何为1G/ 2G/ 3G/ 4G技术

G代表generation的简称，有代的意思。
1G代表作是大哥大，采用通信标准是模拟制式，只能进行语音通话，不能上网。
2G代表作是小灵通 通信标准是gsm ，既可以进行语音通话也可以简单上网，上网的协议采用wap。
3G时代的通信标准是WCDMA/CDMA2000等，说到3G，中国联通公司宣传的比好（选3G，就用沃）。
4G时代通信标准是TD-LTE，LTE（Long Time Evolution）说到4G中国移动的宣传的比较给力。
区别：最主要区别是采用的制式和上网速度的不同

Android的起源

Android系统最初由安迪·鲁宾等人开发制作，最初开发这个系统的目的是创建一个数码相机的先进操作系统；后来发现市场需求不够大，加上智能手机市场快速成长，于是Android被改造为一款面向智能手机的操作系统。Android系统原来的公司名字就叫做Android，于2005年8月被美国科技企业Google收购。2008年第一款搭载Android系统的手机面世（T-Mobile G1也叫HTC Dream）。

Android的发展史

Android发展史：诞生八年一路辉煌，Android是以Linux为基础的开放源码操作系统。其公司于2003年在美国加州成立。2005年由Google收购注资，并组建开放手机联盟。2007年11月12日，Android Beta操作系统SDK正式发布。 重要版本：1.5 ， 2.3 ， 4.0

版本号	版本名称
1.0	Astro(阿童木) 内测版
1.1	Bender(发条机器人)内测版
1.5	Cupcake(纸杯蛋糕)
1.6	Donut(甜甜圈)
2.1	Eclair(闪电泡芙)
2.2	Froyo(冻酸奶)
2.3	Gingerbread(姜饼)
3.0	Honeycomb(蜂巢)
4.0	Ice cream SandWich(冰激凌三明治)
4.1/4.2/4.3	Jelly Bean(果冻豆)
4.4	KitKat (雀巢巧克力)
5.0/5.1	Lollipop(棒棒糖)(Android L)
6.0	Marshmallow(棉花糖)(Android M)
7.0	“Nougat”（牛轧糖）

Android体系结构 Android开发平台及框架原理

Android其本质就是在标准的Linux系统上增加了Java虚拟机Dalvik，并在Dalvik虚拟机上搭建了一个JAVA的application framework，所有的应用程序都是基于JAVA的application framework之上。Android主要应用于ARM平台，但不仅限于ARM，通过编译控制，在X86、MAC等体系结构的机器上同样可以运行。

Android分四层，

最底层 **linux内核（Linux Kernel），**
函数库层（Libraries），
应用框架层（Application Framework）
最上层 **应用层（Applications）。**

Android体系结构如下图：

蓝色的代表java程序，黄色的代码为运行JAVA程序而实现的虚拟机，绿色部分为C/C++语言编写的程序库，红色的代码内核(linux内核+driver)。在Application Framework之下，由C/C++的程序库组成，通过JNI完成从JAVA到C的调用。

1) 应用程序

所有的应用程序都是使用JAVA语言编写的，每一个应用程序由一个或者多个活动组成，活动必须以Activity类为超类，活动类似于操作系统上的进程，但是活动比操作系统的进程要更为灵活，与进程类似的是，活动在多种状态之间进行切换。

利用JAVA的跨平台性质，基于Android框架开发的应用程序可以不用编译运行于任何一台安装有android系统的平台，这点正是Android的精髓所在。

2) 应用程序框架

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）。帮助程序员快速的开发程序，并且该应用程序重用机制也使用户可以方便的替换程序组件。

隐藏在每个应用后面的是一系列的服务和系统, 其中包括;

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

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

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

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

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

3) 系统运行库

a)程序库

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

以下是一些核心库：

主要包括基本的C库、以及多媒体库以支持各种多媒体格式、位图和矢量字体、2D和3D图形引擎、浏览器、数据库支持。

1.Bionic系统C库。

2.媒体库，基于PacketVideo OpenCORE。

3.Surface Manager 顾名思义，用于管理Surface。

4.Webkit,LibWebCore 浏览器，基于Webkit引擎。

5.SGL 底层的2D图形引擎

6.3D libraries 基于OpenGL ES 1.0 APIs实现

7.FreeType 位图（bitmap）和矢量（vector）字体显示。

8.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。

另外这里还有一个硬件抽象层。其实Android并非所有的设备驱动都放在linux内核里面，有一部分实现在用户空间，

这么做的主要原因是可以避开Linux所遵循的GPL协议，一般情况下如果要将Android移植到其他硬件去运行，

只需要实现这部分代码即可。包括：显示器驱动，声音，相机，GPS，GSM等等。

b)Android 运行库

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

每一个 Android应用程序都在它自己的进程中运行，都拥有一个独立的Dalvik虚拟机实例。

Dalvik被设计成一个设备可以同时高效地运行多个虚拟系统。

Dalvik虚拟机执行（.dex）的Dalvik可执行文件，该格式文件针对小内存使用做了优化。

同时虚拟机是基于寄存器的，所有的类都经由JAVA编译器编译，然后通过SDK中的 "dx" 工具转化成.dex格式由虚拟机执行。

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

4) Linux 内核

Android 的核心系统服务依赖于 Linux 2.6 内核 ，如安全性，内存管理，进程管理， 网络协议栈和驱动模型。

Linux 内核也同时作为硬件和软件栈之间的抽象层。其外还对其做了部分修改，主要涉及两部分修改：

a)Binder

(IPC)：提供有效的进程间通信，虽然linux内核本身已经提供了这些功能，但Android系统很多服务都需要用到该功能，为了某种原因

其实现了自己的一套。

b)电源管理：为手持设备节省能耗。

Android所采用的语言，其应用开发采用java语言，我们所说的java一般包含三个部分：

a) java语言：即其语法，其写代码的程式

b) java虚拟机：为了实现一次编译到处可以运行的原则，java在编译连接以后并没有产生目标机器语言，而是采用了Java bytecode

这种Java共用指令，这时就需要一个虚拟机来执行改指令。

c) 库：跟我们常用的C语言一样提供一些常用的库

后两者结合就是Java Runtime Environment。

Dalvik虚拟机

Dalvik虚拟机是Google公司自主设计的运行于Android平台的Java虚拟机。JVM是归Oracle公司所有的，运行在基于PC机上的Window 和Linux操作系上的java虚拟机。Google工程师之所以设计Dalvik虚拟机主要有以下两点原因：若要使用JVM需要获得授权许可，这意味着需要缴纳大量的费用；JVM主要是针对于CPU快，内存大的传统PC机，不适合移动设备。

JVM和Dalvik VM的区别

JVM(Java Virtual Machine)，Java虚拟机是虚构出来的运行Java程序的运行时，是通过在实际的计算机上仿真模拟各种计算机功能的实现。它具有完善的硬件架构(如处理器、堆栈、寄存器等)，还具有相应的指令系统，使用JVM就是使用Java程序，与操作系统无关。理论上在任何操作系统中，只要有对应的JVM，就可以运行Java程序。

Dalvik VM是在Android系统上运行Android程序的虚拟机，其指令集是基于寄存器架构的，执行特有的文件格式-dex字节码来完成对象生命周期管理、堆栈管理、线程管理、安全异常管理、垃圾回收等重要功能。

最主要的区别在于：

1、打包过程不同：

JVM是将每个java文件编译成独立的class文件，最后以jar形式打包发布，那么在JVM上运行的还是单个独立的class文件

在运行的过程中由于有多个class文件，每个文件都有自己的文件头，所以每个文件头都需要进行处理识别

DVM是将所有的class文件打包成一个.dex文件，最后以apk形式发布。

只有一个文件，所以文件头的处理更快。

DVM的文件占位空间更小，速度更快

2、架构不同：

JVM基于栈内存

DVM基于寄存器

ART模式与DVM的区别

谷歌Android4.4系统新增的一种应用运行模式，与传统的Dalvik模式不同，ART模式可以实现更为流畅的安卓系统体验。

ART模式下，应用运行速度更快，因为ART模式是将应用在安装时，就将class解释为OS认识的机器码，应用在执行时无需再次解释给系统。

缺点：安装应用慢，占用空间大

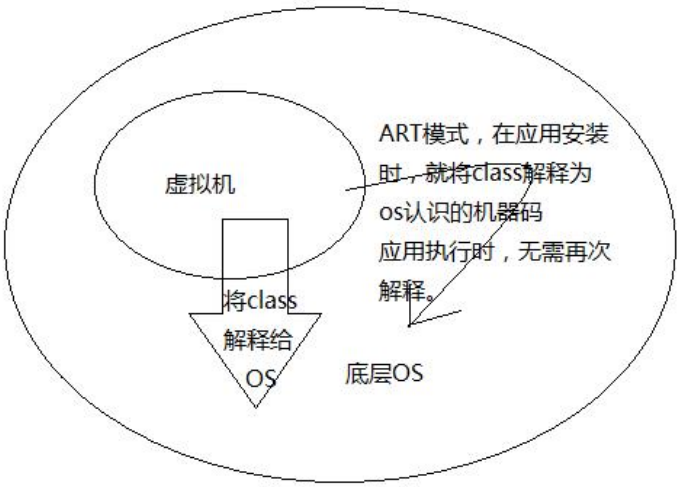
DVM与ART模式的区别

效果：ART模式下，应用运行速度更快

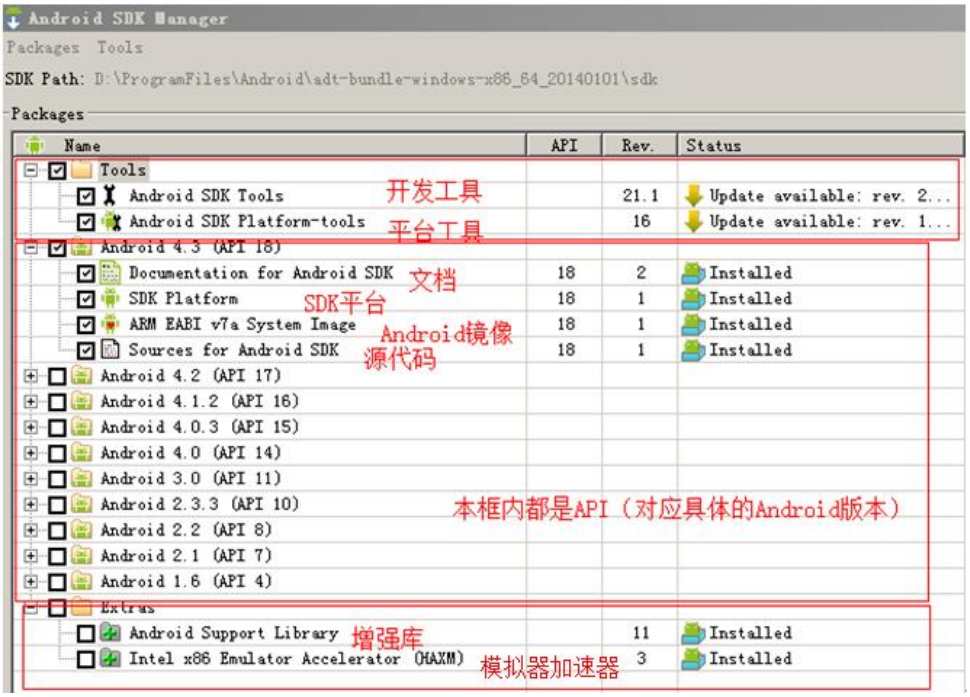
ART模式缺点：安装过程稍慢，安装后的空间占用更大。

ART从4.4开始提供

从5.0开始，强制使用。



SDK Manager　是一个下载开发工具的应用组件



sdk目录结构

adb环境配置：

sdk的安装路径下找到adb.exe的位置设置到环境变量中去

如：　**path**

E:\AndroidStudio\SDK\platform-tools;

adb环境变量

ADB是Android开发调试桥，用来调试应用是否完整正确。实际的应用在DDMS中

常用命令：shell devices

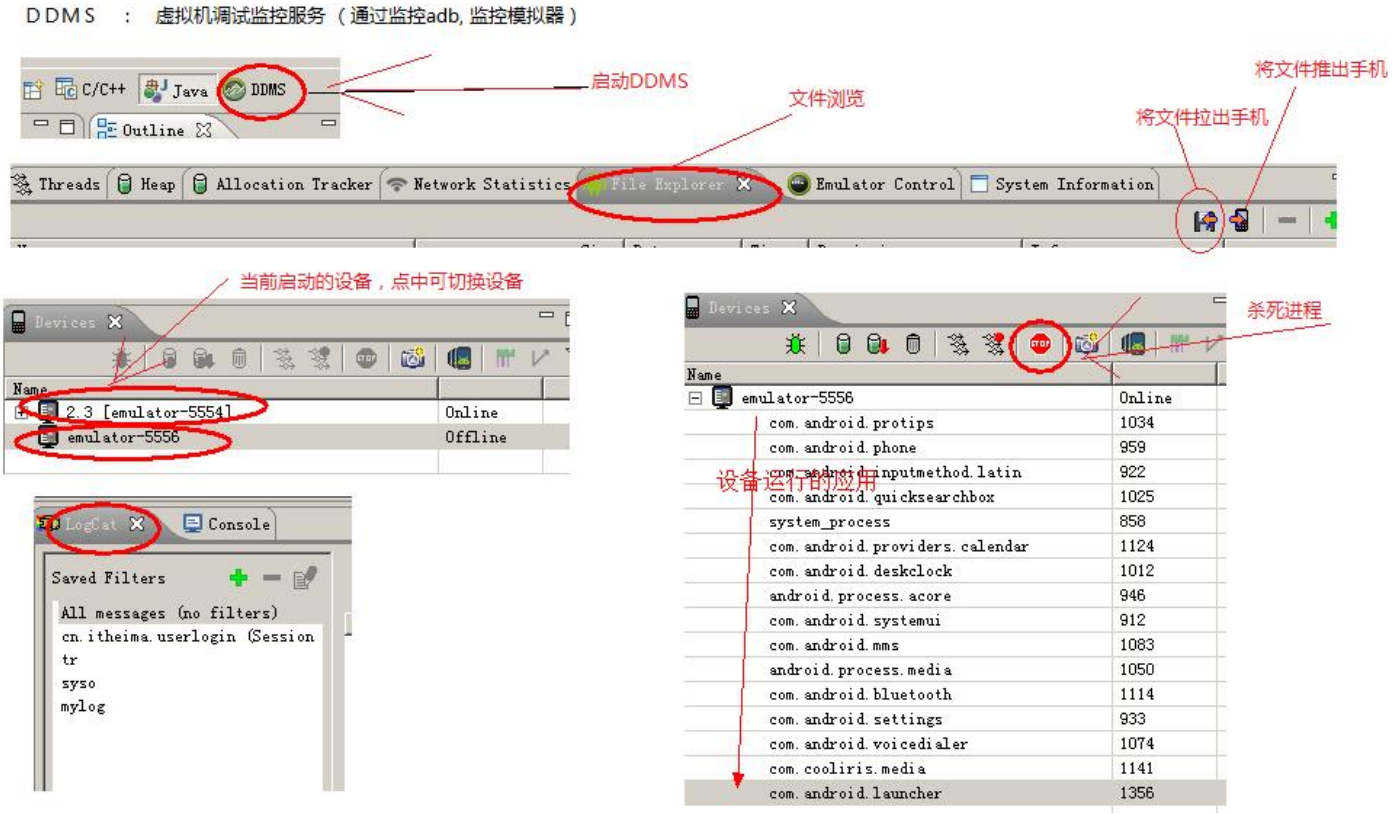
指令	功能
adb devices	列出所有的设备
adb start-server	开启adb服务
adb kill-server	关闭adb服务
adb logcat	查看log

adb shell	挂载到linux空间
adb install	安装应用程序 如adb install helloworld.apk
adb uninstall	卸载应用程序 如 adb uninstall helloworld.apk
adb push	把文件/目录推到手机里
adb pull	把文件/目录从手机里面拉出来

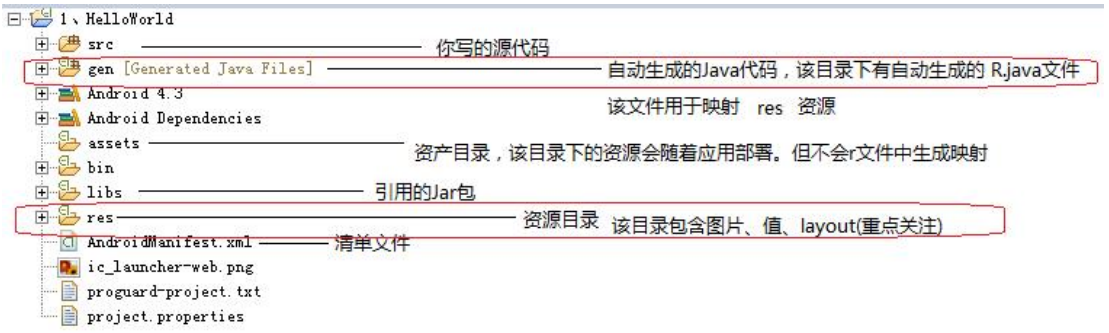
DDMS功能

主要作用是虚拟机调试监控服务，监控ADB，监控模拟器的运行。

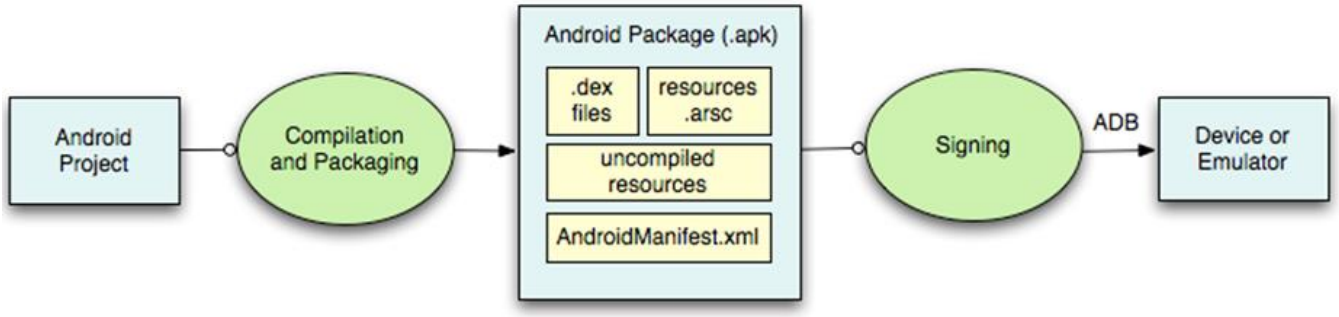
可以通过进程控制模拟器，通过LogCat日志猫查看模拟器的运行情况



创建Android项目工程



apk编译打包过程



第一步：打包资源文件，生成R.java文件
第二步：处理AIDL文件，生成对应的.java文件
第三步：编译Java文件，生成对应的.class文件
第四步：把.class文件转化成Dalvik VM支持的classes.dex文件
第五步：打包生成未签名的.apk文件
第六步：对未签名的.apk文件进行签名
第七步：对前后的.apk文件进行对齐处理。
aapt: Android Application Package Tools 打包apk文件的工具

应用签名

Android系统要求每一个Android应用程序必须要经过数字签名才能够安装到系统中，也就是说如果一个Android应用程序没有经过数字签名，是没有办法安装到系统中的。我们使用开发工具Eclipse直接将应用程序部署到手机或者模拟器上的时候使用了工具默认的debug签名。

Android通过数字签名来表示应用程序的作者和在应用程序之间建立信任关系，这个数字签名由应用程序的作者完成，并不需要权威的数字证书签名机构认证，它只是用来让应用程序包自我认证的。

清单文件解释：
应用的唯一标识符：`package="cn.itheima.hello"`
该标识符是应用升级时我的应用覆盖谁？

程序入口：
在对应的活动上配置

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

应用签名（公司的核心机密）
告诉应用市场，我是应用的所有者。
如果没有签名，应用无法发布
签名在哪里？
默认签名：
你开发是使用的windows账号 \.android\ debua.kevstore

应用的开发步骤：

1、理清应用的功能
2、画应用的界面
在res---->layout------>.xml文件中画界面
按钮是：<Button />
文本输入框是：<EditText />
3、设置相关的事件
写事件代码是在src---->.java的文件中
4、添加权限
在清单文件中（AndroidMainfest.xml）------> Permissions --------> Add ------->添加相应的功能权限
5、运行调试优化
6、应用签名
在清单文件中（AndroidMainfest.xml）------> AndroidMainfest.xml --------> 签名设置

4种按钮点击事件

匿名内部类：UI组件ID.setOnClickListener(new OnClickListener() {重写 onClick(View v)方法});

特点:必须要定义组件ID,只能实现该ID的点击事件

自定义类实现: UI组件ID.setOnClickListenerOnClickListener(new 自定义类()) 自定义类重写onClick(View v)方法

特点:必须要定义组件ID,可以实现该ID的多个点击事件

MainActivity类实现OnClickListener接口:

```
public class MainActivity extends Activity implements OnClickListener {
```

重写onClick(View v)方法

特点:必须要定义组件ID,只能实某个组件的点击事件,多个的组件点击事件必须要做判断是哪个组件点击的

布局绑带点击事件方法名:在布局文件,相应的UI组件中加onClick属性： activity_main.xml ----> 相应组件中 android:onClick="wrid1" ------>在SRC------>MainActivity.JAVA中写对应方法名的方法public void wrid1(View v){}

特点:不要定义组件ID,可以实现所有组件的点击事件,且每个组件的点击事件条例清晰有序.

android五种布局模式 绝对布局已不在使用

android:scaleType="" **ImageView**设置图片填充的模式

将UI组件呈现在手机屏幕上的方式就是布局,

LinearLayout （线性布局） 线形布局中预览和真机中完全一样。

线性布局，这个东西，从外框上可以理解为一个div，UI组件布局方式分两种:水平排列,垂直排列

垂直布局（android:orientation="vertical"）和水平布局（android:orientation="horizontal" ）.

android:padding="10dp" 指定边距

输入类型为密码：android:inputType="textPassword"

UI前景色背景色设置：android:background="#0088FF" 值越大颜色越深

android:background="#FFFFFF" 透明色

限制只能一行输入：android:singleLine="true"

限制为多行显示:android:inputType="textMultiLine"

显示行数:minlines="N";

组件内容对齐:gravity="位置";

限制输入字符长度：android:maxLength="8"

UI内边距：android:padding

外边距：android:layout_marginBottom="3dp"

UI显示提示信息：android:hint="请输入用户名"/>

设置UI点击事件：android:onClick

drawableBottom的用法:

android:drawableBottom：在text的下方输出一个drawable，可以是图片，样式，颜色等。

android:drawableLeft：在text的左边输出一个drawable，可以是图片，样式，颜色等。

android:drawableRight：在text的右边输出一个drawable，可以是图片，样式，颜色等。

android:drawableTop：在text的正上方输出一个drawable，可以是图片，样式，颜色等。

android:drawableStart：在text的开始处输出一个drawable，可以是图片，样式，颜色等。

android:drawableEnd：在text的结束处输出一个drawable，可以是图片，样式，颜色等。

android:drawablePadding：设置text与drawable的间距，是与drawableLeft、 drawableRight、drawableTop、drawableBottom一起使用。

LinearLayout中有一个重要的属性**android:layout_weight="1"**，这个weight在垂直布局时，代表行距；水平的时候代表列宽；weight值越大就越大。

android:orientation：vertical (垂直方向) 、horizontal(水平方向)

FrameLayout（帧布局）

android:layout_gravity="center"　　指定居中

android:background=　　"背景色或图片"

android:textColor="　　字体颜色　　"

FrameLayout是最简单的一个布局对象。它被定制为你屏幕上的一个空白备用区域，之后你可以在其中填充一个单一对象 — 比如，一张你要发布的图片。所有的子元素将会固定在屏幕的左上角；你不能为FrameLayout中的一个子元素指定一个位置。后一个子元素将会直接在前一个子元素之上进行覆盖填充，把它们部份或全部挡住（除非后一个子元素是透明的）。

RelativeLayout（相对布局）

相对布局可以理解为某一个元素为参照物，来定位的布局方式。

android:layout_centerHorizontal="true"　　水平居中

android:layout_centerVertical="true"　　垂直居中

android:layout_above="　@id/id名"　　在某组件上面

android:layout_below="　 @id/id名"　　在某组件下面

android:layout_toLeftOf="@id/id名"　　在某组件左面

android:layout_toRightOf="@id/id名"　　在某组件右面

android:layout_alignLeft="　@id/id名"　　以某组件左对齐

android:layout_alignTop="@id/id名"　　以某组件上对齐

android:layout_alignRight="@id/id名"　　以某组件右对齐

android:layout_alignBottom="@id/id名"　　以某组件下对齐

android:layout_alignParentRight="true"　　以父组件的右对齐

android:focusable="true"　　组件获取焦点

android:focusableInTouchMode="true"　　组件获取焦点

android:ellipsize="marquee"　　跑马灯功能 TextView

android:marqueeRepeatLimit="marquee_forever"　　跑马灯无限循环

android:clickable="true"　　添加可点击事件功能

设置EditText　 setEnabled(false);　 不可用不可输入

设置EditText　setSelection(num.length());　　光标显示位置

设置EditText输入　EditText.addTextChangedListener(new　TextWatcher()　设置实现查询功能

android：layout_方向 = id　表示 在这个id对应的控件的方向上（上|下）

android：layout_align方向　= id 表示和这个控件的（上下左右）对齐

android：　layout_to方向Of　= id 表示在这个控件的 左或者右

TableLayout（表格布局）　**TabRow只论行，不论列（列自定义）**

<TableRow　　每一个<TableRow>　　</TableRow> 代表一列 ---------->

<TextView　　每一列里的UI组件代表一行

```
android:layout_margin="25dp"          指定外边距
        android:layout_width="wrap_content"|
        android:layout_height="wrap_content"/>
        <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</TableRow>
```

Android中常使用的单位

px: 像素　　px转dp

```
/**
 * 根据手机的分辨率从 px(像素) 的单位 转成为 dp
 */
public int px2dip(Context context, float pxValue) {
    final float scale = context.getResources().getDisplayMetrics().density;
    return (int) (pxValue / scale + 0.5f);
}
```

dp: 独立设备像素（像素密度）：Android中，尺寸单位通常用dp

```
/**
 * 根据手机的分辨率从 dp 的单位 转成为 px(像素)
 */
public int dip2px(Context context, float dpValue) {
    final float scale = context.getResources().getDisplayMetrics().density;
    return (int) (dpValue * scale + 0.5f);
}
```

sp: 文字大小