

2.2.1 LinearLayout(线性布局)

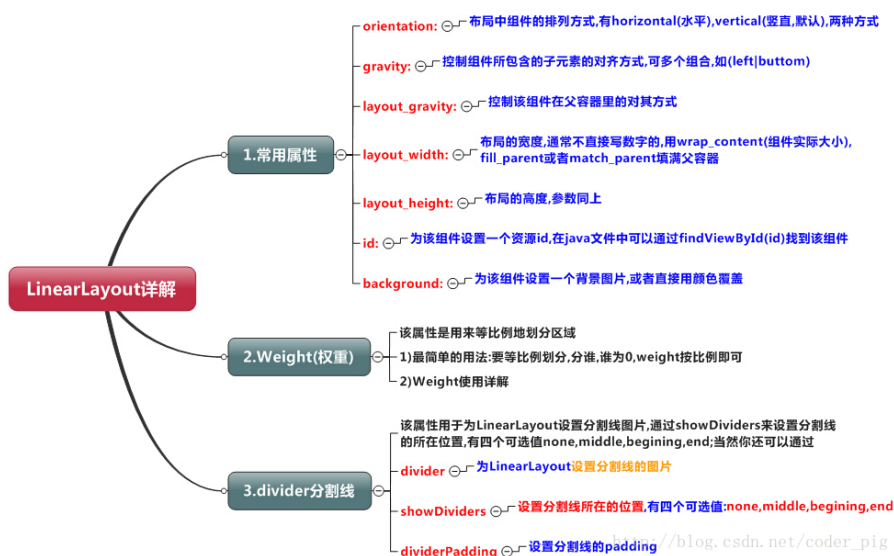
分类 **Android 基础入门教程**

本节引言

本节开始讲Android中的布局，Android中有六大布局,分别是：

LinearLayout(线性布局)，RelativeLayout(相对布局)，TableLayout(表格布局) FrameLayout(帧布局)，AbsoluteLayout(绝对布局)，GridLayout(网格布局) 而今天我们要讲解的就是第一个布局，LinearLayout(线性布局)，我们屏幕适配的使用 用的比较多的就是LinearLayout的weight(权重属性),在这一节里,我们会详细地解析 LinearLayout,包括一些基本的属性,Weight属性的使用,以及比例如何计算,另外还 会说下一个用的比较少的属性:android:divider 绘制下划线！

1.本节学习图

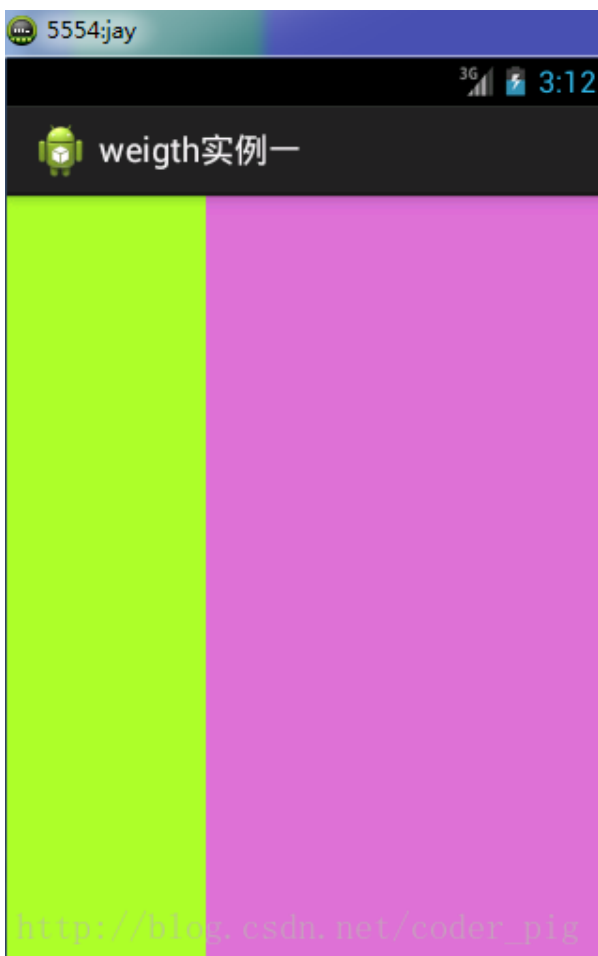
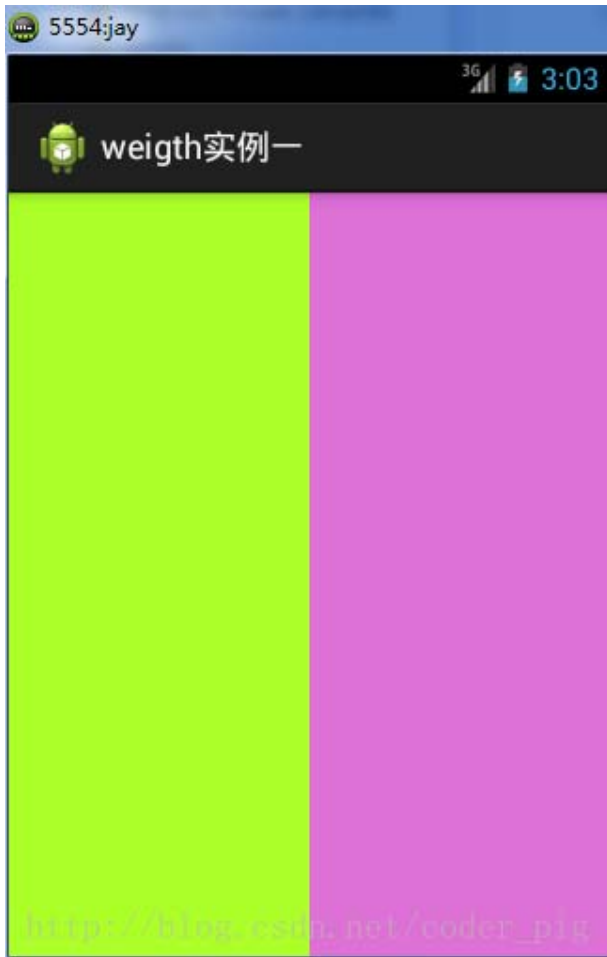


2.weight(权重)属性详解：

①最简单用法：

- 1.0 Android基础入门...
- 1.0.1 2015年最新Andr...
- 1.1 背景相关与系统架...
- 1.2 开发环境搭建
- 1.2.1 使用Eclipse + AD...
- 1.2.2 使用Android Stu...
- 1.3 SDK更新不了问题...
- 1.4 Genymotion模拟...
- 1.5.1 Git使用教程之本...
- 1.5.2 Git之使用GitHub...
- 1.6 .9(九妹)图片怎么玩
- 1.7 界面原型设计
- 1.8 工程相关解析(各种...
- 1.9 Android程序签名...
- 1.11 反编译APK获取代...
- 2.1 View与ViewGroup...
- 2.2.1 LinearLayout(线...
- 2.2.2 RelativeLayout(...
- 2.2.3 TableLayout(表...
- 2.2.4 FrameLayout(帧...
- 2.2.5 GridLayout(网格...
- 2.2.6 AbsoluteLayout(...
- 2.3.1 TextView(文本框...
- 2.3.2 EditText(输入框)...
- 2.3.3 Button(按钮)与I...
- 2.3.4 ImageView(图像...
- 2.3.5.RadioButton(单...
- 2.3.6 开关按钮Toggle...
- 2.3.7 ProgressBar(进...
- 2.3.8 SeekBar(拖动条)

如图：



实现代码：

- 2.3.9 RatingBar(星级...
- 2.4.1 ScrollView(滚动条)
- 2.4.2 Date & Time组...
- 2.4.3 Date & Time组...
- 2.4.4 Adapter基础讲解
- 2.4.5 ListView简单实用
- 2.4.6 BaseAdapter优化
- 2.4.7ListView的焦点问...
- 2.4.8 ListView之check...
- 2.4.9 ListView的数据...
- 2.5.0 构建一个可复用...
- 2.5.1 ListView Item多...
- 2.5.2 GridView(网格视...
- 2.5.3 Spinner(列表选...
- 2.5.4 AutoCompleteT...
- 2.5.5 ExpandableListV...
- 2.5.6 ViewFlipper(翻转...
- 2.5.7 Toast(吐司)的基...
- 2.5.8 Notification(状态...
- 2.5.9 AlertDialog(对话...
- 2.6.0 其他几种常用对...
- 2.6.1 PopupWindow(...
- 2.6.2 菜单(Menu)
- 2.6.3 ViewPager的简...
- 2.6.4 DrawerLayout(...
- 3.1.1 基于监听的事件...
- 3.2 基于回调的事件处...
- 3.3 Handler消息传递...
- 3.4 TouchListener PK ...
- 3.5 监听EditText的内...
- 3.6 响应系统设置的事...
- 3.7 AsyncTask异步任...
- 3.8 Gestures(手势)
- 4.1.1 Activity初学乍练
- 4.1.2 Activity初窥门径

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="fill_parent"
        android:background="#ADFF2F"
        android:layout_weight="1"/>

    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="fill_parent"
        android:background="#DA70D6"
        android:layout_weight="2"/>

</LinearLayout>
```

要实现第一个的1:1的效果,只需要分别把两个LinearLayout的weight改成1和1就可以了 用法归纳: 按比例划分水平方向:将涉及到的View的android:width属性设置为0dp,然后设置为android weight属性设置比例即可;类推,竖直方向,只需设android:height为0dp,然后设weight属性即可! 大家可以自己写个竖直方向的等比例划分的体验下简单用法!

②weight属性详解:

当然,如果我们不适用上述那种设置为0dp的方式,直接用wrap_content和match_parent的话,则要接着解析weight属性了,分为两种情况,wrap_content与match_parent! 另外还要看 LinearLayout的orientation是水平还是竖直,这个决定哪个方向等比例划分

1)wrap_content比较简单,直接就按比例的了

4.1.3 Activity登堂入室

4.2.1 Service初涉

4.2.2 Service进阶

4.2.3 Service精通

4.3.1 BroadcastReceiver...

4.3.2 BroadcastReceiver...

4.4.1 ContentProvider...

4.4.2 ContentProvider...

4.5.1 Intent的基本使用

4.5.2 Intent之复杂数...

5.1 Fragment基本概述

5.2.1 Fragment实例精...

5.2.2 Fragment实例精...

5.2.3 Fragment实例精...

5.2.4 Fragment实例精...

5.2.5 Fragment实例精...

6.1 数据存储与访问之...

6.2 数据存储与访问之...

6.3.1 数据存储与访问...

6.3.2 数据存储与访问...

7.1.1 Android网络编...

7.1.2 Android Http请...

7.1.3 Android HTTP请...

7.1.4 Android HTTP请...

7.2.1 Android XML数...

7.2.2 Android JSON数...

7.3.1 Android 文件上传

7.3.2 Android 文件下...

7.3.3 Android 文件下...

7.4 Android 调用 Web...

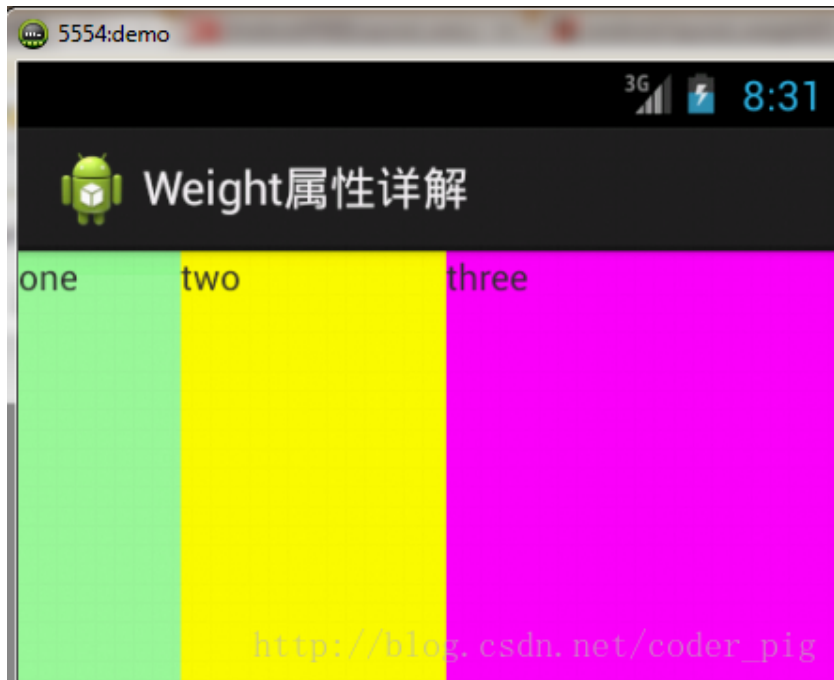
7.5.1 WebView(网页视...

7.5.2 WebView和Java...

7.5.3 Android 4.4后W...

7.5.4 WebView文件下载

7.5.5 WebView缓存问题



实现代码：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >

    <TextView
        android:layout_weight="1"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:text="one"
        android:background="#98FB98"
    />

    <TextView
        android:layout_weight="2"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:text="two"
        android:background="#FFFF00"
    />

    <TextView
        android:layout_weight="3"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:text="three"
        android:background="#FF00FF"
    />

</LinearLayout>
```

7.5.6 WebView处理网...

7.6.1 Socket学习网络...

7.6.2 基于TCP协议的S...

7.6.3 基于TCP协议的S...

7.6.4 基于UDP协议的S...

8.1.1 Android中的13...

8.1.2 Android中的13...

8.1.3 Android中的13...

8.2.1 Bitmap(位图)全...

8.2.2 Bitmap引起的O...

8.3.1 三个绘图工具类...

8.3.2 绘图类实战示例

8.3.3 Paint API之一一 ...

8.3.4 Paint API之一一 ...

8.3.5 Paint API之一一 ...

8.3.6 Paint API之一一 ...

8.3.7 Paint API之一一 ...

8.3.8 Paint API之一一 ...

8.3.9 Paint API之一一 ...

8.3.10 Paint API之一...

8.3.11 Paint API之一...

8.3.12 Paint API之一...

8.3.13 Paint API之一...

8.3.14 Paint几个枚举/...

8.3.15 Paint API之一...

8.3.16 Canvas API详解...

8.3.17 Canvas API详解...

8.3.18 Canvas API详解...

8.4.1 Android动画合...

8.4.2 Android动画合...

8.4.3 Android动画合...

8.4.4 Android动画合...

9.1 使用SoundPool播...

9.2 MediaPlayer播放...

9.3 使用Camera拍照

2)match_parent(fill_parent):这个则需要计算了

我们写这段简单的代码：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
        android:layout_weight="1"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:text="one"
        android:background="#98FB98"
    />
    <TextView
        android:layout_weight="2"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:text="two"
        android:background="#FFFF00"
    />
    <TextView
        android:layout_weight="3"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:text="three"
        android:background="#FF00FF"
    />

</LinearLayout>
```

运行效果图：

9.4 使用MediaRecord...

10.1 TelephonyMana...

10.2 SmsManager(短...

10.3 AudioManager(...

10.4 Vibrator(振动器)

10.5 AlarmManager(...

10.6 PowerManager(...

10.7 WindowManager...

10.8 LayoutInflater(布...

10.9 WallpaperManag...

10.10 传感器专题(1)一...

10.11 传感器专题(2)一...

10.12 传感器专题(3)一...

10.12 传感器专题(4)一...

10.14 Android GPS初涉

11.0 《2015最新Andro...





这个时候就会有疑问了,怎么会这样,这比例是2:1吧,那么three去哪了?代码里面明明有three的啊,还设置了3的,而1和2的比例也不对耶,1:2:3却变成了2:1:0,怎么会这样呢? 答:这里其实没那么简单的,还是需要我们计算的,网上给出的算法有几种,这里就给出笔者觉得比较容易理解的一种: **step 1**: 个个都是fill_parent,但是屏幕只有一个啦,那么 $1 - 3 = -2$ fill_parent **step 2**: 依次比例是1/6,2/6,3/6 **step 3**: 先到先得,先分给one,计算: $1 - 2 * (1/6) = 2/3$ fill_parent 接着到two,计算: $1 - 2 * (2/6) = 1/3$ fill_parent 最后到three,计算 $1 - 2 * (3/6) = 0$ fill_parent **step 4**: 所以最后的结果是:one占了两份,two占了一份,three什么都木有 以上就是为什么three没有出现的原因了,或许大家看完还是有点蒙,没事,我们举多几个例子试试就知道了!

比例为 : 1 : 1 : 1



按照上面的计算方法算一次,结果是:1/3 1/3 1/3,没错

接着我们再试下:2:3:4





计算结果:5/9 3/9 1/9,对比效果图,5:3:1,也没错,所以这个计算方法你可得mark下了!

③Java代码中设置weight属性：

```
setLayoutParams(new LayoutParams(LayoutParams.FILL_PARENT,
    LayoutParams.WRAP_CONTENT, 1));
```

3.为LinearLayout设置分割线

很多界面开发中都会设置一些下划线,或者分割线,从而使得界面更加整洁美观,比如下面的酷狗 音乐的注册页面:

对于这种线,我们通常的做法有两种 ①**直接在布局中添加一个view**,这个view的作用仅仅是显示出一条线,代码也很简单:



```
<View
    android:layout_width="match_parent"
    android:layout_height="1px"
    android:background="#000000" />
```

这个是水平方向上的黑线,当然你也可以改成其他颜色,或者使用图片

②第二种则是使用**LinearLayout**的一个**divider**属性,直接为**LinearLayout**设置分割线 这里就需要你自己准备一张线的图片了 1)android:divider设置作为分割线的图片 2)android:showDividers设置分割线的位置,none(无),begining(开始),end(结束),middle(每两个组件间) 3)dividerPadding设置分割线的Padding

使用示例：



实现代码：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:divider="@drawable/ktv_line_div"
    android:orientation="vertical"
    android:showDividers="middle"
    android:dividerPadding="10dp"
    tools:context="com.jay.example.linearlayoutdemo.MainActivity" >

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="按钮1" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="按钮2" />

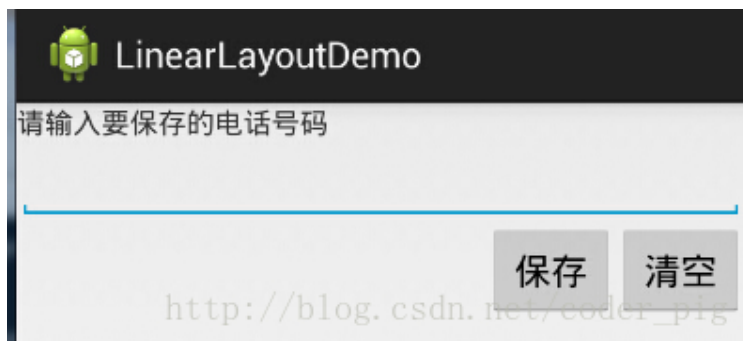
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```



```
android:text="按钮3" />
```

```
</LinearLayout>
```

4.LinearLayout的简单例子:



实现代码如下：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    tools:context=".MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="请输入要保存的电话号码" />

    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:gravity="right">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="保存" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="清空" />

    </LinearLayout>
</LinearLayout>
```

5.注意事项:



使用Layout_gravity的一个很重要的问题!!! 问题内容: 在一个LinearLayout的水平方向中布置两个TextView,想让一个左,一个右,怎么搞? 或许你会脱口而出:"gravity设置一个left,一个right就可以啦!" 真的这么简单?你试过吗?写个简单的Layout你就会发现,事与愿违了: 代码如下 :

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    tools:context="com.jay.example.getscreendemo.MainActivity"
>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="200dp"
        android:layout_gravity="left"
        android:background="#FF7878"
        android:gravity="center"
        android:text="O(∩_∩)O哈哈~" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="200dp"
        android:layout_gravity="right"
        android:background="#FF7428"
        android:gravity="center"
        android:text="(*^__^*) 嘻嘻....." />

</LinearLayout>
```

运行结果图:



看到这里你会说:哎呀,真的不行耶,要不在外层LinearLayout加个gravity=left的属性,然后设置第二个 TextView的layout_gravity为right,恩,好我们试一下:



```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:gravity="left"
    tools:context="com.jay.example.getscreendemo.MainActivity"
>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="200dp"
        android:background="#FF7878"
        android:gravity="center"
        android:text="O(∩_∩)O哈哈~" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="200dp"
        android:layout_gravity="right"
        android:background="#FF7428"
        android:gravity="center"
        android:text="(*^__^*) 嘻嘻....." />

</LinearLayout>
```

结果还是一样：



好吧,没辙了,怎么办好?

当 `android:orientation="vertical"` 时，只有水平方向的设置才起作用，垂直方向的设置不起作用。即：`left`，`right`，`center_horizontal` 是生效的。当 `android:orientation="horizontal"` 时，只有垂直方向的设置才起作用，水平方向的设置不起作用。即：`top`，`bottom`，`center_vertical` 是生效的。



