《Android零基础入门课程》—— 涂涂IT学堂

第四章 Android UI 基础知识

目标

- 了解Android UI
- 布局
- 常用UI控件
- 写一个简单UI项目

01 Android UI

1.1 UI

- **用户界面**（**User Interface，简称 UI**，亦称使用者界面）是系统和用户之间进行交互和信息交换的**媒介**，它实现信息的内部形式与人类可以接受形式之间的转换。
- **软件设计**可分为两个部分：**编码设计与UI设计。**

1.2 Android UI

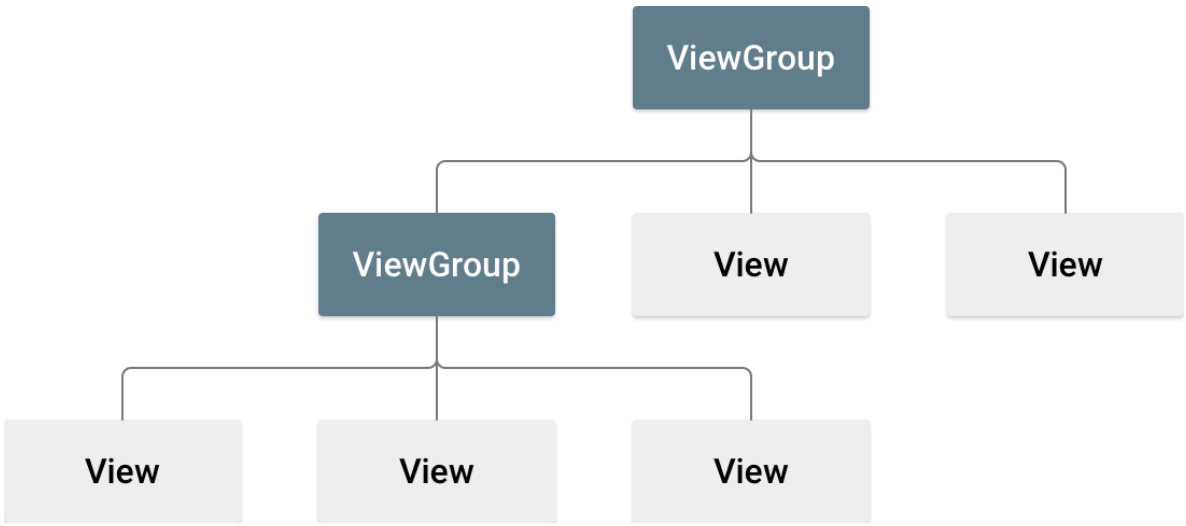
- **Android应用界面**包含用户可查看并与之交互的所有内容。Android 提供丰富多样的**预置 UI 组件**，例如**结构化布局对象和 UI 控件**，您可以利用这些组件为您的应用构建图形界面。Android 还提供其他**界面模块**，用于构建特殊界面，例如**对话框、通知和菜单。**
- Android UI 都是由布局和控件组成的

02 布局

布局(layout)可定义应用中的界面结构（例如 Activity 的界面结构）。布局中的所有元素均使用 View 和 ViewGroup 对象的层次结构进行构建。**View 通常绘制用户可查看并进行交互的内容。**然而，**ViewGroup 是不可见容器，用于定义 View 和其他 ViewGroup 对象的布局结构。**

2.1 布局的结构

- 定义界面布局的视图层次结构图示:



- `View` 对象通常称为“微件”，可以是众多子类之一，例如 `Button` 或 `TextView`。
- `ViewGroup` 对象通常称为“布局”，可以是提供其他布局结构的众多类型之一，例如 `LinearLayout` 或 `ConstraintLayout`。

2.2 声明布局

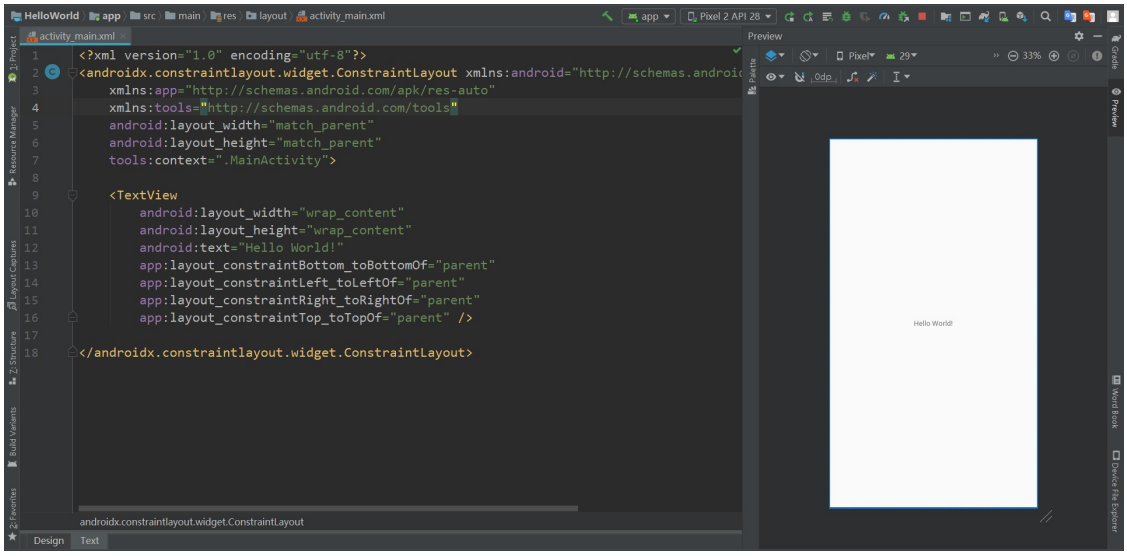
- **在 XML 中声明界面元素**，Android 提供对应 View 类及其子类的简明 XML 词汇，如用于微件和布局的词汇。

  您也可使用 Android Studio 的 **Layout Editor**，并采用拖放界面来构建 XML 布局。

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

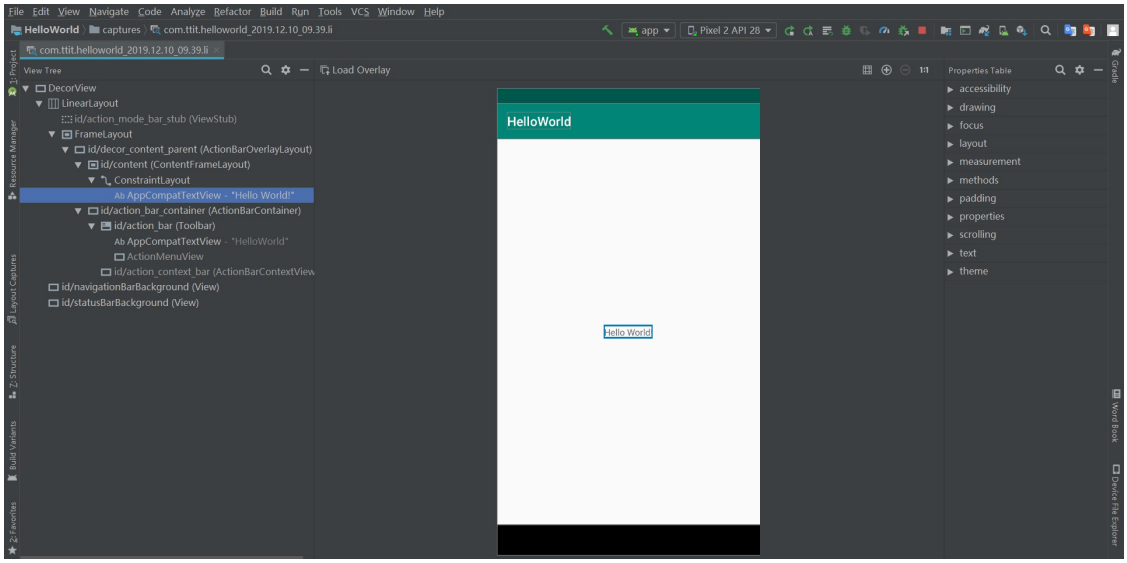


- **在运行时实例化布局元素。**您的应用可通过编程创建 View 对象和 ViewGroup 对象（并操纵其属性）。

```
ConstraintLayout constraintLayout = new ConstraintLayout(this);
TextView view = new TextView(this);
view.setText("Hello World!");
constraintLayout.addView(view);
```

  - *** 提示：**使用 **Layout Inspector** 调试布局，可以查看通过代码创建的布局

    1. 在连接的设备或模拟器上[**运行您的应用**]。
    2. 依次点击 **Tools > Layout Inspector**。
    3. 在显示的 **Choose Process** 对话框中，选择要检查的应用进程，然后点击 **OK**。



2.3 编写XML

- 利用 Android 的 XML 词汇，按照在 HTML 中创建包含一系列嵌套元素的**网页的相同方式**快速设计 UI 布局及其包含的屏幕元素
- 每个布局文件都**必须只包含一个根元素**，并且该元素**必须是视图对象或 ViewGroup 对象**
- 定义根元素后，可以子元素的形式**添加其他布局对象或控件**，从而逐步构建定义布局的视图层次结构
- 在 XML 中声明布局后，以 `.xml` 扩展名将文件保存在Android 项目的 `res/layout/` 目录中

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

2.4 加载 XML 资源

- 当编译应用时，系统会将每个 XML 布局文件编译成 `View` 资源。在 `Activity.onCreate()` 回调内，通过调用 `setContentView()`，并以 `R.layout.*layout_file_name*` 形式向应用代码传递布局资源的引用，加载应用代码中的布局资源。

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
```

2.5 属性

> 每个 View 对象和 ViewGroup 对象均支持自己的各种 `XML 属性`。某些属性是 View 对象的**特有属性**（例如，TextView 支持 `textSize` 属性），但可扩展此类的任一 View 对象也会继承这些属性。某些属性是所有 View 对象的**共有属性**，因为它们继承自 View 根类（例如 `id` 属性）。此外，其他属性被视为“布局参数”，即描述 View 对象特定布局方向的属性，如由该对象的父 ViewGroup 对象定义的属性。

```xml
<TextView
    android:id="@+id/tv"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    android:textSize="24sp"/>

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

2.6 ID

> 任何 View 对象均可拥有与之关联的整型 ID，用于在结构树中对 View 对象进行唯一标识。编译应用后，系统会以整型形式引用此 ID，但在布局 XML 文件中，系统通常会以字符串的形式在 `id` 属性中指定该 ID。这是**所有 View 对象共有的 XML 属性**（由 `View` 类定义），并且您会经常使用该属性。

- XML 标记内部的 ID 语法是：

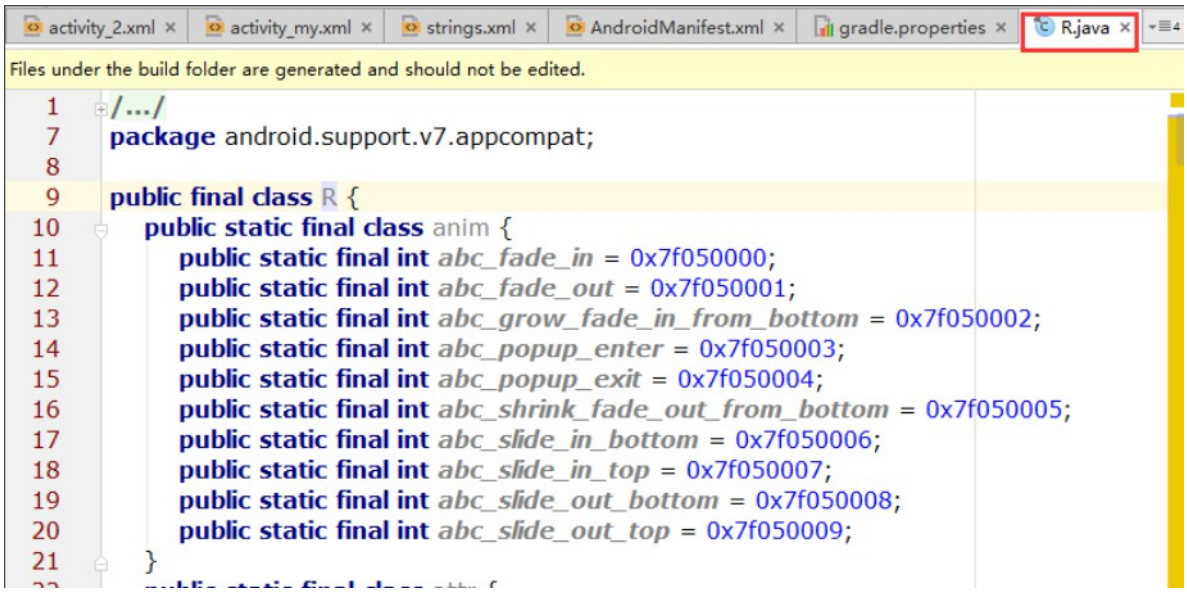
```
android:id="@+id/tv"
```

- 字符串开头处的 @ 符号指示 XML 解析器应解析并展开 ID 字符串的其余部分，并将其标识为 ID 资源。加号 (+) 表示这是一个新的资源名称，必须创建该名称并将其添加到我们的资源（在 `R.java` 文件中）内。Android 框架还提供许多其他 ID 资源。引用 Android 资源 ID 时，不需要加号，但必须添加 `android` 软件包命名空间

  ```
  android:id="@android:id/empty"
  ```

  添加 `android` 软件包命名空间后，我们现在将从 `android.R` 资源类而非本地资源类引用 ID

  - `R.java` 文件



> - **tips:** @+id 和 @id区别：
>
> 其实@+id就是在R.java文件里新增一个id名称，如果之前已经存在相同的id名称，那么会覆盖之前的名称。而@id则是直接引用R.java文件的存在的id资源，如果不存在，会编译报错。

- **注意：** ID 字符串名称，在同一布局中必须是唯一的，不能重名，不同布局中可以同名；
- 通过ID值创建我们视图对象的实例

```
<TextView
     android:id="@+id/tv"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="Hello World!"
     android:textSize="24sp"/>
```

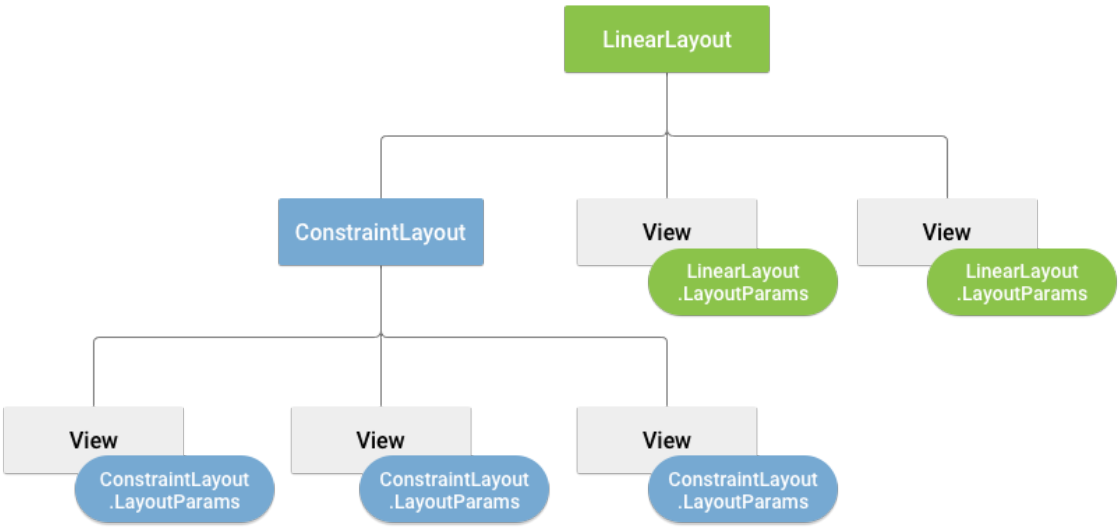
```
TextView textView = (TextView) findViewById(R.id.tv);
```

2.7 布局参数 LayoutParams

- `layout_***` 的布局属性

```
    <TextView
        android:layout_width="100dp"
        android:layout_height="200dp"
        android:layout_marginLeft="10dp" //左边距
        android:layout_marginTop="10dp" //上边距
        android:text="Hello World!" />
```

- 布局参数作用是给我们的视图设定在布局中位置和大小
- ViewGroup 类会实现一个扩展 `ViewGroup.LayoutParams` 的嵌套类，里面包含一些设置视图 `view` 的尺寸和位置的属性。
    - 视图层次结构图，包含每个视图关联的布局参数：



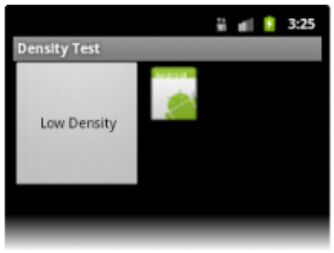

```
TextView tv = new TextView(this);
LinearLayout linearLayout = new LinearLayout(this);
LinearLayout.LayoutParams layoutParams =
(LinearLayout.LayoutParams)tv.getLayoutParams();
layoutParams.leftMargin = 30; //左边距
layoutParams.topMargin = 30;//上边距
layoutParams.width = 100;//宽
layoutParams.height = 200;//高
tv.setLayoutParams(layoutParams);
linearLayout.addView(tv);
```

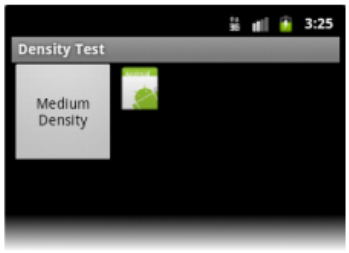
- 一般而言，建议不要使用绝对单位（如像素）来指定布局宽度和高度。更好的方法是使用相对测量单位（如与密度无关的像素单位 `(dp)`、`wrap_content` 或 `match_parent`），因为其有助于确保您的应用在各类尺寸的设备屏幕上正确显示。
    - `wrap_content` 指示您的视图将其大小调整为内容所需的尺寸。
    - `match_parent` 指示您的视图尽可能采用其父视图组所允许的最大尺寸。

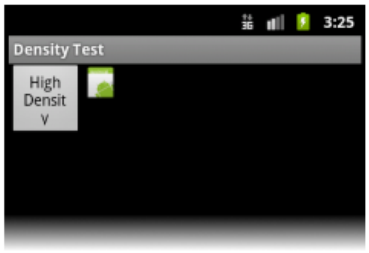
2.8 布局位置

- 视图可以通过调用 `getLeft()` 方法和 `getTop()` 方法来获取视图的坐标位置，也可以通过 `getWidth()` 和 `getHeight()` 获取视图的尺寸，这些方法返回的值是相对于其父视图的位置。

- 位置和尺寸的单位是像素（`px`）
- px 与 dp 区别
  - px 即像素，1px代表屏幕上一个物理的像素点；
    - 给视图设置px单位，不同分辨率下，尺寸不一样



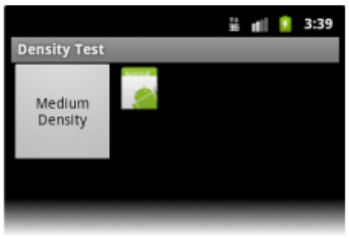


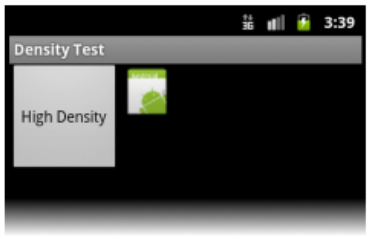


  - dp (dip) Density independent pixels，设备无关像素。它与“像素密度”密切相关；

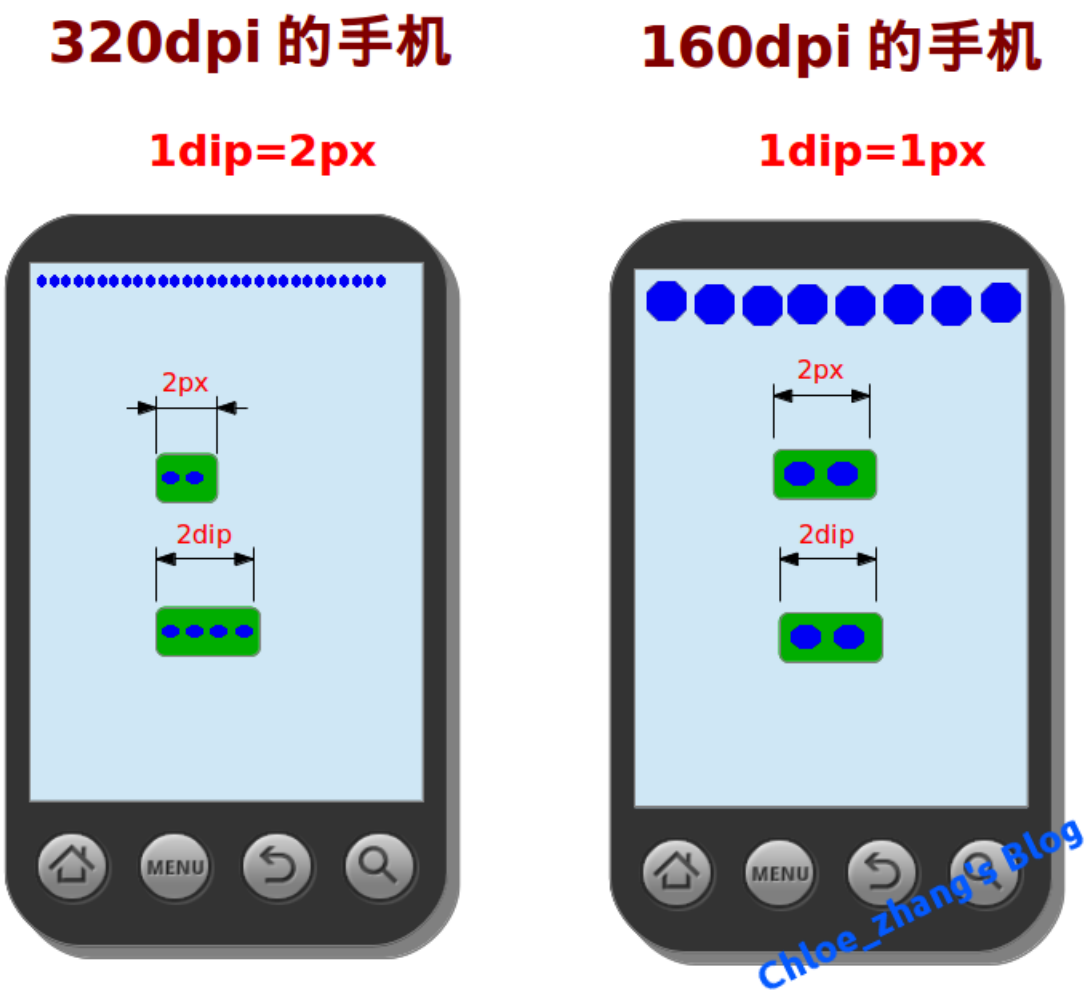
    dpi像素密度： 每英寸包含的像素数

    > 假设有一部手机，屏幕的物理尺寸为1.5英寸x2英寸，屏幕分辨率为240x320，则我们可以计算出在这部手机的屏幕上，每英寸包含的像素点的数量为240/1.5=160dpi（横向）或320/2=160dpi（纵向），160dpi就是这部手机的像素密度，像素密度的单位dpi是Dots Per Inch的缩写，即每英寸像素数量.

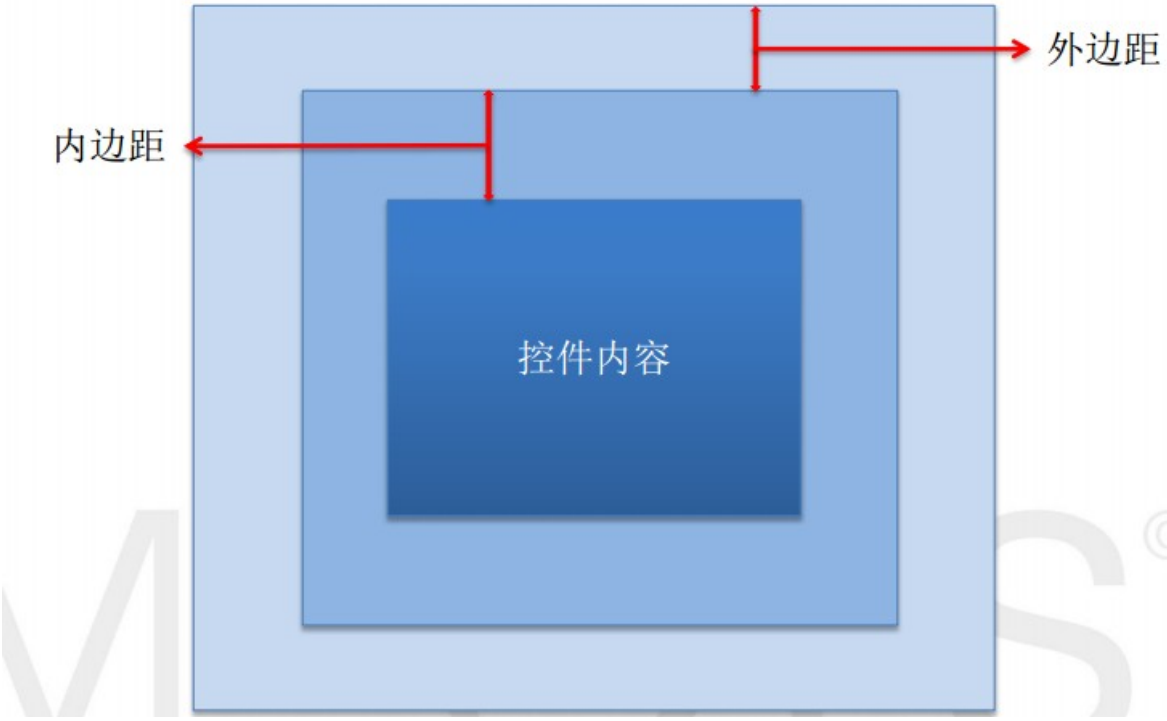




密度类型	代表的分辨率（px）	屏幕密度（dpi）	换算（px/dp）	比例
低密度（ldpi）	240x320	120	1dp=0.75px	3
中密度（mdpi）	320x480	160	1dp=1px	4
高密度（hdpi）	480x800	240	1dp=1.5px	6
超高密度（xhdpi）	720x1280	320	1dp=2px	8
超超高密度（xxhdpi）	1080x1920	480	1dp=3px	12


320dpi 的手机
1dip=2px
160dpi 的手机
1dip=1px
2px
2dip
2px
2dip
MENU
MENU
Chloe_zhang's Blog

2.9 内边距和外边距

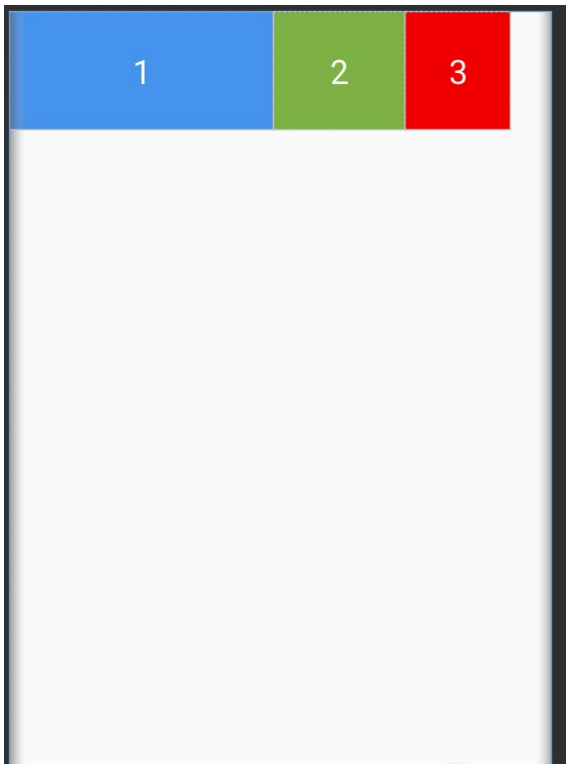

外边距
内边距
控件内容

外边距		内边距	
layout_margin	外边距	padding	内边距
layout_marginTop	上外边距	paddingTop	上内边距
layout_marginBottom	下外边距	paddingBottom	下内边距
layout_marginLeft	左外边距	paddingLeft	左内边距
layout_marginRight	右外边距	paddingRight	右内边距

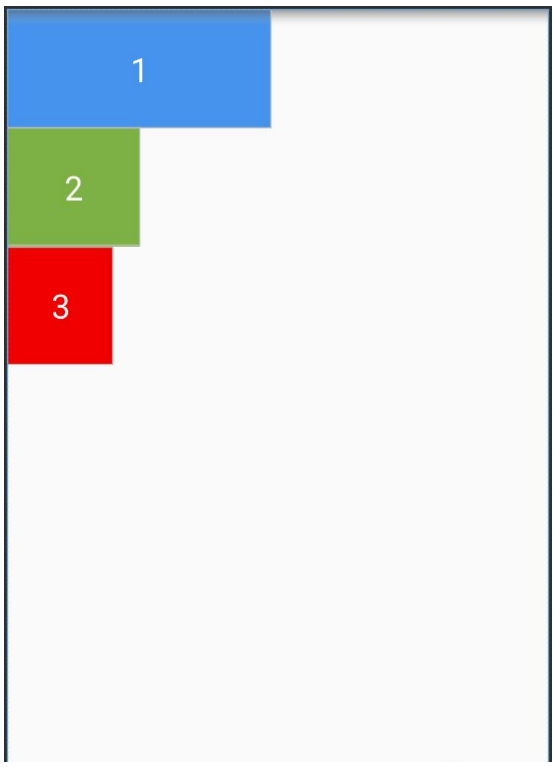
03 常用布局

3.1 线性布局 - LinearLayout

- `LinearLayout` 是一个视图容器，用于使所有子视图在单个方向（**垂直或水平**）保持对齐。您可使用 `android:orientation` 属性指定布局方向。
- `android:orientation="horizontal"`



- `android:orientation="vertical"`



- **布局权重** `android:layout_weight`

  通过给子视图设置权重值，来分配子视图所占空间的权重（比例），如图三个子视图权重分别设置为1，均分页面空间

3.2 相对布局 - RelativeLayout

- 相对布局： 子视图可通过相应的布局属性，设定相对于另一个兄弟视图或父视图容器的相对位置
- 属性说明：
  - 相对于兄弟元素

属性名称	属性含义

属性名称	属性含义
android:layout_below="@id/aaa"	在指定View的下方
android:layout_above="@id/aaa"	在指定View的上方
android:layout_toLeftOf="@id/aaa"	在指定View的左边
android:layout_toRightOf="@id/aaa"	在指定View的右边
android:layout_alignTop="@id/aaa"	与指定View的上边界一致
android:layout_alignBottom="@id/aaa"	与指定View下边界一致
android:layout_alignLeft="@id/aaa"	与指定View的左边界一致
android:layout_alignRight="@id/aaa"	与指定View的右边界一致

- 相对于父元素

属性名称	属性含义
android:layout_alignParentLeft="true"	在父元素内左边
android:layout_alignParentRight="true"	在父元素内右边
android:layout_alignParentTop="true"	在父元素内顶部
android:layout_alignParentBottom="true"	在父元素内底部

- 对齐方式

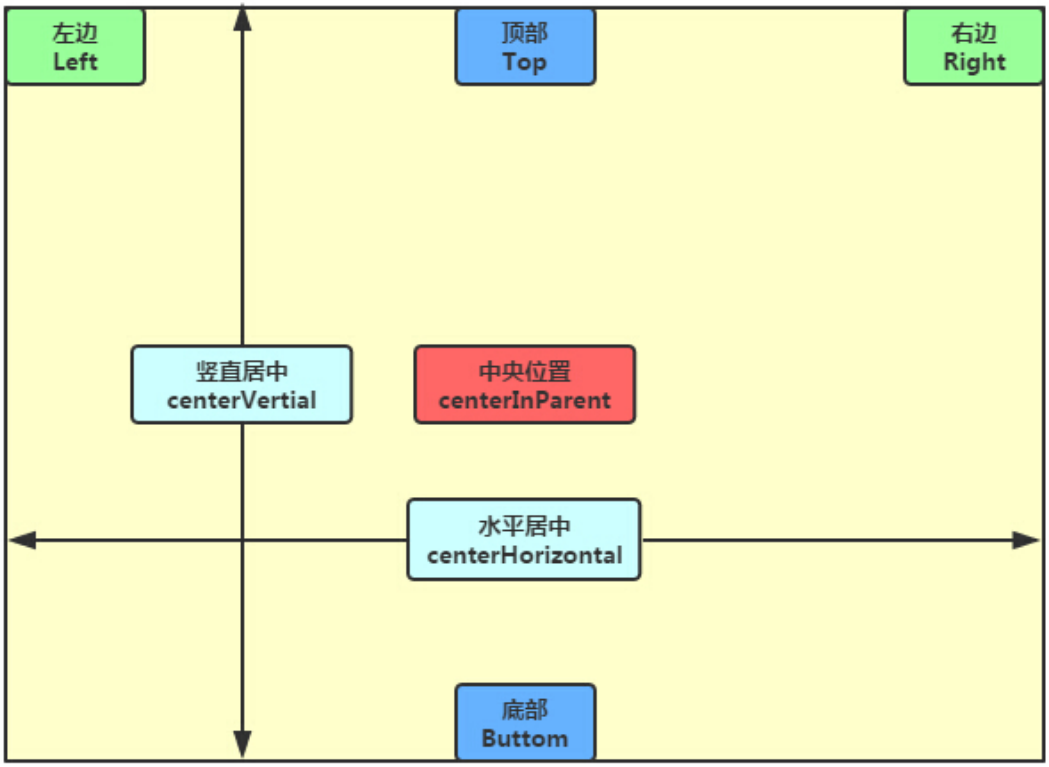
属性名称	属性含义
android:layout_centerInParent="true"	居中布局
android:layout_centerVertical="true"	垂直居中布局
android:layout_centerHorizontal="true"	水平居中布局

- 间隔

属性名称	属性含义

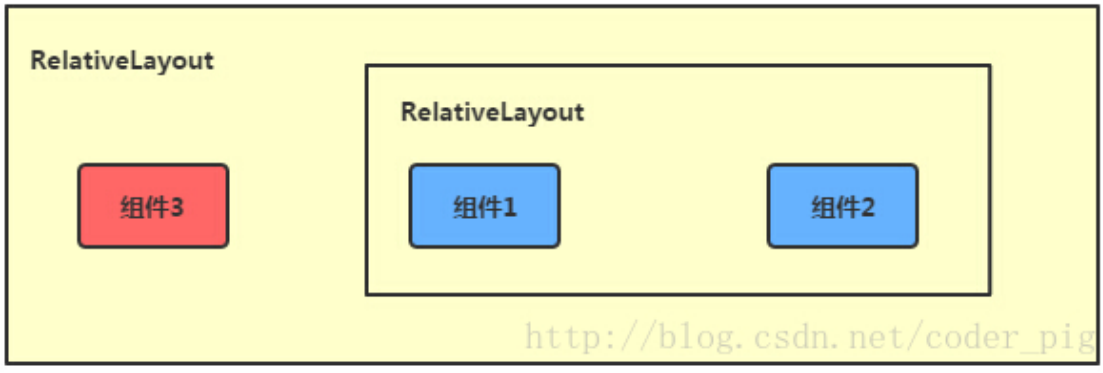
属性名称	属性含义
android:layout_marginBottom=""	离某元素底边缘的距离
android:layout_marginLeft=""	离某元素左边缘的距离
android:layout_marginRight =""	离某元素右边缘的距离
android:layout_marginTop=""	离某元素上边缘的距离
android:layout_paddingBottom=""	往内部元素底边缘填充距离
android:layout_paddingLeft=""	往内部元素左边缘填充距离
android:layout_paddingRight =""	往内部元素右边缘填充距离
android:layout_paddingTop=""	往内部元素右边缘填充距离

- 父容器定位属性示意图



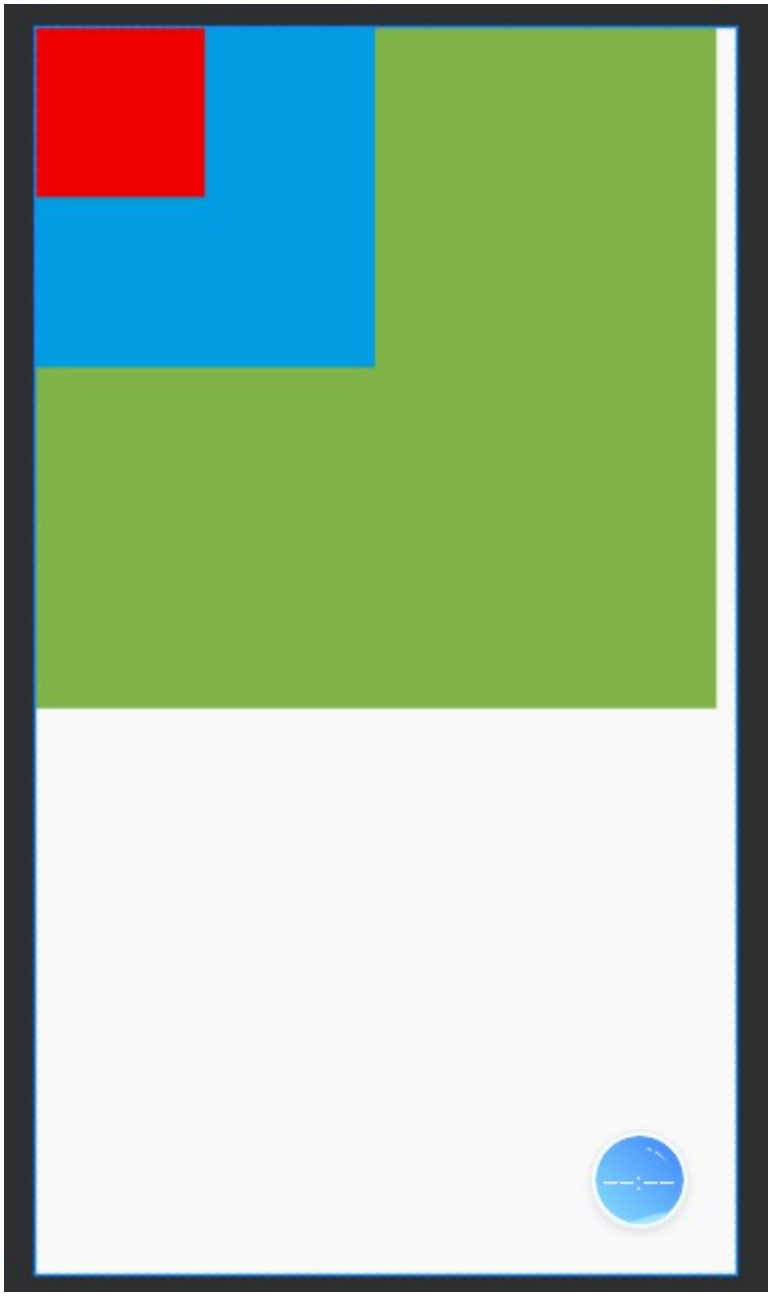
根据父容器定位示意图 http://blog.csdn.net/coder_pig

- 根据兄弟组件定位



3.3 帧布局 - FrameLayout

- 最简单的一种布局，没有任何定位方式，当我们往里面添加控件的时候，会默认把他们放到这块区域的左上角，帧布局的大小由控件中最大的子控件决定，如果控件的大小一样大的话，那么同一时刻就只能看到最上面的那个组件，后续添加的控件会覆盖前一个



3.4 网格布局 GridLayout

- 属性说明：

名称	含义
android:columnCount	列数
android:rowCount	行数
android:layout_columnSpan	横跨的列数
android:layout_rowSpan	横跨的行数

04 常用组件

4.1 TextView

TextView (文本框)，用于显示文本的一个控件。

- 文本的字体尺寸单位为`sp`：
- **sp: scaled pixels(放大像素)**. 主要用于字体显示。
- 文本常用属性：

属性名	作用
id	为TextView设置一个组件id，根据id，我们可以在Java代码中通过findViewById()的方法获取到该对象，然后进行相关属性的设置
layout_width	组件的宽度
layout_height	组件的高度
gravity	设置控件中内容的对齐方向，TextView中是文字，ImageView中是图片等等
text	设置显示的文本内容，一般我们是把字符串写到string.xml文件中，然后通过@String/xxx取得对应的字符串内容的
textColor	设置字体颜色，同上，通过colors.xml资源来引用
textStyle	设置字体风格，三个可选值：**normal**(无效果)，**bold**(加粗)，**italic**(斜体)
textSize	字体大小，单位一般是用sp
background	控件的背景颜色，可以理解为填充整个控件的颜色，可以是图片
autoLink	识别链接类型 （web, email, phone ,map ,none, all）

- 文本设置边框
  - 实现原理：

    编写一个**ShapeDrawable的资源文件**！然后TextView将`background`设置为这个drawable资源即可
  - **ShapeDrawable**的资源文件
    - <**solid** android:color = "xxx"> 这个是设置背景颜色的
    - <**stroke** android:width = "xdp" android:color="xxx"> 这个是设置边框的粗细,以及边框颜色的
    - <**padding** androidLbottom = "xdp"...> 这个是设置边距的
    - <**corners** android:topLeftRadius="10px"...> 这个是设置圆角的
    - <**gradient**> 这个是设置渐变色的,可选属性有: **startColor**:起始颜色 **endColor**:结束颜色 **centerColor**:中间颜色 **angle**:方向角度,等于0时,从左到右,然后逆时针方向转,当angle = 90度时从下往上 **type**:设置渐变的类型
      - 编写**矩形边框**的Drawable：

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

```
<shape
xmlns:android="http://schemas.android.com/apk/res/android" >
    <!-- 设置一个黑色边框 -->
    <stroke android:width="2px" android:color="#000000"/>
    <!-- 渐变 -->
    <gradient
        android:angle="270"
        android:endColor="#C0C0C0"
        android:startColor="#FCD209" />
    <!-- 设置一下边距,让空间大一点 -->
    <padding
        android:left="5dp"
        android:top="5dp"
        android:right="5dp"
        android:bottom="5dp"/>
</shape>
```

- 编写**圆角矩形边框**的Drawable

```
<?xml version="1.0" encoding="utf-8"?>
<shape
xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- 设置透明背景色 -->
    <solid android:color="#87CEEB" />
    <!-- 设置一个黑色边框 -->
    <stroke
        android:width="2px"
        android:color="#000000" />
    <!-- 设置四个圆角的半径 -->
    <corners
        android:bottomLeftRadius="10px"
        android:bottomRightRadius="10px"
        android:topLeftRadius="10px"
        android:topRightRadius="10px" />
    <!-- 设置一下边距,让空间大一点 -->
    <padding
        android:bottom="5dp"
        android:left="5dp"
        android:right="5dp"
        android:top="5dp" />
</shape>
```

-

- 带图片(drawableXxx)的TextView

属性名	作用
android:drawableLeft	文本左边设置图片
android:drawableRight	文本右边设置图片
android:drawableBottom	文本下边设置图片
android:drawableTop	文本上边设置图片

  - 应用场景

  - 属性使用:

```
<RelativeLayout
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"
    tools:context="com.jay.example.test.MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:drawableTop="@drawable/show1"
        android:drawableLeft="@drawable/show1"
        android:drawableRight="@drawable/show1"
        android:drawableBottom="@drawable/show1"
        android:drawablePadding="10dp"
        android:text="张全蛋" />

</RelativeLayout>
```

4.2 EditText(输入框)

- EditText 输入框，集成与TextView, 也继承其属性
- EditText 特有属性

属姓名	说明

属姓名	说明
android:hint	默认提示文本
android:textColorHint	默认提示文本的颜色
android:selectAllOnFocus	布尔值。点击输入框获得焦点后，获取到输入框中所有的文本内容
android:inputType	对输入的数据进行限制
android:minLines	设置最小行数
android:maxLines	设置最大行数 *PS:当输入内容超过maxline,文字会自动向上滚动！！*
android:singleLine	只允许单行输入，而且不会滚动
android:textScaleX	设置字与字的水平间隔
android:textScaleY	设置字与字的垂直间隔
android:capitalize	**sentences**：仅第一个字母大写； **words**：每一个单词首字母大小，用空格区分单词；**characters:**每一个英文字母都大写

- **文本类型，多为大写、小写和数字符号**

```
android:inputType="none"
android:inputType="text"
android:inputType="textCapCharacters"
android:inputType="textCapWords"
android:inputType="textCapSentences"
android:inputType="textAutoCorrect"
android:inputType="textAutoComplete"
android:inputType="textMultiLine"
android:inputType="textImeMultiLine"
android:inputType="textNoSuggestions"
android:inputType="textUri"
android:inputType="textEmailAddress"
android:inputType="textEmailSubject"
android:inputType="textShortMessage"
android:inputType="textLongMessage"
android:inputType="textPersonName"
android:inputType="textPostalAddress"
android:inputType="textPassword"
android:inputType="textVisiblePassword"
android:inputType="textWebEditText"
android:inputType="textFilter"
android:inputType="textPhonetic"
```

- **数值类型**

```
android:inputType="number"
android:inputType="numberSigned"
android:inputType="numberDecimal"
android:inputType="phone"//拨号键盘
android:inputType="datetime"
android:inputType="date"//日期键盘
android:inputType="time"//时间键盘
```

- 设置EditText获得焦点，同时弹出小键盘

```
edit.requestFocus(); //请求获取焦点
edit.clearFocus(); //清除焦点
```

> 低版本的系统直接requestFocus就会自动弹出小键盘了
>
> 稍微高一点的版本则需要我们手动地去弹键盘：
>
> 第一种：
>
> ```
> InputMethodManager imm = (InputMethodManager)
> getSystemService(Context.INPUT_METHOD_SERVICE);
> imm.toggleSoftInput(0, InputMethodManager.HIDE_NOT_ALWAYS);
> ```
>
> 第二种：
>
> ```
> InputMethodManager imm = (InputMethodManager)
> getSystemService(Context.INPUT_METHOD_SERVICE);
> imm.showSoftInput(view,InputMethodManager.SHOW_FORCED);
> imm.hideSoftInputFromWindow(view.getWindowToken(), 0); //强制隐藏键盘
> ```

- EditText光标位置的控制

```
setSelection();//一个参数的是设置光标位置的，两个参数的是设置起始位置与结束位置的中间括的部分，即部分选中
```

4.3 Button(按钮)

- `Button` 控件继承 `TextView`，拥有 `TextView` 的属性
- `StateListDrawable` 简介

  `StateListDrawable` 是Drawable资源的一种，可以根据不同的状态，设置不同的图片效果，关键节点 `< selector >`，我们只需要将Button的 `background` 属性设置为该drawable资源即可轻松实现，按下 按钮时不同的按钮颜色或背景！

属性名	说明
drawable	引用的Drawable位图,我们可以把他放到最前面,就表示组件的正常状态~
state_focused	是否获得焦点
state_window_focused	是否获得窗口焦点
state_enabled	控件是否可用
state_checkable	控件可否被勾选
state_checked	控件是否被勾选
state_selected	控件是否被选择,针对有滚轮的情况
state_pressed	控件是否被按下
state_active	控件是否处于活动状态
state_single	控件包含多个子控件时,确定是否只显示一个子控件
state_first	控件包含多个子控件时,确定第一个子控件是否处于显示状态
state_middle	控件包含多个子控件时,确定中间一个子控件是否处于显示状态
state_last	控件包含多个子控件时,确定最后一个子控件是否处于显示状态

- `btn_bg1.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@color/color1" android:state_pressed="true" />
    <item android:drawable="@color/color4" android:state_enabled="false" />
    <item android:drawable="@color/color3" />
</selector>
```

- `layout_btn.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingTop="50dp">

    <Button
        android:id="@+id/btnOne"
        android:layout_width="match_parent"
        android:layout_height="64dp"
        android:background="@drawable/btn_bg1"
        android:text="按钮"
        android:textColor="#ffffff"
        android:textSize="20sp"
        android:textStyle="bold" />

    <Button
```

```
        android:id="@+id/btnTwo"
        android:layout_width="match_parent"
        android:layout_height="64dp"
        android:text="按钮不可用"
        android:textColor="#000000"
        android:textSize="20sp"
        android:textStyle="bold" />
</LinearLayout>
```

- `MainActivity.java`

```
public class MainActivity extends Activity {
    private Button btnOne,btnTwo;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btnOne = (Button) findViewById(R.id.btnOne);
        btnTwo = (Button) findViewById(R.id.btnTwo);
        btnTwo.setOnClickListener(new OnClickListener() {  //按钮绑定点击事件
            @Override
            public void onClick(View v) {
                if(btnTwo.getText().toString().equals("按钮不可用")){
                    btnOne.setEnabled(false);
                    btnTwo.setText("按钮可用");
                }else{
                    btnOne.setEnabled(true);
                    btnTwo.setText("按钮不可用");
                }
            }
        });
    }
}
```

4.4 ImageView(图像视图)

`ImageView` 见名知意，就是用来显示图像的一个View或者说控件

需掌握的知识点:

> 1. ImageView的src属性和blackground的区别;
> 2. adjustViewBounds设置图像缩放时是否按长宽比
> 3. scaleType设置缩放类型
> 4. 最简单的绘制圆形的ImageView

- src属性和background属性的区别

在API文档中我们发现ImageView有两个可以设置图片的属性，分别是：src和background
常识：
① background通常指的都是背景,而src指的是内容!!
② 当使用src填入图片时,是按照图片大小直接填充,并不会进行拉伸,而使用background填入图片,则是会根据ImageView给定的宽度来进行拉伸

案例:

```
<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="vertical"
    tools:context="com.jay.example.imageviewdemo.MainActivity" >

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/pen" />

    <ImageView
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:background="@drawable/pen" />

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/pen" />

    <ImageView
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:src="@drawable/pen" />

</LinearLayout>
```

- Java代码中设置blackground和src属性:

```
前景(对应src属性):setImageDrawable();
背景(对应background属性):setBackgroundDrawable();
```

- 两者结合妙用

```
   <ImageView
        android:layout_gravity="center"
        android:padding="20dp"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:background="@drawable/shape_bg"
        android:src="@mipmap/pen" />
```

- scaleType 属性 `android:scaleType`

android:scaleType用于设置显示的图片如何缩放或者移动以适应ImageView的大小 Java代码中可以通过imageView.setScaleType(ImageView.ScaleType.CENTER);来设置~ 可选值如下：

fitXY:对图像的横向与纵向进行独立缩放,使得该图片完全适应ImageView,但是图片的横纵比可能会发生改变
fitStart:保持纵横比缩放图片,知道较长的边与Image的编程相等,缩放完成后将图片放在ImageView的左上角
fitCenter:同上,缩放后放于中间;
fitEnd:同上,缩放后放于右下角;
center:保持原图的大小，显示在ImageView的中心。当原图的size大于ImageView的size，超过部分裁剪处理。
centerCrop:保持横纵比缩放图片,知道完全覆盖ImageView,可能会出现图片的显示不完全
centerInside:保持横纵比缩放图片,直到ImageView能够完全地显示图片
matrix:默认值，不改变原图的大小，从ImageView的左上角开始绘制原图， 原图超过ImageView的部分作裁剪处理

1. `fitEnd`, `fitStart`, `fitCenter`

```
<ImageView
        android:background="#ffc"
        android:layout_width="300dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:scaleType="fitStart"
        android:src="@mipmap/ic_launcher" />
```

2. `centerCrop`与`centerInside`

```
centerCrop:按横纵比缩放,直接完全覆盖整个ImageView
centerInside:按横纵比缩放,使得ImageView能够完全显示这个图片
```

3. `fitXY`

```
不按比例缩放图片，目标是把图片塞满整个View
```

4. `matrix`

```
从ImageView的左上角开始绘制原图，原图超过ImageView的部分作裁剪处理
```

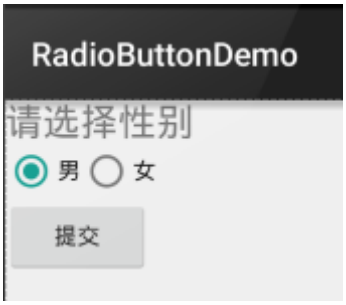
5. `center`

```
保持原图的大小，显示在ImageView的中心。当原图的size大于ImageView的size，超过部分裁剪处理。
```

4.5 RadioButton(单选按钮)&Checkbox(复选框)

1. `RadioButton`(单选按钮) 基本用法与事件处理:

> 如题单选按钮，就是只能够选中一个，所以我们需要把RadioButton放到RadioGroup按钮组中，从而实现 单选功能！先熟悉下如何使用RadioButton，一个简单的性别选择的例子： 另外我们可以为外层RadioGroup设置orientation属性然后设置RadioButton的排列方式，是竖直还是水平~




```
<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="vertical"
    tools:context=".MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="请选择性别"
        android:textSize="23dp"
        />

    <RadioGroup
        android:id="@+id/radioGroup"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <RadioButton
            android:id="@+id/btnMan"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="男"
            android:checked="true"/>

        <RadioButton
            android:id="@+id/btnWoman"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="女"/>
    </RadioGroup>

    <Button
        android:id="@+id/btnpost"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="提交"/>

</LinearLayout>
```

获得选中的值：这里有两种方法

第一种是为`RadioButton`设置一个事件监听器`setOnCheckChangeListener`

```
RadioGroup radgroup = (RadioGroup) findViewById(R.id.radioGroup);
        //第一种获得单选按钮值的方法
        //为radioGroup设置一个监听器:setOnCheckedChanged()
        radgroup.setOnCheckedChangeListener(new OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(RadioGroup group, int checkedId) {
                RadioButton radbtn = (RadioButton) findViewById(checkedId);
                Toast.makeText(getApplicationContext(), "按钮组值发生改变,你选了" +
radbtn.getText(), Toast.LENGTH_LONG).show();
            }
        });
```

PS：另外有一点要切记，要为每个`RadioButton`添加一个id，不然单选功能不会生效！！！

第二种方法是通过单击其他按钮获取选中单选按钮的值，当然我们也可以直接获取，这个看需求~

```
Button btnchange = (Button) findViewById(R.id.btnpost);
        RadioGroup radgroup = (RadioGroup) findViewById(R.id.radioGroup);
        //为radioGroup设置一个监听器:setOnCheckedChanged()
        btnchange.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                for (int i = 0; i < radgroup.getChildCount(); i++) {
                    RadioButton rd = (RadioButton) radgroup.getChildAt(i);
                    if (rd.isChecked()) {
                        Toast.makeText(getApplicationContext(), "点击提交按钮,获取
你选择的是:" + rd.getText(), Toast.LENGTH_LONG).show();
                        break;
                    }
                }
            }
        });
```

代码解析： 这里我们为提交按钮设置了一个`setOnClickListener`事件监听器,每次点击的话遍历一次RadioGroup判断哪个按钮被选中我们可以通过下述方法获得RadioButton的相关信息!

- **getChildCount**( )获得按钮组中的单选按钮的数目;
- **getChinldAt**(i):根据索引值获取我们的单选按钮
- **isChecked**( ):判断按钮是否选中

2.`CheckBox`(复选框)

> 如题复选框，即可以同时选中多个选项，至于获得选中的值，同样有两种方式： 1.为每个CheckBox添加事件：setOnCheckedChangeListener 2.弄一个按钮，在点击后，对每个checkbox进行判断:isChecked();


```
public class MainActivity extends AppCompatActivity implements
View.OnClickListener,CompoundButton.OnCheckedChangeListener{

    private CheckBox cb_one;
    private CheckBox cb_two;
    private CheckBox cb_three;
    private Button btn_send;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        cb_one = (CheckBox) findViewById(R.id.cb_one);
        cb_two = (CheckBox) findViewById(R.id.cb_two);
        cb_three = (CheckBox) findViewById(R.id.cb_three);
        btn_send = (Button) findViewById(R.id.btn_send);

        cb_one.setOnCheckedChangeListener(this);
        cb_two.setOnCheckedChangeListener(this);
        cb_three.setOnCheckedChangeListener(this);
        btn_send.setOnClickListener(this);

    }

    @Override
    public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
       if(compoundButton.isChecked())
Toast.makeText(this,compoundButton.getText().toString(),Toast.LENGTH_SHORT).show
();
    }

    @Override
    public void onClick(View view) {
        String choose = "";
```

```
        if(cb_one.isChecked())choose += cb_one.getText().toString() + "";
        if(cb_two.isChecked())choose += cb_two.getText().toString() + "";
        if(cb_three.isChecked())choose += cb_three.getText().toString() + "";
        Toast.makeText(this,choose,Toast.LENGTH_SHORT).show();
    }
}
```

自定义点击效果

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:state_enabled="true"
        android:state_checked="true"
        android:drawable="@mipmap/checked"/>
    <item
        android:state_enabled="true"
        android:state_checked="false"
        android:drawable="@mipmap/uncheck" />
</selector>
```

写好后，我们有两种方法设置，也可以说一种吧！你看看就知道了~

①android:button属性设置为上述的selctor

```
android:button="@drawable/checkbox"
```

②在style中定义一个属性，然后通过android style属性设置，先往style添加下述代码：

```
    <style name="MyCheckBox"
parent="@android:style/Widget.CompoundButton.CheckBox">
        <item name="android:button">@drawable/checkbox</item>
    </style>
```

然后布局那里:

```
style="@style/MyCheckBox"
```

修改文字与选择框的距离

```
android:background="@null"
android:paddingLeft="20dp"
```

4.6 开关按钮ToggleButton和开关Switch

1. `ToggleButton`

属性名	说明

属性名	说明
android:disabledAlpha	设置按钮在禁用时的透明度
android:textOff	按钮没有被选中时显示的文字
android:textOn	按钮被选中时显示的文字 另外，除了这个我们还可以自己写个selector，然后设置下Background属性即可

2. `Switch`

属性名	说明
android:showText	设置on/off的时候是否显示文字,boolean
android:splitTrack	是否设置一个间隙，让滑块与底部图片分隔,boolean
android:switchMinWidth	设置开关的最小宽度
android:switchPadding	设置滑块内文字的间隔
android:switchTextAppearance	设置开关的文字外观
android:textOff	按钮没有被选中时显示的文字
android:textOn	按钮被选中时显示的文字
android:textStyle	文字风格，粗体，斜体写划线那些
android:track	底部的图片
android:thumb	滑块的图片
android:typeface	设置字体，默认支持这三种:sans, serif, monospace;除此以外还可以使用 其他字体文件(***.ttf**)

4.7 ProgressBar进度条

- 常用属性

```
android:max：进度条的最大值
android:progress：进度条已完成进度值
android:progressDrawable：设置轨道对应的Drawable对象
android:indeterminate：如果设置成true，则进度条不精确显示进度
android:indeterminateDrawable：设置不显示进度的进度条的Drawable对象
android:indeterminateDuration：设置不精确显示进度的持续时间
android:secondaryProgress：二级进度条，类似于视频播放的一条是当前播放进度，一条是缓冲进度，前者通过progress属性进行设置！
```

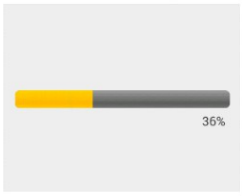
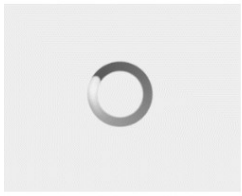
对应的再Java中我们可调用下述方法：

```
getMax()：返回这个进度条的范围的上限
getProgress()：返回进度
getSecondaryProgress()：返回次要进度
incrementProgressBy(int diff)：指定增加的进度
isIndeterminate()：指示进度条是否在不确定模式下
setIndeterminate(boolean indeterminate)：设置不确定模式下
```

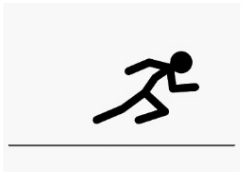
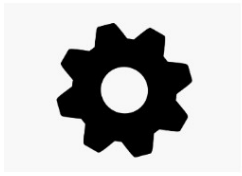
设置ProgressBar的样式，不同的样式会有不同的形状和模式：

Widget.ProgressBar.Horizontal
横向进度条（精确模式或模糊模式，这取决于Android:indeterminate）。
Widget.ProgressBar
中号的圆形进度条（模糊模式）。
Widget.ProgressBar.Small
小号的圆形进度条（模糊模式）。
Widget.ProgressBar.Large
大号的圆形进度条（模糊模式）。
Widget.ProgressBar.Inverse
中号的圆形进度条（模糊模式），该样式适用于亮色背景（例如白色）。
Widget.ProgressBar.Small.Inverse
小号的圆形进度条（模糊模式），该样式适用于亮色背景（例如白色）。
Widget.ProgressBar.Large.Inverse

1.1 标准的ProgressBar

精确模式	模糊模式（圆形）	模糊模式（横向）
		

1.2 自定义的ProgressBar

奔跑的小人	旋转的齿轮	反转的齿轮
		

4.8 SeekBar拖动条

```
android:max="100" //滑动条的最大值
android:progress="60" //滑动条的当前值
android:secondaryProgress="70" //二级滑动条的进度
android:thumb = "@mipmap/sb_icon" //滑块的drawable
```

接着要说下SeekBar的事件了，**SeekBar.OnSeekBarChangeListener** 我们只需重写三个对应的方法：

```
onProgressChanged：进度发生改变时会触发
onStartTrackingTouch：按住SeekBar时会触发
onStopTrackingTouch：放开SeekBar时触发
```

SeekBar定制

1.滑块状态Drawable：sb_thumb.xml

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

2.条形栏Bar的Drawable：sb_bar.xml

这里用到一个layer-list的drawable资源！其实就是层叠图片，依次是:背景，二级进度条，当前进度：

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@android:id/background">
        <shape>
            <solid android:color="#FFFFD042" />
        </shape>
    </item>
    <item android:id="@android:id/secondaryProgress">
        <clip>
            <shape>
                <solid android:color="#FFFFFFFF" />
            </shape>
        </clip>
    </item>
    <item android:id="@android:id/progress">
        <clip>
            <shape>
                <solid android:color="#FF96E85D" />
            </shape>
        </clip>
    </item>
</layer-list>
```

3.然后布局引入SeekBar后，设置下progressDrawable与thumb即可

```
<SeekBar
        android:id="@+id/sb_normal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:maxHeight="5.0dp"
        android:minHeight="5.0dp"
        android:progressDrawable="@drawable/sb_bar"
        android:thumb="@drawable/sb_thumb"/>
```

4.9 ScrollView(滚动条)

```
我们可以直接利用ScrollView给我们提供的:fullScroll()方法:
scrollView.fullScroll(ScrollView.FOCUS_DOWN);滚动到底部
scrollView.fullScroll(ScrollView.FOCUS_UP);滚动到顶部
```

隐藏滑块：
android:scrollbars="none"

设置滚动速度：
继承ScrollView，然后重写一个 public void fling (int velocityY)的方法：

```
@Override
public void fling(int velocityY) {
    super.fling(velocityY / 2);     //速度变为原来的一半
}
```

tips: ScrollView控件中只能包含一个View或一个ViewGroup

```
public class ScrollViewActivity extends AppCompatActivity implements
View.OnClickListener {
    private Button btn_down;
    private Button btn_up;
    private ScrollView scrollView;
    private TextView txt_show;

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


    private void bindViews() {
        btn_down = (Button) findViewById(R.id.btn_down);
        btn_up = (Button) findViewById(R.id.btn_up);
        scrollView = (ScrollView) findViewById(R.id.scrollView);
        txt_show = (TextView) findViewById(R.id.txt_show);
        btn_down.setOnClickListener(this);
        btn_up.setOnClickListener(this);

        StringBuilder sb = new StringBuilder();
        for (int i = 1; i <= 100; i++) {
            sb.append("我是一条文本内容 * " + i + "\n");
        }
        txt_show.setText(sb.toString());

    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btn_down:
                scrollView.fullScroll(ScrollView.FOCUS_DOWN);
                break;
            case R.id.btn_up:
                scrollView.fullScroll(ScrollView.FOCUS_UP);
                break;
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <Button
        android:id="@+id/btn_down"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="滚动到底部" />

    <Button
        android:id="@+id/btn_up"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="滚动到顶部" />

    <ScrollView
        android:id="@+id/scrollView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1">

        <TextView
            android:id="@+id/txt_show"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="" />

    </ScrollView>
</LinearLayout>
```

4.10 Date & Time组件

1. TextClock(文本时钟)

> TextClock是在Android 4.2(API 17)后推出的用来替代DigitalClock的一个控件!
> TextClock可以以字符串格式显示当前的日期和时间，因此推荐在Android 4.2以后使用TextClock。
> 这个控件推荐在24进制的android系统中使用，TextClock提供了两种不同的格式， 一种是在24进制中显示时间和日期，另一种是在12进制中显示时间和日期。大部分人喜欢默认的设置。

> 另外他给我们提供了下面这些方法，对应的还有get方法：
>
> | Attribute Name | Related Method | Description |
> | --- | --- | --- |
> | android:format12Hour | setFormat12Hour(CharSequence) | 设置12时制的格式 |
> | android:format24Hour | setFormat24Hour(CharSequence) | 设置24时制的格式 |
> | android:timeZone | setTimeZone(String) 设置时区 | |

```xml
<TextClock
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:format12Hour="MM/dd/yy h:mmaa"/>
    <TextClock
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:format12Hour="MMM dd, yyyy h:mmaa"/>
    <TextClock
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:format12Hour="MMMM dd, yyyy h:mmaa"/>
    <TextClock
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:format12Hour="E, MMMM dd, yyyy h:mmaa"/>
    <TextClock
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:format12Hour="EEEE, MMMM dd, yyyy h:mmaa"/>
    <TextClock
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:format12Hour="Noteworthy day: 'M/d/yy"/>
```

2. AnalogClock(模拟时钟)

```
android:dial  //表背景图片
android:hand_hour  //表时针图片
android:hand_minute  //分时针图片
```

```
<AnalogClock
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:dial="@mipmap/ic_c_bg"
        android:hand_hour="@mipmap/zhen_shi"
        android:hand_minute="@mipmap/zhen_fen" />
```

3. Chronometer(计时器)

```
就是一个简单的计时器
```

```
<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="vertical"
    tools:context=".MainActivity">


    <Chronometer
        android:id="@+id/chronometer"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textColor="#ff0000"
        android:textSize="60dip" />

    <LinearLayout
        android:layout_width="fill_parent"
```

```
        android:layout_height="wrap_content"
        android:layout_margin="10dip"
        android:orientation="horizontal">

        <Button
            android:id="@+id/btnStart"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="开始记时" />

        <Button
            android:id="@+id/btnStop"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="停止记时" />

        <Button
            android:id="@+id/btnReset"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="重置" />

        <Button
            android:id="@+id/btn_format"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="格式化" />
    </LinearLayout>

</LinearLayout>
```

```
public class MainActivity extends AppCompatActivity implements
View.OnClickListener,Chronometer.OnChronometerTickListener{

    private Chronometer chronometer;
    private Button btn_start,btn_stop,btn_base,btn_format;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        initView();
    }

    private void initView() {
        chronometer = (Chronometer) findViewById(R.id.chronometer);
        btn_start = (Button) findViewById(R.id.btnStart);
        btn_stop = (Button) findViewById(R.id.btnStop);
        btn_base = (Button) findViewById(R.id.btnReset);
        btn_format = (Button) findViewById(R.id.btn_format);

        chronometer.setOnChronometerTickListener(this);
        btn_start.setOnClickListener(this);
        btn_stop.setOnClickListener(this);
```

```
        btn_base.setOnClickListener(this);
        btn_format.setOnClickListener(this);

    }

    @Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.btnStart:
                chronometer.start();// 开始计时
                break;
            case R.id.btnStop:
                chronometer.stop();// 停止计时
                break;
            case R.id.btnReset:
                chronometer.setBase(SystemClock.elapsedRealtime());// 复位
                break;
            case R.id.btn_format:
                chronometer.setFormat("Time：%s");// 更改时间显示格式
                break;
        }
    }

    @Override
    public void onChronometerTick(Chronometer chronometer) {
        String time = chronometer.getText().toString();
        if(time.equals("00:00")){
            Toast.makeText(MainActivity.this,"时间到了
~",Toast.LENGTH_SHORT).show();
        }
    }
}
```

4. DatePicker(日期选择器)

```
android:calendarTextColor ：  日历列表的文本的颜色
android:calendarViewShown：是否显示日历视图
android:datePickerMode：组件外观，可选值:spinner，calendar 前者效果如下，默认效果是
后者
android:dayOfWeekBackground：顶部星期几的背景颜色
android:dayOfWeekTextAppearance：顶部星期几的文字颜色
android:endYear：去年(内容)比如2010
android:firstDayOfWeek：设置日历列表以星期几开头
android:headerBackground：整个头部的背景颜色
android:headerDayOfMonthTextAppearance：头部日期字体的颜色
android:headerMonthTextAppearance：头部月份的字体颜色
android:headerYearTextAppearance：头部年的字体颜色
android:maxDate：最大日期显示在这个日历视图mm / dd / yyyy格式
android:minDate：最小日期显示在这个日历视图mm / dd / yyyy格式
android:spinnersShown：是否显示spinner
android:startYear：设置第一年(内容)，比如19940年
android:yearListItemTextAppearance：列表的文本出现在列表中。
android:yearListSelectorColor：年列表选择的颜色
```

```
public class MainActivity extends AppCompatActivity implements
DatePicker.OnDateChangedListener{
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        DatePicker dp_test = (DatePicker) findViewById(R.id.dp_test);
        Calendar calendar = Calendar.getInstance();
        int year=calendar.get(Calendar.YEAR);
        int monthOfYear=calendar.get(Calendar.MONTH);
        int dayOfMonth=calendar.get(Calendar.DAY_OF_MONTH);
        dp_test.init(year,monthOfYear,dayOfMonth,this);
    }

    @Override
    public void onDateChanged(DatePicker view, int year, int monthOfYear,
int dayOfMonth) {
        Toast.makeText(MainActivity.this,"您选择的日期是："+year+"年"+
(monthOfYear+1)+"月"+dayOfMonth+"日!",Toast.LENGTH_SHORT).show();
    }
}
```

5. TimePicker(时间选择器)

```
public class MainActivity extends AppCompatActivity{

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        TimePicker tp_test = (TimePicker) findViewById(R.id.tp_test);
        tp_test.setOnTimeChangedListener(new
TimePicker.OnTimeChangedListener() {
            @Override
            public void onTimeChanged(TimePicker view, int hourOfDay, int
minute) {
                Toast.makeText(MainActivity.this,"您选择的时间
是："+hourOfDay+"时"+minute+"分!",Toast.LENGTH_SHORT).show();
            }
        });
    }

}
```

6. CalendarView(日历视图)

```
android:firstDayOfWeek：设置一个星期的第一天
android:maxDate ：最大的日期显示在这个日历视图mm / dd / yyyy格式
android:minDate：最小的日期显示在这个日历视图mm / dd / yyyy格式
android:weekDayTextAppearance：工作日的文本出现在日历标题缩写
```

```
public class MainActivity extends AppCompatActivity{
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        CalendarView cv_test = (CalendarView) findViewById(R.id.cv_test);
```

```
        cv_test.setOnDateChangeListener(new
CalendarView.OnDateChangeListener() {
            @Override
            public void onSelectedDayChange(CalendarView view, int year, int
month, int dayOfMonth) {
                Toast.makeText(MainActivity.this,"您选择的时间是："+ year +
"年" + month + "月" + dayOfMonth + "日",Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

4.11 ListView

```
BaseAdapter最基本的几个方法：
1. getCount     填充的数据集数
2. getItem        数据集中指定索引对应的数据项
3. getItemId     指定行所对应的ID
4. getView        每个Item所显示的类容
```

```
public class News {
    private String title;
    private String content;
    private int aIcon;

    public News() {
    }

    public News(String title, String content, int aIcon) {
        this.title = title;
        this.content = content;
        this.aIcon = aIcon;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getContent() {
        return content;
    }

    public void setContent(String content) {
        this.content = content;
    }

    public int getaIcon() {
        return aIcon;
    }

    public void setaIcon(int aIcon) {
        this.aIcon = aIcon;
```

```
    }
}
```

```
public class NewsAdapter extends BaseAdapter {

    private List<News> mData;
    private Context mContext;

    public NewsAdapter(List<News> mData, Context mContext) {
        this.mData = mData;
        this.mContext = mContext;
    }

    @Override
    public int getCount() {
        return mData.size();
    }

    @Override
    public Object getItem(int position) {
        return mData.get(position);
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        convertView =
LayoutInflater.from(mContext).inflate(R.layout.listview_item_layout, parent,
false);
        ImageView img_icon = (ImageView)
convertView.findViewById(R.id.img_icon);
        TextView title = (TextView) convertView.findViewById(R.id.tv_title);
        TextView content = (TextView) convertView.findViewById(R.id.tv_content);
        img_icon.setBackgroundResource(mData.get(position).getaIcon());
        title.setText(mData.get(position).getTitle());
        content.setText(mData.get(position).getContent());
        return convertView;
    }
}
```

```
public class ListViewActivity extends AppCompatActivity {
    private List<News> mData = null;
    private Context mContext;
    private NewsAdapter mAdapter = null;
    private ListView listView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.listview_layout);
        mContext = this;
        listView = (ListView) findViewById(R.id.listview);
```

```
        mData = new LinkedList<News>();
        for (int i = 0; i < 10; i++) {
            mData.add(new News("我是一个新闻标题---- " + i, "我是一个新闻内容---- " +
i, R.mipmap.news));
        }
        mAdapter = new NewsAdapter(mData, mContext);
        listView.setAdapter(mAdapter);
        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
                Toast.makeText(mContext, "点击了第" + position + "条数据",
Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="100dp"
    android:gravity="center_vertical"
    android:orientation="horizontal"
    android:padding="15dp">

    <ImageView
        android:id="@+id/img_icon"
        android:layout_width="130dp"
        android:layout_height="80dp"
        android:src="@mipmap/news"/>

    <RelativeLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_weight="1">

        <TextView
            android:id="@+id/tv_title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:text="我是一个新闻标题---- 1"
            android:textColor="#000000"
            android:textSize="18dp" />

        <TextView
            android:id="@+id/tv_content"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:text="我是新闻内容---- 1"
            android:textColor="#000000"
            android:textSize="14dp" />

    </RelativeLayout>
```

```
</LinearLayout>
```

`BaseAdapter`优化

```
一.复用ConvertView
  inflate()每次都要加载一次xml，其实这个convertView是系统提供给我们的可供复用的View 的缓存对象
  if(convertView == null){
        convertView = LayoutInflater.from(mContext).inflate(R.layout.item_list_animal,parent,false);
    }

二.ViewHolder重用组件
    static class ViewHolder{
    ImageView img_icon;
    TextView txt_aName;
    TextView txt_aSpeak;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        ViewHolder holder = null;
        if (convertView == null) {
            convertView = LayoutInflater.from(mContext).inflate(R.layout.listview_item_layout
                    , parent
                    , false);
            holder.img_icon = (ImageView) convertView.findViewById(R.id.img_icon);
            holder.title = (TextView) convertView.findViewById(R.id.tv_title);
            holder.content = (TextView) convertView.findViewById(R.id.tv_content);
            convertView.setTag(holder);
        } else {
            holder = (ViewHolder) convertView.getTag();
        }
        holder.img_icon.setBackgroundResource(mData.get(position).getaIcon());
        holder.title.setText(mData.get(position).getTitle());
        holder.content.setText(mData.get(position).getContent());
        return convertView;
    }
```

`ListView` item多布局实现

```
重写getItemViewType()方法对应View是哪个类别，以及getViewTypeCount()方法iew返回 总共多少个类别！然后再getView那里调用getItemViewType获得对应类别，再加载对应的View!
```

```
package com.ttit.helloworld.adapter;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
```

```java
import android.widget.TextView;

import com.ttit.helloworld.R;
import com.ttit.helloworld.entity.News;

import java.util.List;

public class NewsAdapter2 extends BaseAdapter {

    private List<News> mData;
    private Context mContext;
    //定义两个类别标志
    private static final int TYPE_NEWS_1 = 0;
    private static final int TYPE_NEWS_2 = 1;

    public NewsAdapter2(List<News> mData, Context mContext) {
        this.mData = mData;
        this.mContext = mContext;
    }

    @Override
    public int getCount() {
        return mData.size();
    }

    @Override
    public Object getItem(int position) {
        return mData.get(position);
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    //返回对应item布局类型
    @Override
    public int getItemViewType(int position) {
        if (position % 2 == 0) {
            return TYPE_NEWS_1;
        } else {
            return TYPE_NEWS_2;
        }
    }

    //总共多少item布局类型
    @Override
    public int getViewTypeCount() {
        return 2;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        int type = getItemViewType(position); //获取布局类型
        ViewHolder holder1 = null;
        ViewHolder2 holder2 = null;
        if (convertView == null) {
            switch (type) {
```

```
                case TYPE_NEWS_1:
                    holder1 = new ViewHolder();
                    convertView =
LayoutInflater.from(mContext).inflate(R.layout.listview_item_layout
                            , parent
                            , false);
                    holder1.img_icon = (ImageView)
convertView.findViewById(R.id.img_icon);
                    holder1.title = (TextView)
convertView.findViewById(R.id.tv_title);
                    holder1.content = (TextView)
convertView.findViewById(R.id.tv_content);
                    convertView.setTag(holder1);
                    break;
                case TYPE_NEWS_2:
                    holder2 = new ViewHolder2();
                    convertView =
LayoutInflater.from(mContext).inflate(R.layout.listview_item_layout2
                            , parent
                            , false);
                    holder2.img_icon = (ImageView)
convertView.findViewById(R.id.img_icon);
                    holder2.title = (TextView)
convertView.findViewById(R.id.tv_title);
                    holder2.content = (TextView)
convertView.findViewById(R.id.tv_content);
                    convertView.setTag(holder2);
                    break;
            }

        } else {
            switch (type) {
                case TYPE_NEWS_1:
                    holder1 = (ViewHolder) convertView.getTag();
                    break;
                case TYPE_NEWS_2:
                    holder2 = (ViewHolder2) convertView.getTag();
                    break;
            }
        }
        switch (type) {
            case TYPE_NEWS_1:

 holder1.img_icon.setBackgroundResource(mData.get(position).getaIcon());
                holder1.title.setText(mData.get(position).getTitle());
                holder1.content.setText(mData.get(position).getContent());
                break;
            case TYPE_NEWS_2:

 holder2.img_icon.setBackgroundResource(mData.get(position).getaIcon());
                holder2.title.setText(mData.get(position).getTitle());
                holder2.content.setText(mData.get(position).getContent());
                break;
        }
        return convertView;
    }

    static class ViewHolder {
```

```
        ImageView img_icon;
        TextView title;
        TextView content;
    }

    static class ViewHolder2 {
        ImageView img_icon;
        TextView title;
        TextView content;
    }
}
```

4.12 GridView网格视图

```
android:columnWidth：设置列的宽度
android:gravity：组件对其方式
android:horizontalSpacing：水平方向每个单元格的间距
android:verticalSpacing：垂直方向每个单元格的间距
android:numColumns：设置列数
android:stretchMode：设置拉伸模式，可选值如下： none：不拉伸；spacingWidth：拉伸元素间的间隔空隙 columnWidth：仅仅拉伸表格元素自身 spacingWidthUniform：既拉元素间距又拉伸他们之间的间隔空袭
```

`GridViewActivity.java`

```
package com.ttit.helloworld;

import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.GridView;
import android.widget.Toast;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import com.ttit.helloworld.adapter.GridViewAdpater;
import com.ttit.helloworld.entity.Icon;

import java.util.ArrayList;
import java.util.List;

public class GridViewActivity extends AppCompatActivity {

    private Context mContext;
    private GridView grid_photo;
    private GridViewAdpater mAdapter = null;
    private List<Icon> mData = null;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.gridview_layout);
        mContext = this;
        //视图层V
```

```
        grid_photo = (GridView) findViewById(R.id.gridView);
        //数据源M
        mData = new ArrayList();
        mData.add(new Icon(R.mipmap.iv_icon_1, "图标1"));
        mData.add(new Icon(R.mipmap.iv_icon_2, "图标2"));
        mData.add(new Icon(R.mipmap.iv_icon_3, "图标3"));
        mData.add(new Icon(R.mipmap.iv_icon_4, "图标4"));
        mData.add(new Icon(R.mipmap.iv_icon_5, "图标5"));
        mData.add(new Icon(R.mipmap.iv_icon_6, "图标6"));
        mData.add(new Icon(R.mipmap.iv_icon_7, "图标7"));
        //控制层C
        mAdapter = new GridViewAdpater(mData, mContext);

        grid_photo.setAdapter(mAdapter);
        //点击事件
        grid_photo.setOnItemClickListener(new AdapterView.OnItemClickListener()
{
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
                Toast.makeText(mContext, "你点击了~" + position + "~项",
Toast.LENGTH_SHORT).show();
            }
        });

    }
}
```

4.13 Spinner列表选项框

android:dropDownHorizontalOffset：设置列表框的水平偏移距离
android:dropDownVerticalOffset：设置列表框的水平竖直距离
android:dropDownSelector：列表框被选中时的背景
android:dropDownWidth：设置下拉列表框的宽度
android:gravity：设置里面组件的对其方式
android:popupBackground：设置列表框的背景
android:prompt：设置对话框模式的列表框的提示信息(标题)，只能够引用string.xml 中的资源id,而不能直接写字符串
android:spinnerMode：列表框的模式，有两个可选值： dialog：对话框风格的窗口 dropdown：下拉菜单风格的窗口(默认)
可选属性：android:entries：使用数组资源设置下拉列表框的列表项目

4.14 ExpandableListView可折叠列表

实现ExpandableAdapter的三种方式
1. 扩展BaseExpandableListAdpter实现ExpandableAdapter。
2. 使用SimpleExpandableListAdpater将两个List集合包装成ExpandableAdapter
3. 使用simpleCursorTreeAdapter将Cursor中的数据包装成SimpleCuroTreeAdapter 本节示例使用的是第一个，扩展BaseExpandableListAdpter，我们需要重写该类中的相关方法， 下面我们通过一个代码示例来体验下！

有一点要注意的是，重写isChildSelectable()方法需要返回true，不然不会触发子Item的点击事件

4.15 Toast 吐司

Android用于提示信息的一个控件

4.16 AlertDialog对话框

```
Step 1：创建AlertDialog.Builder对象；
Step 2：调用setIcon()设置图标，setTitle()或setCustomTitle()设置标题；
Step 3：设置对话框的内容：setMessage()还有其他方法来指定显示的内容；
Step 4：调用setPositive/Negative/NeutralButton()设置：确定，取消，中立按钮；
Step 5：调用create()方法创建这个对象，再调用show()方法将对话框显示出来；
```

```
java设计模式：建造者模式-Builder模式
```

4.17 PopupWindow 悬浮框

```
与AlertDialog区别：
本质区别为：AlertDialog是非阻塞式对话框：AlertDialog弹出时，后台还可以做事情；而PopupWindow是阻塞式对话框：PopupWindow弹出时，程序会等待，在PopupWindow退出前，程序一直等待，只有当我们调用了dismiss方法的后，PopupWindow退出，程序才会向下执行。这两种区别的表现是：AlertDialog弹出时，背景是黑色的，但是当我们点击背景，AlertDialog会消失，证明程序不仅响应AlertDialog的操作，还响应其他操作，其他程序没有被阻塞，这说明了AlertDialog是非阻塞式对话框；PopupWindow弹出时，背景没有什么变化，但是当我们点击背景的时候，程序没有响应，只允许我们操作PopupWindow，其他操作被阻塞。
```

```
setContentView(View contentView)：设置PopupWindow显示的View
getContentView()：获得PopupWindow显示的View
showAsDropDown(View anchor)：相对某个控件的位置（正左下方），无偏移
showAsDropDown(View anchor, int xoff, int yoff)：相对某个控件的位置，有偏移
showAtLocation(View parent, int gravity, int x, int y)： 相对于父控件的位置（例如正中央Gravity.CENTER，下方Gravity.BOTTOM等），可以设置偏移或无偏移 PS:parent这个参数只要是activity中的view就可以了！
setWidth/setHeight：设置宽高，也可以在构造方法那里指定好宽高， 除了可以写具体的值，还可以用WRAP_CONTENT或MATCH_PARENT， popupWindow的width和height属性直接和第一层View相对应。
setFocusable(true)：设置焦点，PopupWindow弹出后，所有的触屏和物理按键都由PopupWindows 处理。其他任何事件的响应都必须发生在PopupWindow消失之后，（home 等系统层面的事件除外）。 比如这样一个PopupWindow出现的时候，按back键首先是让PopupWindow消失，第二次按才是退出 activity，准确的说是想退出activity你得首先让PopupWindow消失，因为不并是任何情况下按back PopupWindow都会消失，必须在PopupWindow设置了背景的情况下 。
setAnimationStyle(int)：设置动画效果
```

05 UI项目实战

写一个简单西瓜视频的UI界面