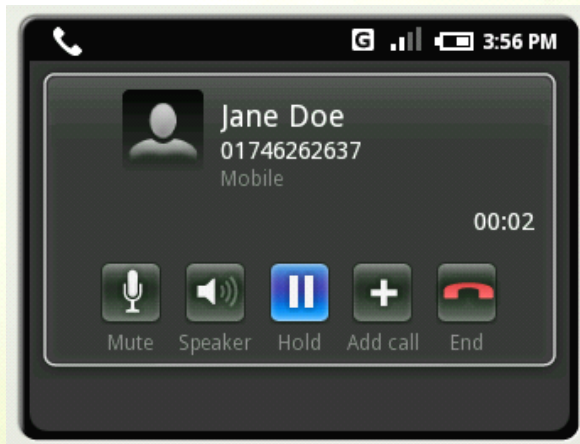
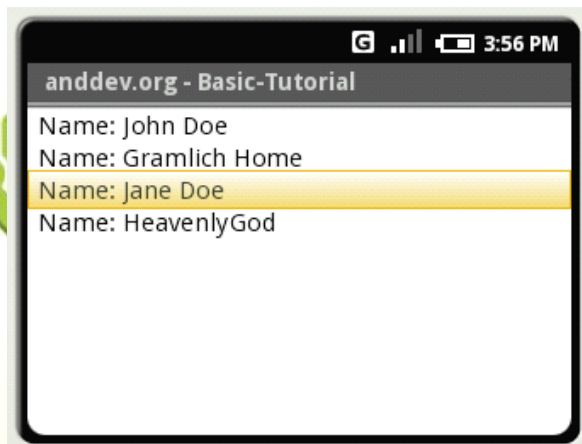
```
SharedPreferences sharedPreferences=getSharedPreferences("ITEM", 0);
//得到共享区"ITEM"的接口引用
String show=sharedPreferences.getString("ChoiceKye", "there is NONE")
/*利用Key"ChoiceKye"从SharedPreferences 接口引用中
得到ChoiceKye 的内容，没有时为默认值"there is NONE"*/
TextView showTextView=(TextView)findViewById(R.id.showR);
showTextView.setText(show);//显示 ChoiceKye 的内容
```

Intents 和 ListActivity:

创建联系人列表并和联系人打电话应用程序

内容: 这个教程教我们怎么开发一个 ListActivity 应用, 联系人的列表装载到了基于 List 的 View. 然后可以选择其中之一打电话.

目的: 可以非常容易的学会 Intents 和 ListActivities 的使用.



Main.xml

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


AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="zyf.CallME"
    android:versionCode="1"
    android:versionName="1.0">
    <application
        android:label="@string/app_name"
        android:icon="@drawable/android">
        <activity
            android:name=".CallME"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="
                    android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="2" />
    <uses-permission android:name="android.permission.READ_CONTACTS">
    </uses-permission>
    <uses-permission android:name="android.permission.CALL_PHONE"></uses-
permission>
</manifest>
```

Java

```
package zyf.CallME;
import android.app.ListActivity;
import android.content.*;
import android.database.Cursor;
import android.os.Bundle;
import android.provider.Contacts.People;
import android.view.View;
import android.widget.*;
public class CallME extends ListActivity {
    private String[] name;
    private int[] to;
    private SimpleCursorAdapter sadapter;
    private Intent callIntent;
    private long PhoneID;
    @Override
    public void onCreate(Bundle icle) {
        super.onCreate(icle);
        try {
            Cursor c =
                getResolver().query(People.CONTENT_URI, null,
                    null, null, null);
            startManagingCursor(c);
            name = new String[] { People.NAME };
            to = new int[] { R.id.row_entry };
            sadapter = new SimpleCursorAdapter(this,R.layout.main, c, name, to);
            this.setListAdapter(sadapter);
        } catch (Exception e) {
            Toast.makeText(this, "联系人读取错误",
                Toast.LENGTH_LONG).show();
        }
    }
    @Override
    protected void onItemClick(ListView l,
        View v, int position, long id) {
        super.onItemClick(l, v, position, id);
        callIntent=new Intent(Intent.ACTION_CALL);
        Cursor c=(Cursor) sadapter.getItem(position);
        PhoneID=c.getLong(c.getColumnIndex(People.PRIMARY_PHONE_ID));
        callIntent.setData(ContentUris.withAppendedId(android.provider.
            Contacts.Phones.CONTENT_URI, PhoneID));
        startActivity(callIntent);
    }
}
```


解析

- ① 在 main.xml 中添加两个 TextView

用于显示"Name: "的静态标签

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Name: "
/>
```

用于动态添加并显示联系人的姓名内容的 TextView

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

- ② 修改 LinearLayout 的布局方向

```
android:orientation="horizontal" //说明横向排布
```

- ③ AndroidManifest.xml 中说明使用权限

```
<uses-permission
    android:name="android.permission.READ_CONTACTS"></uses-permission>
//说明可以访问读取联系人信息

<uses-permission
    android:name="android.permission.CALL_PHONE"></uses-permission>
//说明可以使用打电话功能
```

- ④ Java 代码中的读取联系人信息处理

```
try {
    Cursor c =
    getContentResolver().query(People.CONTENT_URI, null, null, null, null);
    //Cursor, 该接口提供从数据库提取信息返回结果中随机读写访问
    //ContentResolver 提供应用程序访问Content模型
    //ContentResolver.query() 抽取给定的URI, 返回一个在结果之上的Cursor
    startManagingCursor(c);
    //该方法允许Activity基于Activity的生命周期来为你处理管理给定的Cursor的生命周期
    name = new String[] { People.NAME };
    //新建用户名数组 People类用于列出联系人 People.NAME得到联系人姓名
```

```

to = new int[] { R.id.row_entry };
//创建一个TextView引用数组 用来放置获取的 People.NAME
sadapter = new SimpleCursorAdapter(this,R.layout.main, c, name, to);
    this.setListAdapter(sadapter);
//创建一个简单的适配器从一个cursor指针到TextView或是ImageView的map专栏适配器。
//构造方法 (Context,layout,Cursor,from,to), 第一参数是设备上下文, 第二个参数是布局文件, 第三个参数是指向联系人URI的Cursor指针, form代表来源的字符串 (联系人名), to把联系人名放到的地方 (TextView)
} catch (Exception e) {
    Toast.makeText(this, "联系人读取错误",Toast.LENGTH_LONG).show();
    //Toast显示提示, 不打扰用户。.show() 用来显示Toast
}

```

⑤ Java 代码中的点击事件处理(选择后电话联系该联系人)

```

protected void onItemClick(ListView l,View v, int position, long id) {
    // TODO Auto-generated method stub
    super.onItemClick(l, v, position, id);
    callIntent=new Intent(Intent.ACTION_CALL);
    //创建一个带有Call动作的Intent
    Cursor c=(Cursor) sadapter.getItem(position);
    //通过选中的Items位置来获取相应的联系人指针 Cursor
    PhoneID=c.getLong(c.getColumnIndex(People.PRIMARY_PHONE_ID));
    //获取相应联系人电话号码
    //getLong() 返回请求数据的一个长整型
    //为给定的列, 返回基于0的索引值
    //People.PRIMARY_PHONE_ID 获取主键电话号码
    callIntent.setData(ContentUris.withAppendedId(android.provider.
                                                Contacts.Phones.CONTENT_URI, PhoneID));

    //为Intent设置操作的数据
    //ContentUris操作带有数据内容的Uri的实用方法、它们带有"Content"体制
    //withAppendedId() 把给定的ID添加到path后面 第一个参数是开始的, 后面参数是添加的
    startActivity(callIntent);
    //开启Intent
}

```


Android Permission 大全

Android SDK 1.0 访问权限许可

程序执行需要读取到安全敏感项必需在 `androidmanifest.xml` 中声明相关权限请求, 完整列表如下:

`android.permission.ACCESS_CHECKIN_PROPERTIES`

允许读写访问"properties"表在 checkin 数据库中, 改值可以修改上传(Allows read/write access to the "properties" table in the checkin database, to change values that get uploaded)

`android.permission.ACCESS_COARSE_LOCATION`

允许一个程序访问 CellID 或 WiFi 热点来获取粗略的位置(Allows an application to access coarse (e.g., Cell-ID, WiFi) location)

`android.permission.ACCESS_FINE_LOCATION`

允许一个程序访问精良位置(如 GPS) (Allows an application to access fine (e.g., GPS) location)

`android.permission.ACCESS_LOCATION_EXTRA_COMMANDS`

允许应用程序访问额外的位置提供命令(Allows an application to access extra location provider commands)

`android.permission.ACCESS_MOCK_LOCATION`

允许程序创建模拟位置提供用于测试(Allows an application to create mock location providers for testing)

`android.permission.ACCESS_NETWORK_STATE`

允许程序访问有关 GSM 网络信息(Allows applications to access information about networks)

`android.permission.ACCESS_SURFACE_FLINGER`

允许程序使用 SurfaceFlinger 底层特性(Allows an application to use SurfaceFlinger's low level features)

`android.permission.ACCESS_WIFI_STATE`

允许程序访问 Wi-Fi 网络状态信息(Allows applications to access information about Wi-Fi networks)

`android.permission.ADD_SYSTEM_SERVICE`

允许程序发布系统级服务(Allows an application to publish system-level services).

`android.permission.BATTERY_STATS`

允许程序更新手机电池统计信息(Allows an application to update the collected battery statistics)

`android.permission.BLUETOOTH`

允许程序连接到已配对的蓝牙设备(Allows applications to connect to paired bluetooth devices)

`android.permission.BLUETOOTH_ADMIN`

允许程序发现和配对蓝牙设备(Allows applications to discover and pair bluetooth devices)

`android.permission.BRICK`

请求能够禁用设备(非常危险)(Required to be able to disable the device (very *erous!).)

`android.permission.BROADCAST_PACKAGE_REMOVED`

允许程序广播一个提示消息在一个应用程序包已经移除后(Allows an application to broadcast a notification that an application package has been removed)

`android.permission.BROADCAST_STICKY`

允许一个程序广播常用 intents(Allows an application to broadcast sticky intents)



`android.permission.CALL_PHONE`

允许一个程序初始化一个电话拨号不需通过拨号用户界面需要用户确认(Allows an application to initiate a phone call without going through the Dialer user interface for the user to confirm the call being placed.)

`android.permission.CALL_PRIVILEGED`

允许一个程序拨打任何号码，包含紧急号码无需通过拨号用户界面需要用户确认(Allows an application to call any phone number, including emergency numbers, without going through the Dialer user interface for the user to confirm the call being placed)



`android.permission.CAMERA`

请求访问使用照相设备(Required to be able to access the camera device.)

`android.permission.CHANGE_COMPONENT_ENABLED_STATE`

允许一个程序是否改变一个组件或其他的启用或禁用(Allows an application to change whether an application component (other than its own) is enabled or not.)

`android.permission.CHANGE_CONFIGURATION`

允许一个程序修改当前设置，如本地化(Allows an application to modify the current configuration, such as locale.)

`android.permission.CHANGE_NETWORK_STATE`

允许程序改变网络连接状态(Allows applications to change network connectivity state)

`android.permission.CHANGE_WIFI_STATE`

允许程序改变 Wi-Fi 连接状态(Allows applications to change Wi-Fi connectivity state)

`android.permission.CLEAR_APP_CACHE`

允许一个程序清楚缓存从所有安装的程序在设备中(Allows an application to clear the caches of all installed applications on the device.)

`android.permission.CLEAR_APP_USER_DATA`

允许一个程序清除用户设置(Allows an application to clear user data)

`android.permission.CONTROL_LOCATION_UPDATES`

允许启用禁止位置更新提示从无线模块(Allows enabling/disabling location update notifications from the radio.)

`android.permission.DELETE_CACHE_FILES`

允许程序删除缓存文件(Allows an application to delete cache files)

`android.permission.DELETE_PACKAGES`

允许一个程序删除包(Allows an application to delete packages)

`android.permission.DEVICE_POWER`

允许访问底层电源管理(Allows low-level access to power management)

`android.permission.DIAGNOSTIC`

允许程序 RW 诊断资源(Allows applications to RW to diagnostic resources.)

`android.permission.DISABLE_KEYGUARD`

允许程序禁用键盘锁(Allows applications to disable the keyguard)

`android.permission.DUMP`

允许程序返回状态抓取信息从系统服务(Allows an application to retrieve state dump information from system services.)



`android.permission.EXPAND_STATUS_BAR`

允许一个程序扩展收缩在状态栏,android 开发网提示应该是一个类似 Windows Mobile 中的托盘程序(Allows an application to expand or collapse the status bar.)

`android.permission.FACTORY_TEST`

作为一个工厂测试程序, 运行在 root 用户(Run as a manufacturer test application, running as the root user.)

`android.permission.FLASHLIGHT`

访问闪光灯,android 开发网提示 HTC Dream 不包含闪光灯(Allows access to the flashlight)

`android.permission.FORCE_BACK`

允许程序强行一个后退操作是否在顶层 activities(Allows an application to force a BACK operation on whatever is the top activity.)

`android.permission.FOTA_UPDATE`

暂时不了解这是做什么使用的, android 开发网分析可能是一个预留权限.

`android.permission.GET_ACCOUNTS`

访问一个帐户列表在 Accounts Service 中(Allows access to the list of accounts in the Accounts Service)

`android.permission.GET_PACKAGE_SIZE`

允许一个程序获取任何 package 占用空间容量(Allows an application to find out the space used by any package.)

`android.permission.GET_TASKS`

允许一个程序获取信息有关当前或最近运行的任务, 一个缩略的任务状态, 是否活动等等(Allows an application to get information about the currently or recently running tasks: a thumbnail representation of the tasks, what activities are running in it, etc.)



`android.permission.HARDWARE_TEST`

允许访问硬件(Allows access to hardware peripherals.)

`android.permission.INJECT_EVENTS`

允许一个程序截获用户事件如按键、触摸、轨迹球等等到一个时间流，android 开发网提醒算是 hook 技术吧 (Allows an application to inject user events (keys, touch, trackball) into the event stream and deliver them to ANY window.)



`android.permission.INSTALL_PACKAGES`

允许一个程序安装 packages(Allows an application to install packages.)

`android.permission.INTERNAL_SYSTEM_WINDOW`

允许打开窗口使用系统用户界面(Allows an application to open windows that are for use by parts of the system user interface.)

`android.permission.INTERNET`

允许程序打开网络套接字(Allows applications to open network sockets)

`android.permission.MANAGE_APP_TOKENS`

允许程序管理(创建、催后、z- order 默认向 z 轴推移)程序引用在窗口管理器中(Allows an application to manage (create, destroy, Z-order) application tokens in the window manager.)

`android.permission.MASTER_CLEAR`

目前还没有明确的解释，android 开发网分析可能是清除一切数据，类似硬格机

`android.permission.MODIFY_AUDIO_SETTINGS`

允许程序修改全局音频设置(Allows an application to modify global audio settings)

`android.permission.MODIFY_PHONE_STATE`

允许修改话机状态，如电源，人机接口等(Allows modification of the telephony state - power on, mmi, etc.)

`android.permission.MOUNT_UNMOUNT_FILESYSTEMS`

允许挂载和反挂载文件系统可移动存储(Allows mounting and unmounting file systems for removable storage.)

`android.permission.PERSISTENT_ACTIVITY`

允许一个程序设置他的 activities 显示(Allow an application to make its activities persistent.)

`android.permission.PROCESS_OUTGOING_CALLS`

允许程序监视、修改有关播出电话(Allows an application to monitor, modify, or abort outgoing calls)

`android.permission.READ_CALENDAR`

允许程序读取用户日历数据(Allows an application to read the user's calendar data.)

`android.permission.READ_CONTACTS`

允许程序读取用户联系人数据(Allows an application to read the user's contacts data.)

`android.permission.READ_FRAME_BUFFER`

允许程序屏幕波或和更多常规的访问帧缓冲数据(Allows an application to take screen shots and more generally get access to the frame buffer data)

`android.permission.READ_INPUT_STATE`

允许程序返回当前按键状态(Allows an application to retrieve the current state of keys and switches.)

`android.permission.READ_LOGS`

允许程序读取底层系统日志文件(Allows an application to read the low-level system log files.)



`android.permission.READ_OWNER_DATA`

允许程序读取所有者数据(Allows an application to read the owner's data)

`android.permission.READ_SMS`

允许程序读取短信息(Allows an application to read SMS messages.)

`android.permission.READ_SYNC_SETTINGS`

允许程序读取同步设置(Allows applications to read the sync settings)

`android.permission.READ_SYNC_STATS`

允许程序读取同步状态(Allows applications to read the sync stats)

`android.permission.REBOOT`

请求能够重新启动设备(Required to be able to reboot the device.)

`android.permission.RECEIVE_BOOT_COMPLETED`

允许一个程序接收到 `ACTION_BOOT_COMPLETED` 广播在系统完成启动(Allows an application to receive the `ACTION_BOOT_COMPLETED` that is broadcast after the system finishes booting.)

`android.permission.RECEIVE_MMS`

允许一个程序监控将收到 MMS 彩信,记录或处理(Allows an application to monitor incoming MMS messages, to record or perform processing on them.)

`android.permission.RECEIVE_SMS`

允许程序监控一个将收到短信息, 记录或处理(Allows an application to monitor incoming SMS messages, to record or perform processing on them.)



`android.permission.RECEIVE_WAP_PUSH`

允许程序监控将收到 WAP PUSH 信息(Allows an application to monitor incoming WAP push messages.)

`android.permission.RECORD_AUDIO`

允许程序录制音频(Allows an application to record audio)

`android.permission.REORDER_TASKS`

允许程序改变 Z 轴排列任务(Allows an application to change the Z-order of tasks)

`android.permission.RESTART_PACKAGES`

允许程序重新启动其他程序(Allows an application to restart other applications)

`android.permission.SEND_SMS`

允许程序发送 SMS 短信(Allows an application to send SMS messages)

`android.permission.SET_ACTIVITY_WATCHER`

允许程序监控或控制 activities 已经启动全局系统中 Allows an application to watch and control how activities are started globally in the system.

`android.permission.SET_ALWAYS_FINISH`

允许程序控制是否活动间接完成在处于后台时 Allows an application to control whether activities are immediately finished when put in the background.

`android.permission.SET_ANIMATION_SCALE`

修改全局信息比例(Modify the global animation scaling factor.)



`android.permission.SET_DEBUG_APP`

配置一个程序用于调试(Configure an application for debugging.)

`android.permission.SET_ORIENTATION`

允许底层访问设置屏幕方向和实际旋转(Allows low-level access to setting the orientation (actually rotation) of the screen.)

`android.permission.SET_PREFERRED_APPLICATIONS`

允许一个程序修改列表参数

`PackageManager.addPackageToPreferred()` 和 `PackageManager.removePackageFromPreferred()` 方法 (Allows an application to modify the list of preferred applications with the `PackageManager.addPackageToPreferred()` and `PackageManager.removePackageFromPreferred()` methods.)

`android.permission.SET_PROCESS_FOREGROUND`

允许程序当前运行程序强行到前台(Allows an application to force any currently running process to be in the foreground.)

`android.permission.SET_PROCESS_LIMIT`

允许设置最大的运行进程数量(Allows an application to set the maximum number of (not needed) application processes that can be running.)

`android.permission.SET_TIME_ZONE`

允许程序设置时间区域(Allows applications to set the system time zone)

`android.permission.SET_WALLPAPER`

允许程序设置壁纸(Allows applications to set the wallpaper)

`android.permission.SET_WALLPAPER_HINTS`

允许程序设置壁纸 hints(Allows applications to set the wallpaper hints)

`android.permission.SIGNAL_PERSISTENT_PROCESSES`

允许程序请求发送信号到所有显示的进程中(Allow an application to request that a signal be sent to all persistent processes)

`android.permission.STATUS_BAR`

允许程序打开、关闭或禁用状态栏及图标 Allows an application to open, close, or disable the status bar and its icons.

`android.permission.SUBSCRIBED_FEEDS_READ`

允许一个程序访问订阅 RSS Feed 内容提供 (Allows an application to allow access the subscribed feeds ContentProvider.)

`android.permission.SUBSCRIBED_FEEDS_WRITE`

系统暂时保留改设置,android 开发网认为未来版本会加入该功能。

`android.permission.SYSTEM_ALERT_WINDOW`

允许一个程序打开窗口使用 TYPE_SYSTEM_ALERT, 显示在其他所有程序的顶层(Allows an application to open windows using the type TYPE_SYSTEM_ALERT, shown on top of all other applications.)

`android.permission.VIBRATE`

允许访问振动设备(Allows access to the vibrator)

`android.permission.WAKE_LOCK`

允许使用 PowerManager 的 WakeLocks 保持进程在休眠时从屏幕消失(Allows using PowerManager WakeLocks to keep processor from sleeping or screen from dimming)



`android.permission.WRITE_APN_SETTINGS`

允许程序写入 API 设置(Allows applications to write the apn settings)

`android.permission.WRITE_CALENDAR`

允许一个程序写入但不读取用户日历数据(Allows an application to write (but not read) the user's calendar data.)

`android.permission.WRITE_CONTACTS`

允许程序写入但不读取用户联系人数据(Allows an application to write (but not read) the user's contacts data.)

`android.permission.WRITE_GSERVICES`

允许程序修改 Google 服务地图(Allows an application to modify the Google service map.)

`android.permission.WRITE_OWNER_DATA`

允许一个程序写入但不读取所有者数据(Allows an application to write (but not read) the owner's data.)

`android.permission.WRITE_SETTINGS`

允许程序读取或写入系统设置(Allows an application to read or write the system settings.)

`android.permission.WRITE_SMS`

允许程序写短信(Allows an application to write SMS messages)

`android.permission.WRITE_SYNC_SETTINGS`

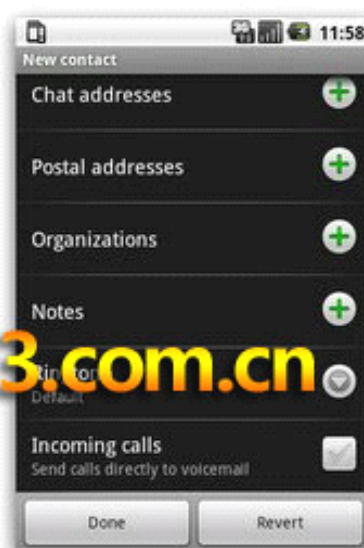
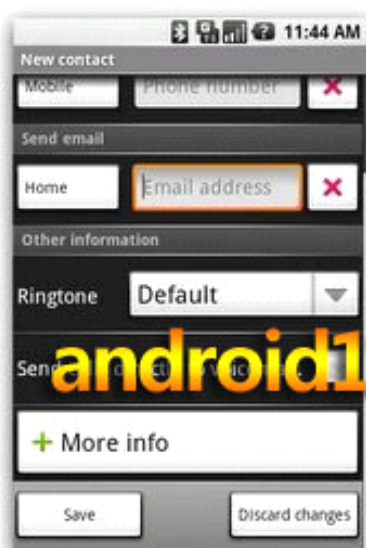
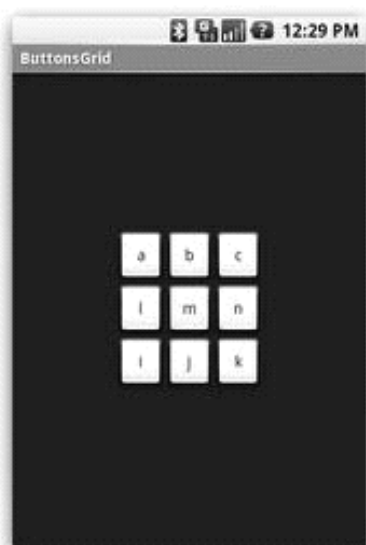
允许程序写入同步设置(Allows applications to write the sync settings)



Android SDK1.5 新视角

用户界面的改进

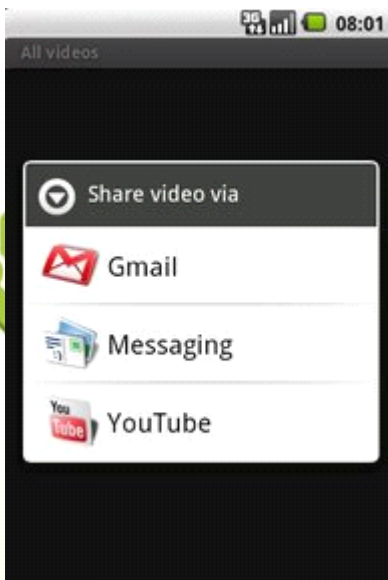
Android SDK1.5 的 UI 做了很多细节的改进，Google 微调了所有核心 UI 组件的样式，使其更加美观，并且对包括 Browser, Gmail, Email, Calendar, Camera&Gallery 以及 Contacts 等在内的自带程序界面都作了优化。让我们来看一个例子，这是新建联系人的界面，左侧是来自 Android 1.1 的截图，右侧是来自 1.5 的截图（此图来自 Google Android 官方 Blog）。我们可以看到下拉选择框和按钮的样式发生了不小的变化，而且添加联系人详情的方式也更加简便了，1.1 中的 More Info 按钮不见了，取而代之的是每一个项目后面的添加和删除按钮。有兴趣的朋友可以打开 1.1 和 1.5 的模拟器来比较一下两者在操作上的差异。



全新的视屏录制功能

上传视频到 Youtube，上传照片到 Picasa

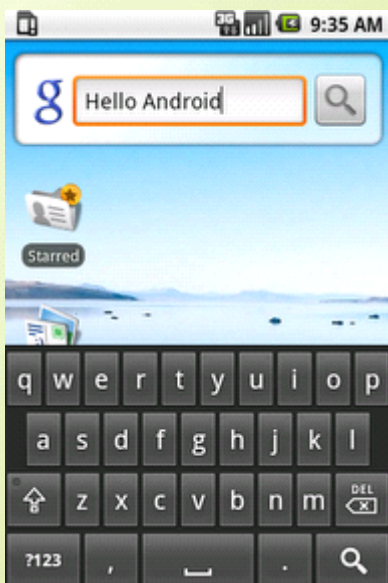
Android 1.5 中，用户可以利用内置的摄像头来拍摄视频，并且立即上传到 Youtube 与朋友分享（尽管目前看来 G1 的 3M 摄像头效果不太理想，希望 G2 的 5M 摄像头效果会好一些）。同样，用户所拍的照片也将可以方便的上传到 Picasa 相册。



当用户完成拍摄/拍照后，只需按一下“Share”按钮，Android 便会让你选择 Share 的方式，只要选择 Youtube/Picasa，然后输入你的账户信息，就可以开始上传了。

软键盘的支持

G1 带有一个 QWERTY 物理键盘，而 G2 将取消全键盘而改用全触摸的设计，所以软键盘也是 Android 1.5 中一个非常重要的特性。下面我们来看一张 Android 1.5 软键盘的截图吧，键盘的布局还是标准的 QWERTY，有一个 Shift 键以及输入模式切换键，进入数字模式以后，可以输入数字以及+*/?!等常用符号，按 ALT 就可以输入一些不太常用的符号。



中文显示和中文输入的支持

Android 1.1 除了英语以外，只能支持德语，而 Android 1.5 的国际化有了更进一步的发展，支持了包括简体/繁体中文、日语、朝鲜语、西班牙语、法语、意大利语、荷兰语、波兰语、俄语在内的各国语言（大概主要语种就差葡萄牙语了吧，Correct me if I’ m wrong :)）。当然我最关心的就是对于中文显示和输入的支持了。启动模拟器，载入 1.5 的 Image 后，默认语言依旧是英语，当我在 Settings ->Locale & text 中把 Locale 改成 Chinese(China)之后，界面语言就切换到中文了。



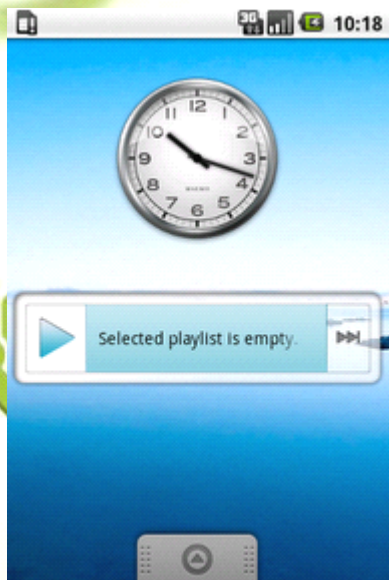
不过这时候，Google 拼音输入法却并不工作，我们调出软键盘以后还是看不到切换到中文的按键，尽管在 Locale & text 设置中，它是被选中的。我们需要反选“Android 键盘”这一选项，然后退出设置菜单，这时 Google 拼音输入法就可以工作了。



关于输入的另一个值得注意的特性是，得益于 IMF 我们可以安装第 3 方的输入法来取代 Google 拼音，也许不久以后就会有 Android 版的搜狗拼音、五笔加加。

桌面 Widgets 和 Live folders

首先，相对于 Android 1.1 来说，自带的桌面 Widget 增加了 Calendar 和 Music player，Music player widget 的样式看起来还不错。



至于 Live folder，自带的包括列出所有联系人，列出所有有电话号码的联系人，以及列出 Starred 联系人，因为缺少可定制性，这个功能现在看来似乎并不怎么吸引人。不过新增的 Home screen widgets API 和 Live folders API 使得开发者可以编写各种新奇的 Widget 和 Live Folder 来丰富 Android 的应用。

蓝牙功能改进

Android 1.5 的蓝牙增加了 A2DP(Advanced Audio Distribution Profile)和 AVRCP (Audio Video Remote Control Profile) 的支持，立体声显著提升了通过蓝牙听音乐的用户体验，AVRCP 则允许用户通过蓝牙进行包括暂停、停止、启动重放、音量控制及其它类型的远程控制操作。另外，Auto-pairing 则使耳机与手机连接更加方便快捷。

捆绑应用功能改进

Android 1.5 内置的 Chrome lite 浏览器采用了最新的 Webkit 引擎和 Squirrelfish Javascript 引擎，并且开始支持网页内容复制和页内搜索功能，窗口下方的 Zoom In/Out 菜单也有变化，同时用户可以指定网页默认编码。我比较感兴趣的就是网页内容复制和页内搜索了，看一下截图吧。



Gtalk、Contacts、SMS、MMS、Gmail 和 Email 程序直接的集成也更加紧密，使得用户可以在 Contacts、SMS、MMS、Gmail 和 Email 中查看 Gtalk 联系人的状态等

Gmail 客户端软件中，批量的存档、删除、加 Label 操作也已经得到支持。

另外，Android 1.5 还对 SD 卡的管理做了改进，增加了文件系统检查和自动修复功能。

系统性能优化

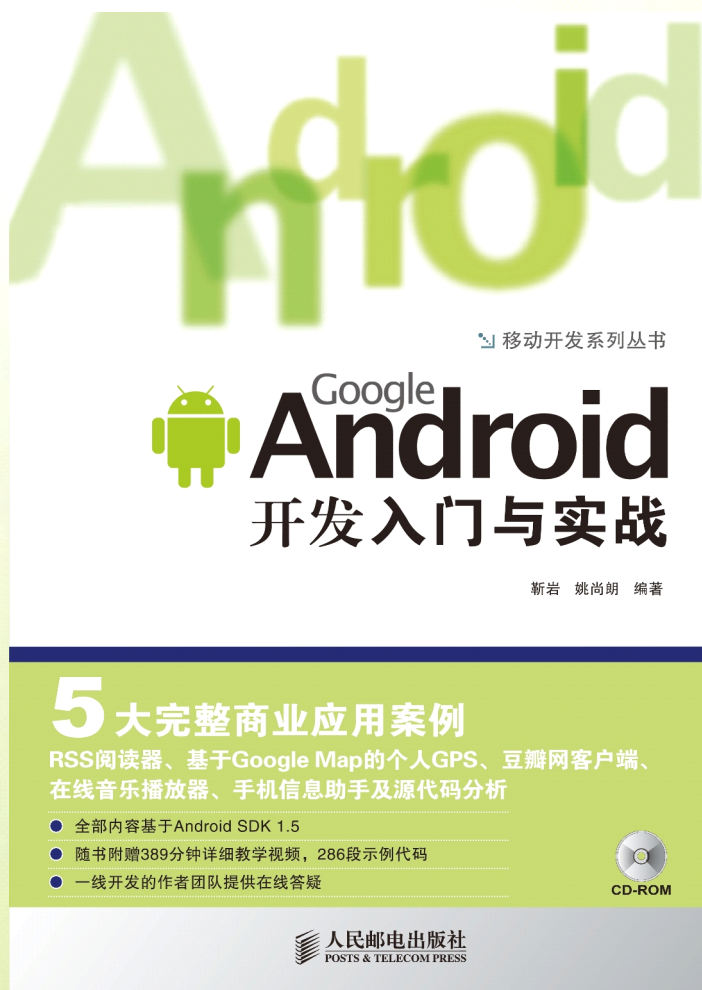
除了上面提到的新功能，Android 1.5 还有不少性能上的改进包括：

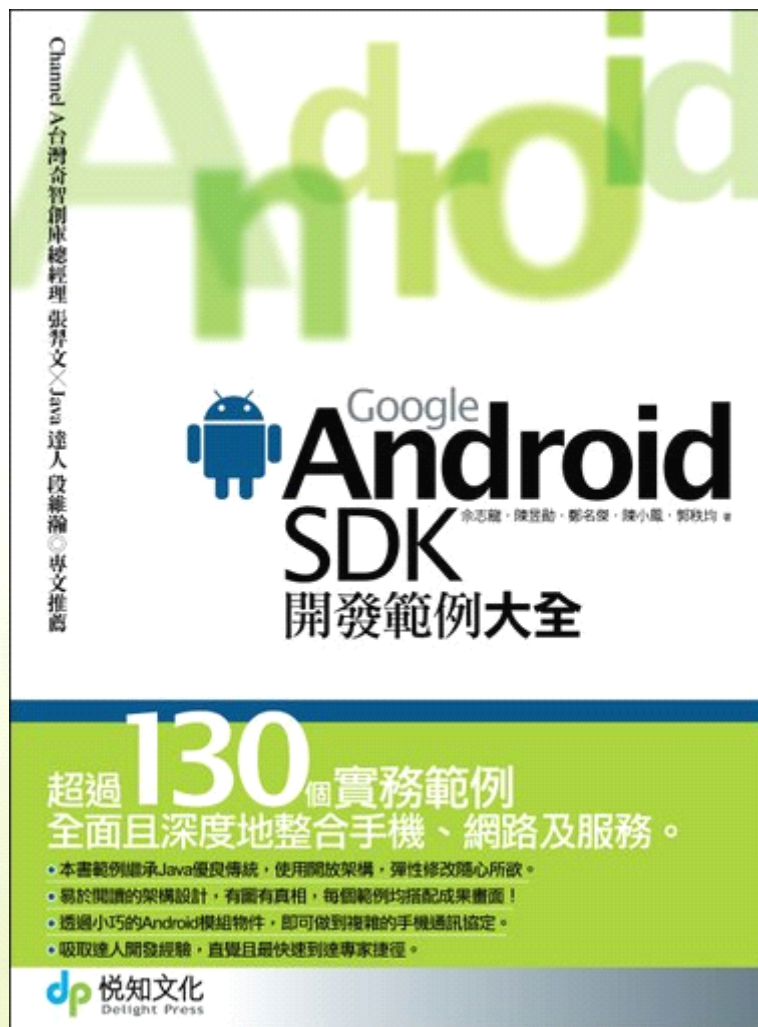
- ◆ * 更快的摄像头启动和拍摄速度
- ◆ * 更快地获取 GPS 位置
- ◆ * 浏览器中更平滑的滚屏
- ◆ * Gmail 中更快的对话列表滚屏等

Android 图书大全

Android 中文教程图书

Google Android 开发入门与实战

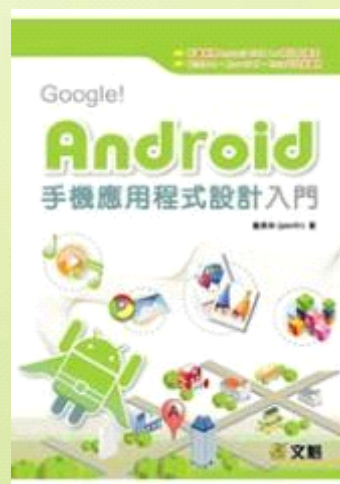




Google Android 程式设计与应用 / 杨文志



Google! Android 手机应用程式设计入门 / 盖索林



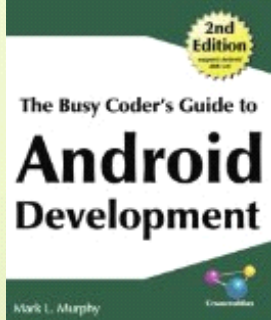
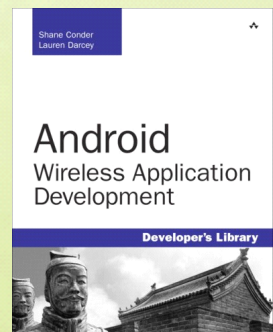
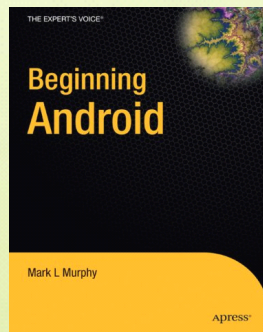
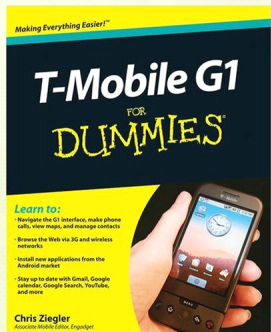
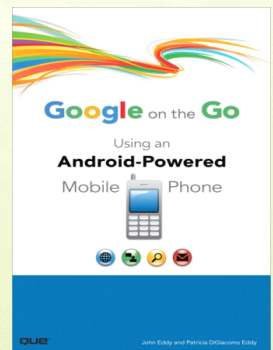
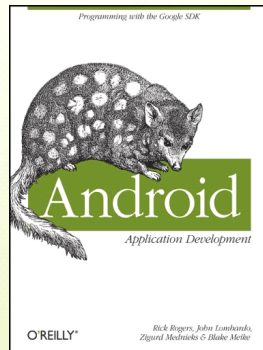
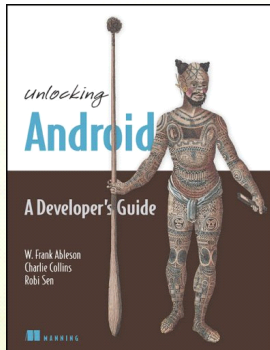
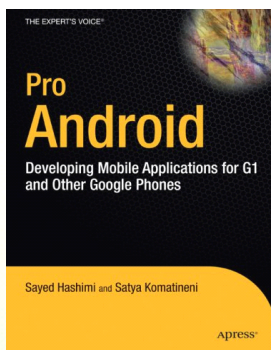
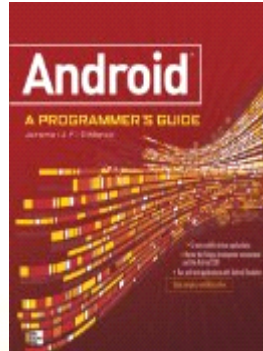
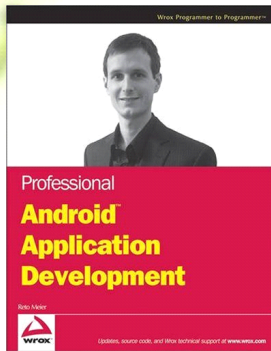








Android 外文书籍



Android 网站资源收集大全

eeAndroid 国内一流的 **Android** 开发社区

<http://www.eeandroid.com/?fromuid=3245>

eeMobile 国内一流的 **Android** 开发团队

<http://www.eemobile.com/index>

Android 爱好者论坛

<http://www.loveandroid.com/>

Android 开发者论坛

<http://www.androidin.com/>

google 主站:

<http://code.google.com/android/>

E 文文档

<http://code.google.com/android/documentation.html>

Android 中国开发者团队 **google groups**

<http://groups.google.com/group/android-developers-zh>

中文社区

<http://www.androidcn.net>

<http://www.androidcn.net/wiki>

google groups

<http://groups.google.com/group/android-developers>

Other Open Handset Alliance Project

<http://code.google.com/p/android/downloads/list>

OHA: 开放手持设备联盟官方网站

<http://www.openhandsetalliance.com/>

GPhone 论坛

<http://bbs.gphone999.com/>

Linux 基金会

<http://www.linux-foundation.org/>

Mobile Linux

http://www.linux-foundation.org/en/Mobile_Linux

OMA: 开放移动联盟

<http://www.linux-foundation.org/>

LiPS Forum

<http://www.lipsforum.org/>

CE Linux Forum

<http://tree.celinuxforum.org/>



Gnome Mobile and Embedded (GMAE)

<http://www.gnome.org/mobile/>

LiMo Foundation

<https://www.limofoundation.org/sf/sfmain/do/home>

Ubuntu Mobile

<https://lists.ubuntu.com/archive...007-May/000289.html>



Mobile & Internet Linux Project

<http://www.moblin.org/>

国外网站

<http://www.androidboards.com/>

<http://www.androidev.com/>

<http://androidcommunity.com/>

<http://anddev.org/>

Android Wiki - Main Page

http://androidwiki.com/wiki/Main_Page

Official Google Android Developers Blog

<http://android-developers.blogspot.com/>

Open Handset Magazine

<http://openhandsmagazine.com/>

Google Android Platform - The Android Log

<http://theandroidlog.com/>

android
developers



android