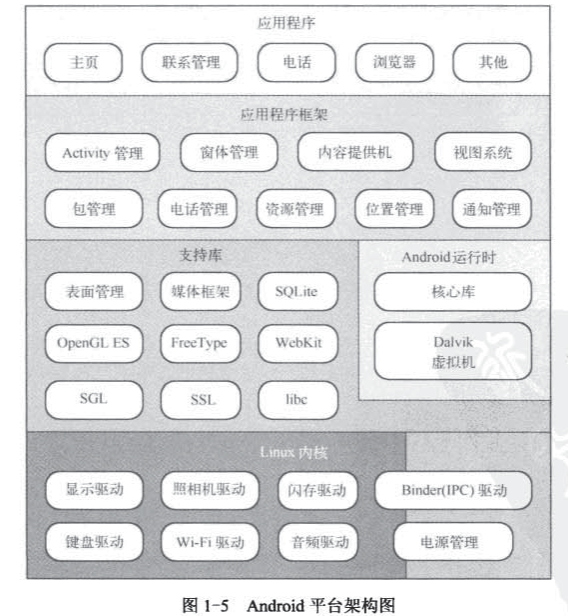
# Android认识

## 什么是android

Android是基于linux的自由以及开放源代码的操作系统

## android体系结构



## 安卓系统的软件后缀名apk简介

APK是AndroidPackage的缩写，即Android安装包(apk)。APK是类似Symbian Sis或Sisx的文件格式。通过将APK文件直接传到Android模拟器或Android手机中执行即可安装。apk文件和sis一样，把android sdk编译的工程打包成一个安装程序文件，格式为apk。 APK文件其实是zip格式，但后缀名被修改为apk，通过UnZip解压后，可以看到Dex文件，Dex是Dalvik VM executes的全称，即Android Dalvik执行程序，并非Java ME的字节码而是Dalvik字节码。

编辑本段文件结构

　　一个APK文件结构为：

　　1. META-INF\ （注：Jar文件中常可以看到）；

　　2. res\ (注：存放资源文件的目录) ；

　　3. AndroidManifest.xml (注：程序全局配置文件) ；

　　4. classes.dex （注：Dalvik字节码）；

　　5. resources.arsc (注：编译后的二进制资源文件)。

　　总结下我们发现Android在运行一个程序时首先需要UnZip，然后类似Symbian那样直接执行安装，和Windows Mobile中的PE文件有区别，这样做对于程序的保密性和可靠性不是很高，通过dexdump命令可以反编译，但这样做符合发展规律，微软的 Windows Gadgets或者说WPF也采用了这种构架方式。

　　在Android平台中dalvik vm的执行文件被打包为apk格式，最终运行时加载器会解压然后获取编译后androidmanifest.xml文件中的permission分支相关的安全访问，但仍然存在很多安全限制，如果你将apk文件传到/system/app文件夹下会发现执行是不受限制的。

　　最终我们平时安装的文件可能不是这个文件夹，而在android rom中系统的apk文件默认会放入这个文件夹，它们拥有着root权限。

# 开始android

## Android调试桥（adb,android debug bridge)

主要开发调试的用途ADB是一个客户端-服务器端程序, 其中客户端是你用来操作的电脑, 服务器端是android设备。如果你已经安装了Android SDK 那么你可以在你的SDK目录下找到 ..\android-sdk-windows\platform-tools 这个目录，ADB工具就放在这里

### 常用的ADB命令

a) adb install <apk文件路径>

这个命令将指定的apk文件安装到设备上

b) adb uninstall <软件名>

adb uninstall -k <软件名>

如果加 -k 参数,为卸载软件但是保留配置和缓存文件.

c)adb shell

这个命令将登录设备的shell

adb shell <command命令>

后面加<command命令>将是直接运行设备命令, 相当于执行远程命令

d)adb help

这个命令将显示帮助信息

c)adb push <本地路径> <远程路径>

从电脑上发送文件到设备

adb remount 在执行push命令之前我们还需要使用remount命令获取上传文件权限，否则会出现

Read-only file system 的错误提示信息

这里要注意的就是如果你需要向一个二级目录里放文件的时候，一定使用"/"而不能使用"\"，原因是android是liunx下的系统，而我们用的是windows。比如：sdcard/aa这样就好啦，否则会提示找不到或者权限不够。

d)adb get-product

获取设备的ID

e)adb get-serialno

获取设备的序列号

f)adb devices

获取当前运行的模拟器/设备的实例的列表及每个实例的状态

g) adb bugreport

查看bug报告

5、常用的ADB Shell命令

通过adb shell命令进入shell

a)访问数据库

sqlite3

b)记录无线通讯日志

一般来说，无线通讯的日志非常多，在运行时没必要去记录，但我们还是可以通过命令，设置记录：

logcat -b radio

c)删除应用

Android没有提供一个卸载应用的命令，需要自己手动删除：

cd /data/app

rm 应用名称.apk

d)查看目录

ls

e)打印或设置当前系统时间

date

f)查看内存信息

cat /proc/meminfo

g)查看CPU信息

cat /proc/cpuinfo

6、删除Android系统Rom自带的软件

a.确定手机root了,取得了root权限才能删除系统文件呀.

b.下载Android\_db.rar,解压到%windir/%System32下.

c.手机连接数据线,在电脑上打开cmd,然后输入命令

adb remount

adb shell

su

执行完成之后,你会看到:

\* daemon not running. starting it now \*

\* daemon started successfully \*

d.接着就是Linux命令行模式了,输入

cd system/app

你会发现没啥变化,然后输入ls回车.

这时候列表显示了system/app里面的所有文件,也就是Rom集成的一些软件了.

e.开始删除吧！比如删除Youtube,他的文件名是Youtube.odex和Youtube.apk

我们要删除这2个文件,敲入以下命令:

rm Youtube.\*

重启,Youtube已经删除掉了,注意各位同学千万不要删除你不知道是啥的东西.

7、sqlite3 adb命令

进入数据库的两种方式

第一种：sqlite3 数据库名称.xx

第二种：adb shell

sqlite3

.databases 列出数据库文件名

.tables ?PATTERN? 列出?PATTERN?匹配的表名

.import FILE TABLE 将文件中的数据导入的文件中

.dump ?TABLE? 生成形成数据库表的SQL脚本

.output FILENAME 将输出导入到指定的文件中

.output stdout 将输出打印到屏幕

.mode MODE ?TABLE? 设置数据输出模式(csv,html,tcl…

.nullvalue STRING 用指定的串代替输出的NULL串

.read FILENAME 执行指定文件中的SQL语句

.schema ?TABLE? 打印创建数据库表的SQL语句

.separator STRING 用指定的字符串代替字段分隔符

.show 打印所有SQLite环境变量的设置

.quit 退出命令行接口

## Dalvik调试监视服务工具----ddms

Dalvik调试监视程序（dalvik debug monitor server，DDMS）工具整合了android平台的虚拟机：可以获取更多的应用程序信息

将应用程序在Dalvik Runtime执行时发生的错误以logcat命令方式返回给开发人员。

　　支持窗体撷取功能，可以将错误的窗体或信息撷取下来以方便后续调试。

　　提供一个port-forwarding服务，可以将这个调试工具安装到实际硬件设备上，也就是说当一般用户使用这些应用程序时，若是发生错误，可以将这些错误信息拦截下来，返回给应用程序开发人员。

　　提供设备上线程（Thread）和堆栈（Heap）的信息和状况。

　　还有无线状态信息、拨入电话、短消息、模仿经纬度位置的数据等等。

　　DDMS与adb互为表里，DDMS需要通过adb衔接到Android仿真器或手机装置进行错误监控。

　　因为Android的平台是Linux，它支持多任务多线程，因此每一个在Android上面运行的应用程序，都有单独的进程，创建起单独的Dalvik虚拟机，并通过单独的port连接到DDMS除错工具。

　　所以即使同时运行多个程序，每一个应用程序发生的错误都会分别发送到DDMS进行记录，不会有漏掉的情况。

　　DDMS工具放在Android SDK的tools目录下，命令行下输入ddms即可启动一个Dalvik Debug Monitor窗口。我们在Eclipse中使用的DDMS就是对该工具的集成。

DDMS显示消息

　　DDMS画面的下面方框实时显示每一个程序（以进程号码PID表示）通过Dalvik虚拟机所传出的实时消息，也就是运行adb logcat时所显示的消息。

　　DDMS有分类功能：

　　　V: Verbose 显示全部消息

　　　　D: Debug

　　　　I: Info 一般消息

　　　　W: Warning

　　　　E: Error

DDMS的资源管理功能

　　可以使用命令行模式adb shell进行文件管理，但是窗口界面更加方便。

　　DDMS功能菜单中的Device选项里的File Explorer项目就可以启动资源管理器。

DDMS截图功能

　　单击Device->Screen Capture或是直接按快捷键Ctrl+S，就可以直接抓取Android仿真器当前屏幕上的画面。

DDMS系统检视功能

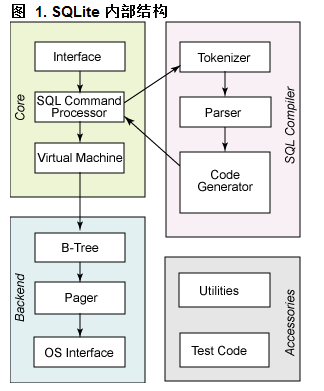
　Device->Show process status会显示Android系统上所有正在运行的进程，也可运行adb shell ps-x指令得到相同的结果。

　　Device->Dump device state就是adb shell dumpsys显示所有的系统消息。

　　Device->Run logcat就是adb logcat显示DDMS画面上的除错消息。

## Sqlite工具----sqlite3

SQLite 是一款非常流行的嵌入式数据库，它支持 SQL 查询，并且只用很少的内存。Android 在运行时集成了 SQLite，所以每个 Android 应用程序都可以使用 SQLite 数据库。对数熟悉 SQL 的开发人员来时，使用 SQLite 相当简单。可以，由于 JDBC 不适合手机这种内存受限设备，所以 Android 开发人员需要学习新的 API 来使用 SQLite。

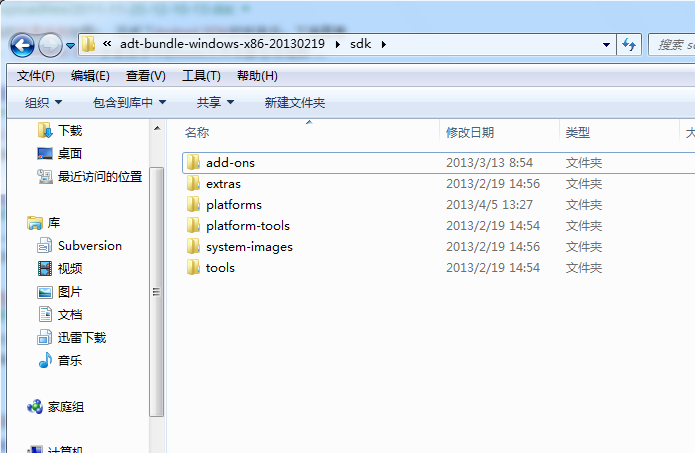


## 资产打包工具----aapt

aapt即Android Asset Packaging Tool , 在SDK的tools/目录下. 该工具可以查看, 创建, 更新ZIP格式的文档附件(zip, jar, apk). 也可将资源文件编译成二进制文件.

## Android目录说明：

打开下载下来的SDK，会发现里面有许多存放不同文件的文件夹，如下图所示，那么这些文件夹里到底存放的是什么呢，下面就对这些内容，进行讲解。



1. add-ons 这里面保存着附加库，比如Google Maps，当然你如果安装了Ophone SDK，这里也会有一些类库在里面。

2. docs 这里面是Android SDK API参考文档，所有的API都可以在这里查到。

3. market\_licensing 作为Android Market版权保护组件，一般发布付费应用到电子市场可以用它来反盗版。

4. platforms 是每个平台的SDK真正的文件，里面会根据API Level划分的SDK版本， 这里就以Android 2.2来说，进入后有一个android-8的文件夹，android-8进入后是Android 2.2 SDK的主要文件，其中ant为ant编译脚本，data保存着一些系统资源，images是模拟器映像文件，skins则是Android模拟器的皮肤，templates是工程创建的默认模板，android.jar则是该版本的主要framework文件，tools目录里面包含了重要的编译工具，比如aapt、aidl、逆向调试工具dexdump和编译脚本dx。

5. platform-tools 保存着一些通用工具，比如adb、和aapt、aidl、dx等文件，Android123提示，这里和platforms目录中tools文件夹有些重复，主要是从android 2.3开始这些工具被划分为通用了。

6. samples 是Android SDK自带的默认示例工程，里面的apidemos强烈推荐初学者运行学习，对于SQLite数据库操作可以查看NotePad这个例子，对于游戏开发Snake、LunarLander都是不错的例子，对于Android主题开发Home则是android m5时代的主题设计原理。

7. tools 作为SDK根目录下的tools文件夹，这里包含了重要的工具，比如ddms用于启动Android调试工具，比如logcat、屏幕截图和文件管理器，而draw9patch则是绘制android平台的可缩放png图片的工具，sqlite3可以在PC上操作SQLite数据库，而monkeyrunner则是一个不错的压力测试应用，模拟用户随机按键，mksdcard则是模拟器SD映像的创建工具，emulator是android模拟器主程序，不过从android 1.5开始，需要输入合适的参数才能启动模拟器，traceview作为android平台上重要的调试工具。

8. usb\_driver 顾名思义，保存着android平台google官方机型的驱动如nexus one、nexus s，同时也有一些老机型驱动的支持，比如说htc dream、htc magic和motorola的droid。

#### ****src文件夹****

顾名思义（src, source code）该文件夹是放项目的源代码的。打开HelloWorld.java文件会看到如下代码：

package helloworld.test;



import android.app.Activity;

import android.os.Bundle;

public class HelloWorld extends Activity {

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

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

}

}

可以知道：我们新建一个简单的HelloWorld项目，系统为我们生成了一个HelloWorld.java文件。他导入了两个类android.app.Activity和android.os.Bundle，HelloWorld类继承自Activity且重写了onCreate方法。

以下说明针对没有学过Java或者Java基础薄弱的人

@Override

在重写父类的onCreate时，在方法前面加上@Override 系统可以帮你检查方法的正确性。例如，public void onCreate(Bundle savedInstanceState){…….}这种写法是正确的，如果你写成public void oncreate(Bundle savedInstanceState){…….}这样编译器回报如下错误——The method oncreate(Bundle) of type HelloWorld must override or implement a supertype method，以确保你正确重写onCreate方法。（因为oncreate应该为onCreate）

而如果你不加@Override，则编译器将不会检测出错误，而是会认为你新定义了一个方法oncreate。

android.app.Activity类：因为几乎所有的活动（activities）都是与用户交互的，所以Activity类关注创建窗口，你可以用方法setContentView(View)将自己的UI放到里面。然而活动通常以全屏的方式展示给用户，也可以以浮动窗口或嵌入在另外一个活动中。有两个方法是几乎所有的Activity子类都实现的：

1. onCreate(Bundle)：初始化你的活动（Activity），比如完成一些图形的绘制。最重要的是，在这个方法里你通常将用布局资源（layout resource）调用setContentView(int)方法定义你的UI，和用findViewById(int)在你的UI中检索你需要编程地交互的小部件（widgets）。setContentView指定由哪个文件指定布局（main.xml），可以将这个界面显示出来，然后我们进行相关操作，我们的操作会被包装成为一个意图，然后这个意图对应有相关的activity进行处理。
2. onPause()：处理当离开你的活动时要做的事情。最重要的是，用户做的所有改变应该在这里提交（通常ContentProvider保存数据）。

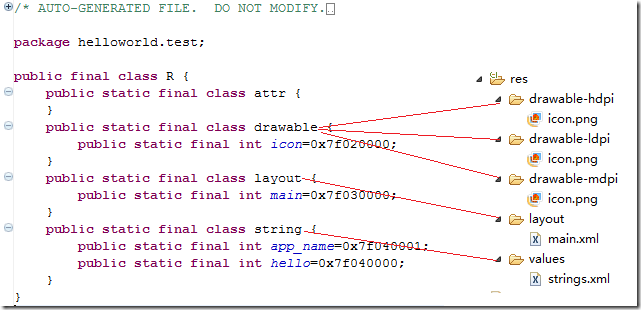
更多的关于Activity类的详细信息此系列以后的文章将做介绍，如果你想了解更多请参阅相关文档。

android.os.Bundle类：从字符串值映射各种可打包的（Parcelable）类型（Bundle单词就是捆绑的意思，所有这个类很好理解和记忆）。如该类提供了公有方法——public boolean containKey(String key)，如果给定的key包含在Bundle的映射中返回true，否则返回false。该类实现了Parceable和Cloneable接口，所以它具有这两者的特性。

#### ****1.2、gen文件夹****

该文件夹下面有个R.java文件，R.java是在建立项目时自动生成的，这个文件是只读模式的，不能更改。R.java文件中定义了一个类——R，R类中包含很多静态类，且静态类的名字都与res中的一个名字对应，即R类定义该项目所有资源的索引。看我们的HelloWorld项目是不是如此，如下图：

图2、R.java对应res



**通过R.java我们可以很快地查找我们需要的资源，另外编绎器也会检查R.java列表中的资源是否被使用到，没有被使用到的资源不会编绎进软件中，这样可以减少应用在手机占用的空间。**

#### ****1.3、Android 2.1文件夹****

该文件夹下包含android.jar文件，这是一个Java 归档文件，其中包含构建应用程序所需的所有的Android SDK 库（如Views、Controls）和APIs。通过android.jar将自己的应用程序绑定到Android SDK和Android Emulator，这允许你使用所有Android的库和包，且使你的应用程序在适当的环境中调试。例如上面的HelloWorld.java源文件中的：

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

这里两行代码就是从android.jar导入包。

#### ****1.4、assets****

包含应用系统需要使用到的诸如mp3、视频类的文件。

#### ****1.5、res文件夹****

资源目录，包含你项目中的资源文件并将编译进应用程序。向此目录添加资源时，会被R.java自动记录。新建一个项目，res目录下会有三个子目录：drawabel、layout、values。

* drawabel-?dpi：包含一些你的应用程序可以用的图标文件(\*.png、\*.jpg)
* layout：界面布局文件(main.xml)与WEB应用中的HTML类同，没修改过的main.xml文件如下（HelloWorld的就没有修改过）：
* values：软件上所需要显示的各种文字。可以存放多个\*.xml文件，还可以存放不同类型的数据。比如arrays.xml、colors.xml、dimens.xml、styles.xml

#### ****1.6、AndroidManifest.xml****

项目的总配置文件，记录应用中所使用的各种组件。这个文件列出了应用程序所提供的功能，在这个文件中，你可以指定应用程序使用到的服务(如电话服务、互联网服务、短信服务、GPS服务等等)。另外当你新添加一个Activity的时候，也需要在这个文件中进行相应配置，只有配置好后，才能调用此Activity。AndroidManifest.xml将包含如下设置：application permissions、Activities、intent filters等。

如果你跟我一样是ASP.NET出生或者学过，你会发现AndroidManifest.xml跟web.config文件很像，可以把它类同于web.config文件理解。

如果你不是，你可以这样理解——众所周知xml是一种数据交换格式，AndroidManifest.xml就是用来存储一些数据的，只不过这些数据时关于android项目的配置数据。

HelloWorld项目的AndroidManifest.xml如下所示：

关于AndroidManifest.xml现在就讲这么多，此系列后面的文章将单独详细介绍。

#### ****1.7、default.properties****

记录项目中所需要的环境信息，比如Android的版本等。 HelloWorld的default.properties文件代码如下所示，代码中的注释已经把default.properties解释得很清楚了：

## Android组件介绍：

Android中没有入口函数所有的组件都是平行的都可以单独实例化：

Android组件是android应用程序的核心：

### Activity组件：---屏幕组件

通常一个activity就是一个屏幕：---通过android的返回键即可返回到这个屏幕（另一activity就会结束）

Acvitity的类的全名是：android.app.acvitity一般通过onCreate()方法来进行初始化---通过setContextView()方法指定一个布局（layout）文件

### Service---服务组件

Service没有用户界面在后台运行每一个服务都是android.app.service的子类

说明：用户在听音乐的时候 还想干其他的事 这时候关闭了音乐的activity但是还想音乐运行这时候就需要service服务组件来进行音乐的播放

### Broadcase receiver---广播组件

用来接收广播并进行回应（启动activity进行回应）

一个应用可以有多个广播接收者---他们都是android.content.ContentProvider的子类（但是只是提供数据和访问数据的接口）访问数据要借助ContentResolver对象

## Android虚拟机Dalvik

Dalvik是android的运行平台：就像java的运行平台JVM一样android是运行在Dalvik上的

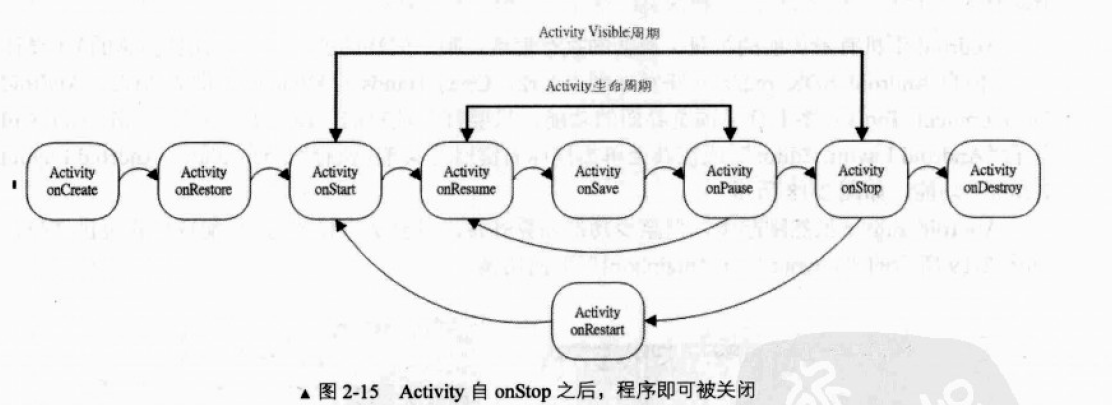
但是Dalivk不同于JVM-----

* 1. 运行专有的.dex文件：专有的.dex文件减少了.class文件的冗余信息并且将所有的.class都整合到一个.dex文件中
  2. 基于寄存器实现：大多数虚拟机都是基于栈的而Dalvik虚拟机则是基于寄存器的（一般来说基于寄存器的具有更好的性能表现,但是在硬件通用性上略差

# Activity---界面组件

界面组件每一个界面都要继承Activity组件：

Activity的生命周期:



Android常用的view类：Android.view.view

