第 3 章 Android 常用布局

目前使用 Android Studio 开发设计 UI(User Interface,用户接口)时还无法提供类似 Visual Studio 所见即所得的图形界面设计方式,但依靠线性布局(LinearLayout)、表格布局(TableLayout)、相对布局(RelativeLayout)、帧布局(FrameLayout)、绝对布局(AbsoluteLayout)、网格布局(GridLayout)和约束布局(ConstraintLayout)等已经能开发出各式各样 UI 界面。本章节通过案例来学习各种布局的特点和相关属性设置。

在 Android 的 UI 开发中需要了解长度单位的几种表示方式。

视频讲解

3.1 Android 长度单位

Android 布局设计的长度单位没有完全统一。常见的单位有 px、dp、sp、pt、mm 和 in 共 6 种。在布局文件的长度相关属性值中输入数字后,弹出智能提示中的长度单位,如图 3-1 所示。智能弹出提示框中会显示 6 种长度单位供开发人员选择。

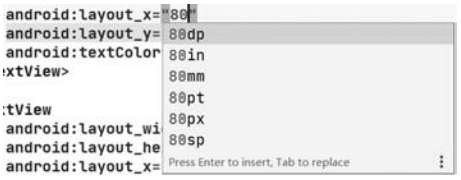

图 3-1 智能提示中的长度单位

以下是与长度相关的技术术语。

(1) px:即像素(pixels),1px 代表屏幕上一个物理的像素点。

(2) dp:独立像素密度(Density Independent Pixels),早期叫 dip),与像素无关。

(3) sp:主要用于设置字体尺寸,会随着系统的字体大小而改变,即同样大小的 dp 和 sp 字体,在 Android 设置中改变字体大小后,以 sp 为单位的字体会随系统字体大小改变而改变,以 dp 为单位的字体大小不会改变。正常字体 1dp=1sp,大字体和超大字体 1sp>1dp。以下是布局文件代码。

【main.xml】

```
01  <?xml version = "1.0" encoding = "UTF - 8"?>
02  < LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
03       android:layout_width = "match_parent"
04       android:layout_height = "match_parent"
05       android:orientation = "vertical">
06
07       < TextView
08            android:layout_width = "wrap_content"
```

```
09            android:layout_height = "wrap_content"
10            android:text = "Hello World! 你好,安卓!18sp"
11            android:textSize = "18sp" />
12
13        < TextView
14            android:layout_width = "wrap_content"
15            android:layout_height = "wrap_content"
16            android:text = "Hello World! 你好,安卓!18dp"
17            android:textSize = "18dp" />
18    </LinearLayout >
```

以上代码按系统默认字体大小的效果如图 3-2 所示。

系统字体改成大字体后的效果如图 3-3 所示。

(4) in：英寸，1in＝2.54cm，一般用于屏幕对角线尺寸单位。

(5) pt：磅，1in/72 的长度，1pt＝1in * 2.54cm/72in≈0.035cm。

(6) 分辨率：如果屏幕的分辨率是 1080 * 1920，是指水平方向上的像素数是 1080px，垂直方向上像素数是 1920px，屏幕分辨率如图 3-4 所示，根据勾股定律对角线则为 2203px。

图 3-2　系统默认字体大小的效果

图 3-3　系统字体改成大字体后的效果

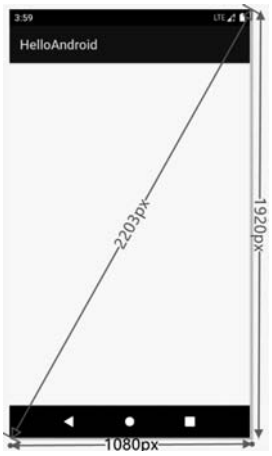

图 3-4　屏幕分辨率

(7) 屏幕像素密度：图 3-4 的对角线的像素数为 2203px，如果是 5 英寸屏(指对角线尺寸)，屏幕像素密度为 2203÷5＝440；如果是 6 寸屏，屏幕像素密度为 2203÷6＝367。如此一来就会有很多不同的屏幕像素密度，同样的图片在屏幕中显示所占比例也就不同。为此 Android 引入像素密度与逻辑密度的概念。

(8) 像素密度与逻辑密度：像素密度(dot per inch，dpi)就是每英寸的像素点数，不同的像素密度对应不同的 Android dpi 名称。如像素密度是 160，意思是每英寸像素数 160px，对应的 dpi 名称为 mdpi。Android Studio 在构建项目时会自动建立一个名为 HelloAndroid\app\src\main\res\mipmap-mdpi 的目录，目录中默认提供的图片分辨率为 48×48。Android Studio 同时也会建立其他 Android dpi 名称的目录，让不同分辨率、不同尺寸的 Android 设备自行调用不同目录中的图片文件，以保证不同参数的屏幕尽可能显示相似界面。像素密度是 40 的倍数。像素密度与逻辑密度如表 3-1 所示。

表 3-1　像素密度与逻辑密度

Android dpi 名称	分辨率	默认图片尺寸	像素密度	比例	逻辑密度
xxxhdpi	3840 * 2160	192 * 192	640	16	4
xxhdpi	1920 * 1080	144 * 144	480	12	3
xhdpi	1280 * 720	96 * 96	320	8	2
hdpi	480 * 800	72 * 72	240	6	1.5
mdpi	480 * 320	48 * 48	160	4	1
ldpi	320 * 240	36 * 36	120	3	0.75

逻辑密度是以 160dpi 为基准,其他像素密度与 160dpi 的比值,或者是像素密度对应的比例值除以 4,也是 dp 转 px 的系数。

dp 与 px 的换算关系为:**px=dp * dpi/160**,如 160dpi 的 mdpi,1dp * 160dpi/160=1px,对于基准 160dpi 而言,逻辑密度 density=1(160dpi/160dpi 或者比例 4/4)。又如 240dpi 的 hdpi,1dp * 240dpi/160=1.5px,对于基准 160dpi 而言,逻辑密度 density=1.5(240dpi/160dpi 或者比例为 6/4)。换言之,要将 mdpi 的图片在 hdpi 上也能同比例显示,只需将 mdpi 下的图片放大 1.5 倍即可。在实际的设计中还可以使用 Java 代码获取屏幕分辨率然后换算比例进行屏幕动态布局,以此保证在不同分辨率的屏幕下都能按同样比例显示。以下是获取屏幕相关尺寸参数代码。

【FirstActivity.java】

```
01   public class FirstActivity extends Activity
02   {
03
04        @Override
05        public void onCreate(Bundle savedInstanceState)
06        {
07             super.onCreate(savedInstanceState);
08             setContentView(R.layout.main);
09
10             String str = "";
11             DisplayMetrics dm = new DisplayMetrics();
12             dm = this.getApplicationContext().getResources().getDisplayMetrics();
13             str += "屏幕分辨率为:" + dm.widthPixels
                  + " * " + dm.heightPixels + "\n";
14             str += "水平方向分辨率:" + dm.widthPixels + "px\n";
15             str += "垂直方向分辨率:" + dm.heightPixels + "px\n";
16             str += "逻辑密度:" + dm.density + "\n";
17             str += "xdpi:" + dm.xdpi + "像素/英寸\n";
18             str += "ydpi:" + dm.ydpi + "像素/英寸\n";
19             Log.i("xj", str);
20        }
21   }
```

程序运行结果如下:

```
I:屏幕分辨率为:1080 * 1776
    水平方向分辨率:1080px
    垂直方向分辨率:1776px
    逻辑密度:3.0
    xdpi:480.0 像素/英寸
    ydpi:480.0 像素/英寸
```

将上面运行结果 xdpi 的 480 除以基准像素密度 160,得到的结果 3 就是逻辑密度。

视频讲解

3.2 线性布局

线性布局是 Android 早期开发版本的默认布局,使用 LinearLayout 标签,通过设置 android:orientation 属性值为 horizontal(水平)或 vertical(垂直)来将其内的控件按照水平方向或垂直方向依次排列。排列的控件不用指定位置的相关属性(简单即是美的体现),控件显示位置与在线性布局中出现的先后顺序相关。线性布局可以互相嵌套形成更复杂的结构。布局文件代码如下:

【main.xml】

```
01  <?xml version = "1.0" encoding = "UTF - 8"?>
02  < LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
03       xmlns:tools = "http://schemas.android.com/tools"
04       android:layout_width = "match_parent"
05       android:layout_height = "match_parent"
06       android:orientation = "vertical">
07
08       < LinearLayout
09            android:layout_width = "match_parent"
10            android:layout_height = "wrap_content"
11            android:orientation = "horizontal">
12
13            < TextView
14                 android:layout_width = "wrap_content"
15                 android:layout_height = "wrap_content"
16                 android:background = "@android:color/holo_red_dark"
17                 android:text = "第一列红色"
18                 android:textColor = "#ffffff"
19                 android:textSize = "25sp" />
20
21            < TextView
22                 android:layout_width = "wrap_content"
23                 android:layout_height = "wrap_content"
24                 android:background = "#00aa00"
25                 android:text = "绿色"
26                 android:textColor = "#ffffff"
27                 android:textSize = "25sp" />
28
```

```
29          <TextView
30              android:layout_width="200dp"
31              android:layout_height="wrap_content"
32              android:background="#0000aa"
33              android:text="第三列蓝色"
34              android:textColor="#ffffff"
35              android:textSize="25sp" />
36      </LinearLayout>
37
38      <LinearLayout
39          android:layout_width="match_parent"
40          android:layout_height="match_parent"
41          android:orientation="vertical">
42
43          <TextView
44              android:layout_width="wrap_content"
45              android:layout_height="wrap_content"
46              android:background="#0088ee"
47              android:text="第一行为 wrap_content"
48              android:textColor="#ffffff"
49              android:textSize="25sp" />
50
51          <TextView
52              android:layout_width="match_parent"
53              android:layout_height="wrap_content"
54              android:background="#191970"
55              android:text="第二行为 match_parent"
56              android:textColor="#ffffff"
57              android:textSize="25sp" />
58
59          <TextView
60              android:layout_width="wrap_content"
61              android:layout_height="wrap_content"
62              android:background="#11ff33"
63              android:text="第三行 wrap_content\n333333333333333333333333333333333333"
64              android:textColor="#ffffff"
65              android:textSize="25sp" />
66
67          <TextView
68              android:layout_width="match_parent"
69              android:layout_height="wrap_content"
70              android:background="#191970"
71              android:text="第四行 match_parent\n444444444444444444444444444444444444"
72              android:textColor="#ffffff"
73              android:textSize="25sp" />
74
75      </LinearLayout>
76
77  </LinearLayout>
```

线性布局的运行结果如图 3-5 所示。

从布局代码和组件树(见图 3-6)所示的结构可以看出,整个布局的结构为,最外层是一个垂直线性布局,内嵌一个水平线性布局(含 3 个 TextView)和一个垂直线性布局(含 4 个 TextView)。TextView 控件的 android:layout_width 属性常用值有 match_parent(早期版本为 fill_parent,将 TextView 控件宽度设置为父容器宽度)、wrap_content(按 TextView 的显示文本内容长度来设置宽度,如果文本内容显示宽度超过父容器宽度则将宽度设为父容器宽度,多余的文本在下一行中显示)和具体数值宽度(如第 30 行的 200dp)。android:layout_height 属性用于设置高度,与设置宽度概念类似。

图 3-5　线性布局的运行结果

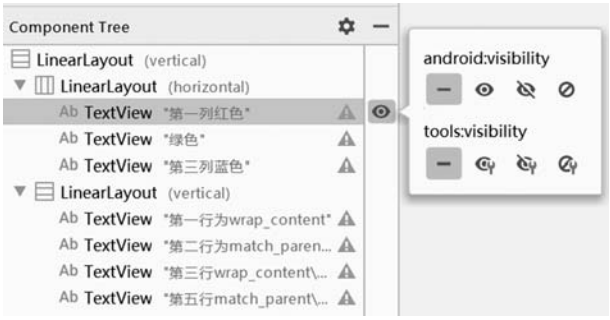

图 3-6　组件树

显示界面标题栏下的第一行分别显示红、绿、蓝 3 种颜色的 TextView,前两个 TextView 按文字长度显示,第三个 TextView 定义的宽度为 200dp,大于文字长度,所以 TextView 控件中文字离控件右边缘有一段距离。

后面几行显示了不同文字长度的 TextView 在 android:layout_width 宽度属性分别设置为 match_parent 和 wrap_content 时的显示差异。读者可修改代码观察运行结果中 TextView 的宽度和折行变化。

在图 3-6 的组件树右侧单击 图标按钮可以修改控件是否可见。 图标提示控件代码有警示信息。本案例中将 TextView 中的文字直接以字符串形式赋予了 android:text,而 Android Studio 建议采用“@string/字符串变量名”的方式定义 TextView 的值。在 Code 视图中警示方式会变为将相应属性背景色变为黄色,如图 3-7 所示。在 Design 视图中单击橙色三角形 ,在弹出信息栏中单击 Fix 按钮(如果是 Code 视图,则将光标置于字符串中,按 lt+Enter 快捷键,在上下文菜单中选择 Extract string resource 菜单),在弹出的界面中输入字符串变量名,系统自动在 strings.xml 中注册相应的字符串变量名和对应的字符串值。如果弹出的三角形是红色,代表控件的属性设置中有错误。开发人员可以用鼠标拖动组件树中的控件位置来改变控件显示顺序,Code 视图中的代码也会自动调整顺序。右击控件树中的线性布局,在弹出的快捷菜单中可选择转换为其他布局方式。

Code 视图如图 3-7 所示,android:background 属性用来设置背景色。android:textColor 属

性用来设置文字颜色,颜色值使用 6 位十六进制数表示,每 2 位为一组,分别表示红、绿、蓝,合成的颜色效果在左侧行号后显示为颜色方块或在 Design 视图中查看效果。

用鼠标双击 Code 视图行号后的颜色方块弹出调色盘,如图 3-8 所示,可实现可视化的颜色调配选择。如果在图 3-8 中选择 Resources 选项卡,可选择系统自定义的颜色,如第 16 行的"@android:color/holo_red_dark",代表使用 Android 自定义的颜色。Design 视图中显示的效果与实际运行结果可能会有差异,以实际运行结果为准。

```
46      <TextView
47          android:layout_width="wrap_content"
48          android:layout_height="wrap_content"
49          android:background="#0088ee"
50          android:text="第一行为wrap_content"
51          android:textColor="#ffffff"
52          android:textSize="25sp" />
```

图 3-7　Code 视图

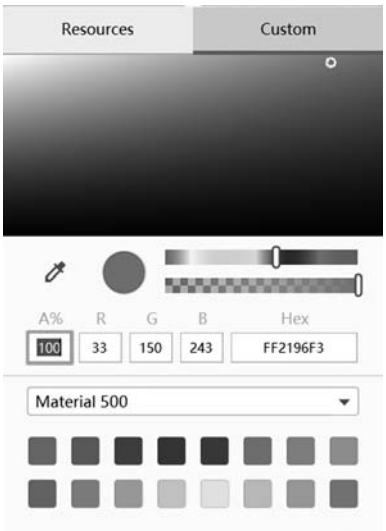

图 3-8　调色盘

3.3　边 线 和 角

所有的布局方式(含 LinearLayout)设定的边界都是没有线条标识的,如果想给相关布局画出边线可以采用以下方式:

【main.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:background = "@drawable/shape_conner"
    android:gravity = "center"
    android:orientation = "vertical">

    < TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:background = "#ffffff"
        android:text = "@string/hello"
        android:textSize = "15dp" />
</LinearLayout >
```

在线性布局标签属性中添加第 5 行的代码，设置线性布局背景使用 drawable 目录下的 shape_conner.xml 文件（此时 shape_conner.xml 文件被当成一个图片文件使用）。

【shape_conner.xml】

```
01  <?xml version="1.0" encoding="UTF-8"?>
02  <shape xmlns:android="http://schemas.android.com/apk/res/android">
03      <!-- 内部背景色 -->
04      <solid android:color="#5f5fdc"/>
05      <!-- 边角半径 -->
06      <corners
07          android:bottomLeftRadius="30dp"
08          android:bottomRightRadius="30dp"
09          android:topLeftRadius="10dp"
10          android:topRightRadius="10dp"/>
11      <!-- 边线颜色和边线宽度 -->
12      <stroke
13          android:width="5dp"
14          android:color="#ff0000"/>
15  </shape>
```

第 4 行定义了背景色。

第 6～10 行定义了屏幕 4 个角的转角半径，如果半径设为 0 则为直角。

第 13 行定义了线条的宽度。

第 14 行定义了线条的颜色。

给布局添加边线和角的运行结果如图 3-9 所示。

图 3-9　给布局添加边线和角的运行结果

当前屏幕为圆角的手机越来越多，而大多数模拟器的 4 个角是直角，如果设计时就处理边角显示适配，则可以考虑采用此方法。

3.4 layout_weight


视频讲解

Android 布局中设置控件的宽度一般使用 android:layout_width 属性,有时会采用属性 android:layout_weight 与 android:layout_width 配合使用。android:layout_weight 的作用是设定同一父容器内控件的长度或宽度的占比。先通过一个布局代码来看实际运行结果,再分析与显示结果不同的原因。

【main.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">

    < LinearLayout
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:orientation = "horizontal">

        < Button
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_weight = "1"
            android:text = "按钮 1"
            android:textSize = "20sp" />

        < Button
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_weight = "2"
            android:text = "按钮 2"
            android:textSize = "20sp" />
    </LinearLayout >

    < LinearLayout
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:orientation = "horizontal">

        < Button
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
            android:layout_weight = "1"
            android:text = "按钮 3"
            android:textSize = "20sp" />

        < Button
```

```
40                android:layout_width = "match_parent"
41                android:layout_height = "wrap_content"
42                android:layout_weight = "2"
43                android:text = "按钮 4"
44                android:textSize = "20sp" />
45        </LinearLayout >
46
47    </LinearLayout >
```

整个布局的结构是一个垂直线性布局内嵌两个水平线性布局，每个水平布局中又放置两个按钮，“按钮 1”和“按钮 3”的 layout_weight 设为 1，“按钮 2”和“按钮 4”的 layout_weight 设为 2。“按钮 1”宽度占屏幕的 1/3，“按钮 2”宽度占屏幕的 2/3，这与设想的相符。“按钮 3”宽度占屏幕的 2/3，“按钮 4”宽度占屏幕的 1/3，这是怎么回事呢？产生差异的原因是按钮布局 android:layout_width 属性是 wrap_content 还是 match_parent。

(1) 当 android:layout_width＝"wrap_content"时(假设按钮的文本内容长度没有超过屏幕占比)，两个按钮占屏幕一行，每个按钮按各自占比设置宽度，如此例中 layout_weight 分别为 1 和 2，则总和为 3，“按钮 1”占 1/3，“按钮 2”占 2/3。

(2) 当 android:layout_width＝"match_parent"时，各按钮的宽度等于父容器宽度加上剩余空间的占比。设父容器宽度为 L，“按钮 3”和“按钮 4”的 android:layout_width＝"match_parent"，所以两个按钮宽度都应该为 L，剩余宽度就为父容器宽度减去两个按钮的宽度：L－(L＋L)＝－L。“按钮 1”占 1/3，所以“按钮 3”的实际宽度是 L(父容器宽度)＋(－L)(剩余宽度)＊1/3＝L＋(－L)＊1/3＝2L/3。同理，“按钮 4”的实际宽度为 L/3。

由此可以看出，Android 在长度设置上除了长度单位不同外，还要考虑不同属性之间的影响。layout_weight 属性并不能精确地控制控件的宽度(或高度)，还会受控件内文字长度的影响(即使文字长度未超过屏幕占比)。如果想精确控制各控件的长度对齐，需考虑使用其他布局。

图 3-10　layout_width 结合 layout_weight 运行结果

layout_width 结合 layout_weight 运行结果如图 3-10 所示。

【提问】 删除第 35 和 42 行将如何显示？

3.5　绝对布局

绝对布局使用 android:layout_x 和 android:layout_y 来设定屏幕水平方向和垂直方向坐标，这种定位方式简单直接，但对于不同分辨率的屏幕，绝对布局的显示效果会有差异，这也是不推荐使用绝对布局的原因。一种解决方式是先获取屏幕的分辨率，然后按照百分比计算绝对布局的 x、y 坐标。在 Android Studio 中查看绝对布局源码，单词 AbsoluteLayout

会出现一条中画线,将鼠标指针放在单词 AbsoluteLayout 上,弹出弃用提示,如图 3-11 所示。按下快捷键 Alt+Shift+Enter,AbsoluteLayout 标签属性中会自动添加一行属性 tools:ignore="Deprecated"来忽略弃用提示,此时会看到单词 AbsoluteLayout 上的中画线消失。“tools:”标识并不影响布局设计,只是对界面设计人员起到辅助的作用。

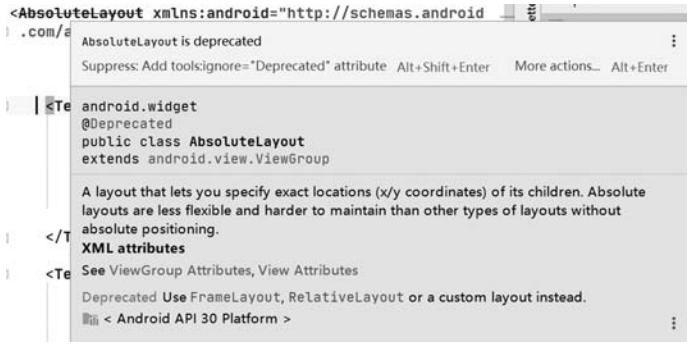

图 3-11　弃用提示

以下是绝对布局的源码和运行结果。

【main.xml】

```
<?xml version = "1.0" encoding = "UTF - 8"?>
< AbsoluteLayout 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:ignore = "Deprecated">

    < TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_x = "0px"
        android:layout_y = "0px"
        android:text = "@string/hello">
    </TextView >

    < TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_x = "80px"
        android:layout_y = "80px"
        android:text = "@string/action">
    </TextView >

    < TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_x = "150px"
        android:layout_y = "150px"
```

```
29          android:text = "@string/hello">
30      </TextView >
31
32      < TextView
33          android:layout_width = "wrap_content"
34          android:layout_height = "wrap_content"
35          android:layout_x = "140px"
36          android:layout_y = "145px"
37          android:text = "@string/collision"
38          android:textColor = "#ff00ff">
39      </TextView >
40
41      < TextView
42          android:layout_width = "wrap_content"
43          android:layout_height = "wrap_content"
44          android:layout_x = "0px"
45          android:layout_y = "750px"
46          android:text = "@string/lastLine"
47          android:textColor = "#000000">
48      </TextView >
49
50  </AbsoluteLayout >
```

第 11～12 行 TextView 的定位为(0,0),从运行结果上可以看出原点在标题栏下方的显示区左上角(不是整个屏幕的左上角)。

第 24～39 行中定义的两个 TextView 坐标非常接近,显示的文本也就有部分重叠。这是其他布局方式难以达到的效果(帧布局和约束布局除外)。这也是为什么还是有一部分开发人员喜欢使用绝对布局,特别是针对单一设备,此时不用考虑屏幕尺寸和分辨率带来的差异。

绝对布局运行结果如图 3-12 所示。

图 3-12　绝对布局运行结果

3.6 相对布局

在新版的 Android Studio 中,相对布局已经归入 Legacy 控件栏中,意味着以后可能会弃用。这里仍然讲解相对布局,其一是因为很多网络上的案例仍在使用相对布局;其二是相对布局中的一些属性和概念也可以用于其他布局中。相对布局的常用属性有以下 4 类。

(1) 当前控件与父容器的相对位置,属性值为 true 或者 false。与父容器相对位置如图 3-13 所示,粗线代表控件对齐的边。layout_alignParentStart 属性默认对应 layout_alignParentLeft,layout_alignParentEnd 属性默认对应 layout_alignParentRight,此时默认布局方向是 left-to-right。如果采用 right-to-left 布局方向,则 layout_alignParentStart 属性对应 layout_alignParentRight,layout_alignParentEnd 属性对应 layout_alignParentLeft。本书后续涉及带 Start 和 End 的属性按默认 left-to-right 布局方向解释为 Left 和 Right。新版 Android Studio 推荐使用 Start 和 End 替代 Left 和 Right。

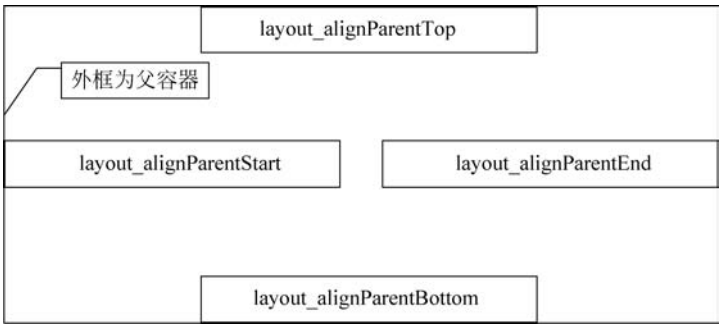

图 3-13 与父容器相对位置

(2) 当前控件与参考控件的相对位置,属性值必须为参考控件 id 的引用名"@id/id-name"。与参考控件相对位置如图 3-14 所示。

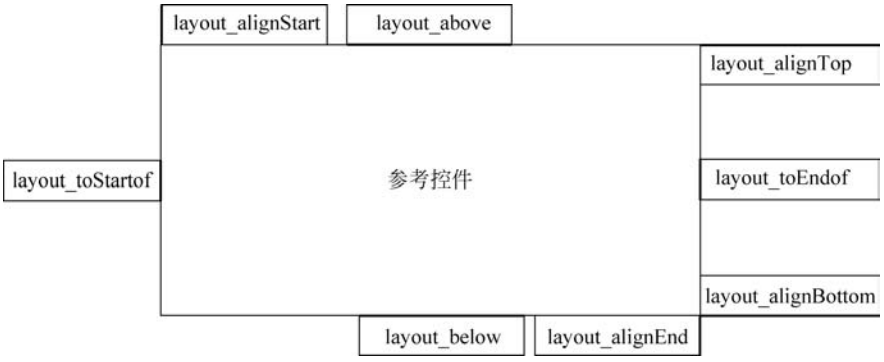

图 3-14 与参考控件相对位置

(3) 属性值是具体的长度或像素,如 android:layout_marginTop="100dp",指明当前控件离父容器上边缘 100dp 距离。

(4) 定义控件边界的空白宽度和控件内部填充宽度属性,相关属性如图 3-15 所示,属性值为长度或像素值。

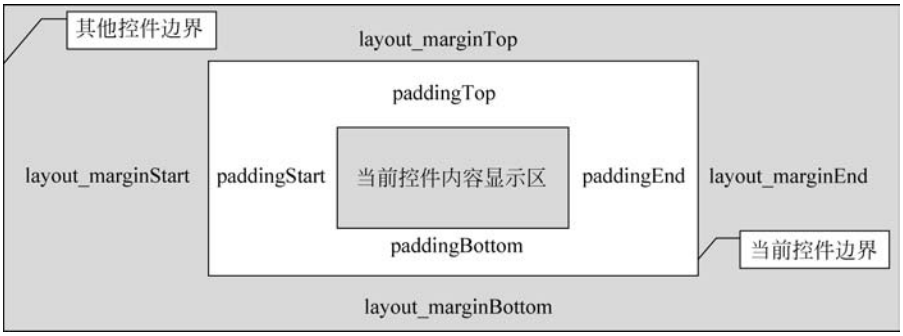

图 3-15　margin 与 padding

【main.xml】

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

    < TextView
        android:id = "@ + id/textView"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:layout_marginTop = "100dp"
        android:text = "@string/layout" />

    < EditText
        android:id = "@ + id/editText1"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:layout_below = "@ + id/textView"
        android:layout_alignParentRight = "true" />

    < Button
        android:id = "@ + id/button1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_below = "@ + id/editText1"
        android:layout_alignParentEnd = "true"
        android:layout_marginEnd = "80dp"
        android:text = "@string/button1" />

    < Button
        android:id = "@ + id/button2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignBottom = "@ + id/button1"
        android:layout_alignParentStart = "true"
        android:text = "@string/button2" />

```

```
37      <Button
38          android:id="@+id/button3"
39          android:layout_width="wrap_content"
40          android:layout_height="wrap_content"
41          android:layout_alignParentBottom="true"
42          android:layout_toEndOf="@+id/button2"
43          android:text="@string/button3" />
44
45  </RelativeLayout>
```

第 10 行指明 TextView 与父容器的上边缘间隔为 100dp。

第 24 行指明 button1 在文本框 editText1 的下方。第 25 行指明 button1 与父容器右对齐。第 26 行让 button1 右侧与对齐边线保持 80dp 的间隔。

第 33 行指明 button2 与 button1 底部对齐。第 34 行指明 button2 与父容器左对齐。

第 41 行指明 button3 对齐父容器的底部,第 42 行指明 button3 的左边与 button2 的右边缘对齐。相对布局运行结果如图 3-16 所示。

图 3-16　相对布局运行结果

案例库中还列出了使用 Java 代码来实现动态设定相对布局,由于还未讲解按钮监听器的使用,读者可在学习相应章节后自行查看、使用相应代码。

【提问】 第 42 行换成 android:layout_toEndOf="@+id/button1"会如何?

如果将第 42 行的 button2 改为 button1,运行后会发现 button3 不见了。这是因为 button1 在第 25 行指明是右对齐屏幕右边缘,虽然第 26 行留出了 80dp 的间隔,但其他控件如果与 button1 右边缘对齐,还是要以没有间隔的位置为准,所以 button3 的位置在屏幕右侧边缘之外导致看不见了。

视频讲解

3.7　帧　布　局

帧布局为每个加入其中的控件创建一个区域(称为一帧),这些帧会根据 layout_gravity 属性执行相应对齐。未设置 layout_gravity 属性值时,控件默认在父容器的左上角。符号"|"用于定义同时拥有多个属性值。layout_gravity 属性值定位示意图如图 3-17 所示。

外框为父容器

默认或者 start|top	top|center_horizontal	top|end
center_vertical|start	center_vertical|center_horizontal	center_vertical|end
bottom|start	bottom|center_horizontal	bottom|end

图 3-17　layout_gravity 属性值定位示意图

【main.xml】

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

    < TextView
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:text = "1111"
        android:textColor = "#ff00ff"
        android:textSize = "50sp" />

    < TextView
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:text = "2222"
        android:textColor = "@android:color/black"
        android:textSize = "40sp" />

    < TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"

        android:layout_gravity = "center_vertical|center_horizontal"
        android:background = "@android:color/holo_orange_light"
        android:text = "3333"
        android:textSize = "30sp"
        tools:layout_width = "wrap_content" />

    < TextView
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:background = "#666666"
        android:layout_gravity = "bottom"
        android:text = "4444"
        android:textColor = "#ffffff"
        android:textSize = "20sp" />
    < TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:background = "#666666"
        android:layout_gravity = "center|end"
        android:text = "5555"
        android:textColor = "#ffffff"
        android:textSize = "20sp" />
</FrameLayout >
```

第 7～19 行的两个 TextView 没有定义 layout_gravity,将叠加显示在父容器(在此例中 TextView 的父容器为 FrameLayout)的左上角。叠加的顺序是后定义的控件显示在之前定义的控件之上。

第 25 行 TextView 同时指定为垂直方向正中和水平方向正中,显示的效果是父容器的中心位置。

第 35 行定义为 bottom,此时变更为 bottom|end 也是相同的效果,因为当前 TextView 的 layout_width 属性定义为 match_parent,意味着 TextView 控件宽度与父容器等宽,所以再加上右对齐属性还是显示为与父容器等宽的右对齐。

案例中用两种方式(Android 自带颜色和十六进制)定义颜色,6 位十六进制数折合二进制是 24 位,也就是平时所说的 24 位色。帧布局运行结果如图 3-18 所示。

图 3-18　帧布局运行结果

视频讲解

3.8　表格布局

表格布局是按照行列的表格方式排列布局,其中 TableRow 用于同一行内多个控件对象的排列,如果没有定义 TableRow,则一个控件对象占用表格的一行。为了控制表格的拉伸和收缩,可设置以下属性。

(1) android:collapseColumns:设置需要被隐藏列的序号,相应列不可见。

(2) android:shrinkColumns:设置允许收缩列的序号。

(3) android:stretchColumns:设置允许拉伸列的序号。

【注】 列的序号是从 0 开始的。

下面的案例设计了一个规整的表格界面,代码如下:

【main.xml】

```
01  <?xml version = "1.0" encoding = "UTF - 8"?>
02  < TableLayout xmlns:android = "http://schemas.android.com/apk/res/android"
03       android:layout_width = "match_parent"
04       android:layout_height = "match_parent"
05       android:background = "＃C9E1F4"
06       android:stretchColumns = "0,1,2">
07
08       < TextView
09            android:background = "＃2241EC"
10            android:gravity = "center"
11            android:text = "学生信息表"
12            android:textColor = "＃FFFFFF"
13            android:textSize = "30dip" />
14       < TextView
15            android:background = "＃2196F3"
16            android:gravity = "center"
```

```
17            android:text = "男生信息"
18            android:textColor = "＃FFFFFF"
19            android:textSize = "30dip" />
20
21        < TableRow >
22
23            < TextView
24                android:layout_column = "0"
25                android:layout_margin = "4dip"
26                android:background = "＃F8F7EE"
27                android:gravity = "center"
28                android:text = "学号" />
29
30            < TextView
31                android:layout_column = "1"
32                android:layout_margin = "4dip"
33                android:background = "＃F8F7EE"
34                android:gravity = "center"
35                android:text = "姓名" />
36
37            < TextView
38                android:layout_margin = "4dip"
39                android:background = "＃F8F7EE"
40                android:gravity = "center"
41                android:text = "出生地" />
42        </TableRow >
43
44        < TableRow >
45
46            < TextView
47                android:layout_margin = "4dip"
48                android:background = "＃F8F7EE"
49                android:gravity = "center"
50                android:text = " 2021001 " />
51
52            < TextView
53                android:layout_margin = "4dip"
54                android:background = "＃F8F7EE"
55                android:gravity = "left"
56                android:text = "张三" />
57
58            < TextView
59                android:layout_margin = "4dip"
60                android:background = "＃F8F7EE"
61                android:gravity = "right"
62                android:text = "云南昆明" />
63        </TableRow >
64
65        < TableRow >
66
```

```
67            <TextView
68                android:layout_margin="4dip"
69                android:background="#F8F7EE"
70                android:gravity="center"
71                android:text=" 2021002 " />
72
73            <TextView
74                android:layout_margin="4dip"
75                android:background="#F8F7EE"
76                android:gravity="left"
77                android:text="李四" />
78
79            <TextView
80                android:layout_margin="4dip"
81                android:background="#F8F7EE"
82                android:gravity="right"
83                android:text="北京" />
84        </TableRow>
85
86        <TableRow>
87
88            <TextView
89                android:layout_margin="4dip"
90                android:background="#F8F7EE"
91                android:gravity="center"
92                android:text="2021003" />
93
94            <TextView
95                android:layout_margin="4dip"
96                android:background="#F8F7EE"
97                android:gravity="left"
98                android:text="王五" />
99
100           <TextView
101               android:layout_margin="4dip"
102               android:background="#F8F7EE"
103               android:gravity="right"
104               android:text="四川成都" />
105       </TableRow>
106   </TableLayout>
```

第 6 行指明表格布局的 0～2 列是可拉伸的,本案例中表格布局有 3 列,默认这 3 列平分父容器宽度。当某列有单元格的字符超出平分表格列格宽度时,此列的宽度会自动扩展,其他列相应收缩,直至其左侧列宽度等于文本宽度或者其右侧列被挤出父容器之外。如果删除此行,则后续 TableRow 中控件按指定宽度或默认 warp_conent 显示。

第 8～19 行定义的 TextView 没有在 TableRow 标签内。表格布局中没有在 TableRow 标签内的控件默认都独占一行,所以两个 TextView 分别占了两行。

第 21～42 行属于 TableRow 标签范围,TableRow 标签内定义的 TextView 都在同

一行。

第 27 行的 android:gravity 用于 TextView 控件内部的文字对齐，android:layout_gravity 用于当前控件对父容器的对齐。详细讲解参见“4.1.3 layout_gravity 与 gravity”中的案例。

表格布局运行结果如图 3-19 所示。

图 3-19　表格布局运行结果

下面的案例设计了一个不规则表格界面，代码如下：

【main.xml】

```
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:orientation = "vertical">

    <TableLayout
        android:id = "@ + id/tablelayout01"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:shrinkColumns = "1"
        android:stretchColumns = "2">

        <Button
            android:id = "@ + id/button01"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "独自一行，不在 TableRow 中" />

        <TableRow>

            <Button
                android:id = "@ + id/button02"
                android:layout_width = "wrap_content"
```

```
24                    android:layout_height = "wrap_content"
25                    android:text = "表 1" />
26
27               < Button
28                    android:id = "@ + id/button03"
29                    android:layout_width = "wrap_content"
30                    android:layout_height = "wrap_content"
31                    android:text = "允许被收缩允许被收缩允许被收缩" />
32
33               < Button
34                    android:id = "@ + id/button04"
35                    android:layout_width = "wrap_content"
36                    android:layout_height = "wrap_content"
37                    android:text = "允许被拉伸允许被拉伸" />
38          </TableRow >
39      </TableLayout >
40
41      < TableLayout
42          android:id = "@ + id/tablelayout02"
43          android:layout_width = "match_parent"
44          android:layout_height = "wrap_content"
45          android:collapseColumns = "1">
46
47          < TableRow >
48
49               < Button
50                    android:id = "@ + id/button05"
51                    android:layout_width = "wrap_content"
52                    android:layout_height = "wrap_content"
53                    android:text = "表 2" />
54
55               < Button
56                    android:id = "@ + id/button06"
57                    android:layout_width = "wrap_content"
58                    android:layout_height = "wrap_content"
59                    android:text = "被隐藏列" />
60
61               < Button
62                    android:id = "@ + id/button07"
63                    android:layout_width = "wrap_content"
64                    android:layout_height = "wrap_content"
65                    android:text = "我是第三列" />
66          </TableRow >
67      </TableLayout >
68
69      < TableLayout
70          android:id = "@ + id/tablelayout03"
71          android:layout_width = "match_parent"
72          android:layout_height = "wrap_content"
73          android:stretchColumns = "1">
```

```
74
75          <TableRow>
76
77              <Button
78                  android:id="@+id/button08"
79                  android:layout_width="wrap_content"
80                  android:layout_height="wrap_content"
81                  android:text="表 3" />
82
83              <Button
84                  android:id="@+id/button09"
85                  android:layout_width="wrap_content"
86                  android:layout_height="wrap_content"
87                  android:text="填满剩余空白" />
88          </TableRow>
89      </TableLayout>
90
91  </LinearLayout>
```

第 10 行指明第 2 列是可收缩的(从 0 开始计算)。

第 11 行指明第 3 列是可拉伸的。

第 33～37 行定义的按钮文字比较多,其在表格布局的第 3 列,按钮宽度根据文字长度而拉伸,相应定义为可收缩的第 2 列按钮的宽度被压缩,其超出按钮宽度的文字将折行显示。

自第 41 行起重新定义了一个表格布局,其中第 45 行定义第 2 列可折叠隐藏。第 55～59 行定义的按钮被隐藏,所以表 2 只显示了两列。

第 69 行定义新的表格布局,其中第 73 行定义为第 2 列可拉伸,在 TableRow 中定义了两个按钮,第 2 个按钮的宽度虽然设置为 wrap_content,但因为缺第 3 列导致可拉伸的第 2 列填满剩余表格行宽度。

不规则表格布局运行结果如图 3-20 所示。

图 3-20　不规则表格布局运行结果

3.9　网 格 布 局

在新版的 Android Studio 中,网格布局已经归入 Legacy 控件栏。官方更推荐表格布局作为类似场景中的布局。网格布局的默认行列高度和宽度是统一的,可以通过调整布局容器大小,设置 android:layout_width 或行列的权重来改变。以下案例设计一个简单计算器界面,具体代码如下:

```
01  <GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
02      android:layout_width="wrap_content"
```

```
    android:layout_height = "wrap_content"
    android:columnCount = "4"
    android:orientation = "horizontal"
    android:rowCount = "6">

    < EditText
        android:id = "@ + id/result"
        android:layout_columnSpan = "4"
        android:layout_gravity = "fill" />

    < Button
        android:id = "@ + id/one"
        android:text = "1" />

    < Button
        android:id = "@ + id/two"
        android:text = "2" />


    < Button
        android:id = "@ + id/three"
        android:text = "3" />

    < Button
        android:id = "@ + id/devide"
        android:text = "/" />

    < Button
        android:id = "@ + id/four"
        android:text = "4" />

    < Button
        android:id = "@ + id/five"
        android:text = "5" />

    < Button
        android:id = "@ + id/six"
        android:text = "6" />

    < Button
        android:id = "@ + id/multiply"
        android:text = " × " />

    < Button
        android:id = "@ + id/seven"
        android:text = "7" />

    < Button
        android:id = "@ + id/eight"
```

```
52            android:text = "8" />
53
54        < Button
55            android:id = "@ + id/nine"
56            android:text = "9" />
57
58        < Button
59            android:id = "@ + id/minus"
60            android:text = " - " />
61
62        < Button
63            android:id = "@ + id/zero"
64            android:layout_columnSpan = "2"
65            android:layout_gravity = "fill"
66            android:text = "0" />
67
68        < Button
69            android:id = "@ + id/point"
70            android:text = "." />
71
72        < Button
73            android:id = "@ + id/plus"
74            android:layout_rowSpan = "2"
75            android:layout_gravity = "fill"
76            android:text = " + " />
77
78        < Button
79            android:id = "@ + id/equal"
80            android:layout_columnSpan = "3"
81            android:layout_gravity = "fill"
82            android:text = " = " />
83    </GridLayout >
```

第 4 行定义网格布局有 4 列，第 6 行定义网格布局有 6 行，最后形成一个 6 行 4 列的网格。

第 10 行定义 EditText 可以拉伸 4 列宽度，再结合第 11 行实现 EditText 占 4 列宽度。同样第 74～75 行指明加号按钮占两行。

网格布局运行结果如图 3-21 所示。

从本案例可以看出，网格布局更适用于较为规整的行列表格，网格布局中的控件按指定的行列顺序依次排列，但相应的灵活性也比表格布局低。网格布局还存在控件间隙不一致的缺陷。

图 3-21　网格布局运行结果

3.10 约束布局

3.10.1 约束布局基础


视频讲解

从 Android Studio 2.3 版本起,约束布局是 Android Studio 布局文件的默认布局。其他布局方式在实现复杂一些的布局设计时存在多种或多个布局嵌套的情况,设备调用这样的布局文件就需要花费更多的时间。约束布局在灵活性和可视化方面比其他布局方式更胜一筹(与号称宇宙第一 IDE 的 Visual Studio 的图形化界面设计相比还有差距)。为减少布局嵌套,使用约束布局的属性更接近于相对布局属性。因此,很多资料在讲解约束布局时采用与相对布局类似的方式,不可避免地就要讲解一堆约束布局的定位属性。约束布局与其他布局的最大区别在于支持图形化拖放操作。可以在布局文件的 Design 视图中采用鼠标拖放操作结合属性栏窗口设置完成约束布局的界面设计,大幅简化布局代码输入和控件间定位关系的人为判断。

【注】 约束布局可以实现图形化拖放设计,但不是真正的所见即所得。Design 视图中的效果与实际运行结果还是有所不同的,经常出现的问题是定位属性设置错误或缺项。

选择布局文件 main.xml,在 Design 视图的工具栏中拖放两个 Button 按钮到 Design 界面中,如图 3-22 所示。

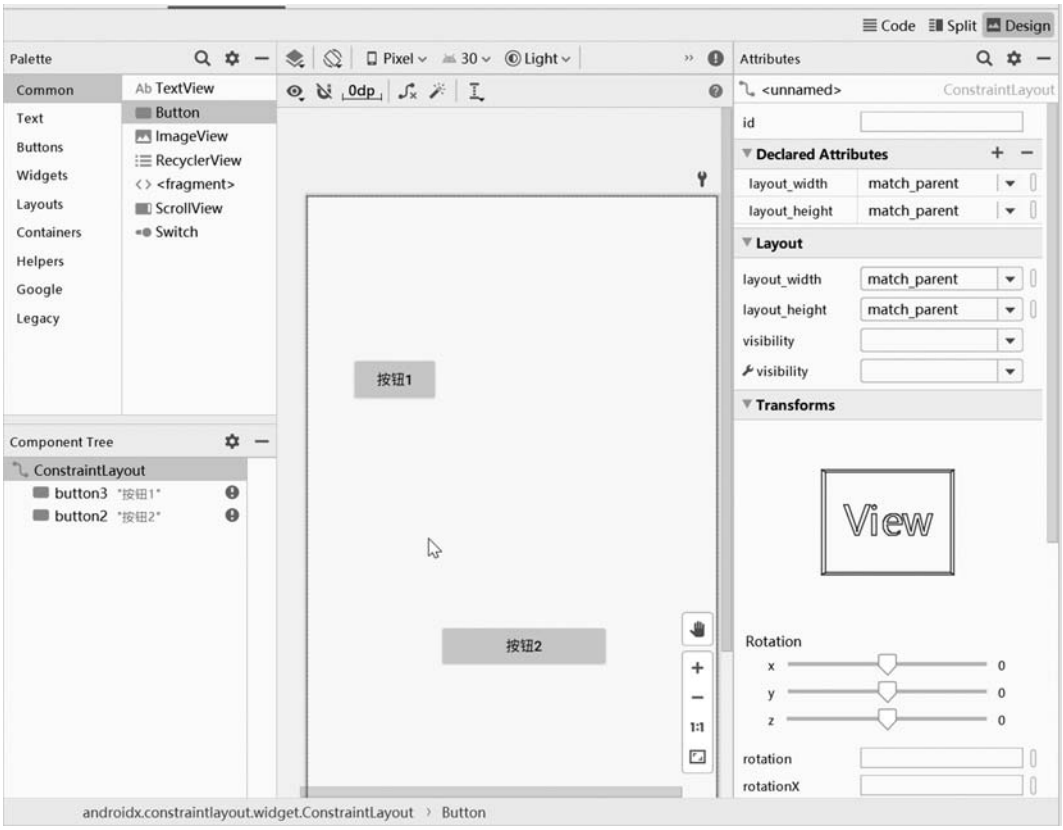

图 3-22 约束布局 Design 视图

此时由于未加入约束定位，组件树中的控件都会用红色惊叹号标识。运行程序，两个按钮会显示在屏幕左上角，坐标为(0,0)。

选中按钮控件，显示控件句柄，如图 3-23 所示。4 个角上的正方形句柄用于调整控件的尺寸，圆形句柄用于设置控件的定位。

图 3-23　控件句柄

在“按钮 1”左侧圆形句柄上按住鼠标左键并拖向屏幕左侧，此时“按钮 1”会自动靠到屏幕左侧，意味着“按钮 1”与屏幕左侧对齐。选中“按钮 1”再次向右拖动到如图 3-24 所示的位置，4 个方框圈出数字为 132，单位为 dp，代表“按钮 1”相对父容器(此时为屏幕左边缘)的距离为 132dp。也可以在属性栏或布局栏(图中右侧中间的 Constraint Widget)中修改数字改变距离值。以上操作对应两个属性：

```
app:layout_constraintStart_toStartOf = "parent"
android:layout_marginStart = "132dp"
```

注意，一个前缀是“app:”，另一个前缀是“android:”。前者引用 app 目录下 build.gradle 文件中 androidx.constraintlayout:constraintlayout 定义的属性；后者引用系统定义的属性。

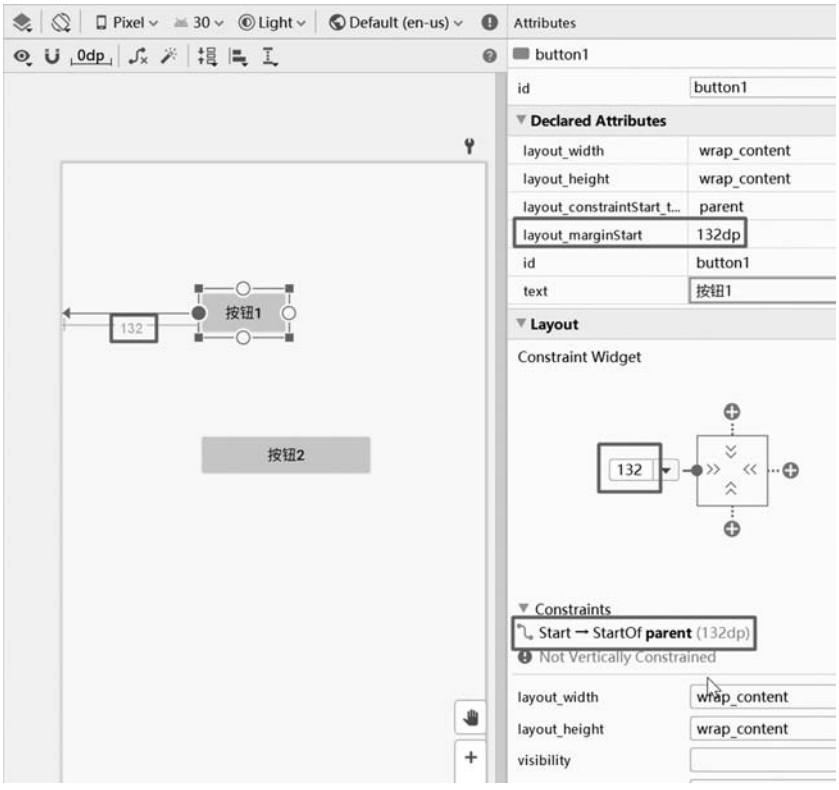

图 3-24　“按钮 1”相对父容器水平方向定位

同样，选择“按钮 1”上方的圆形句柄定位屏幕上边缘距离。重新运行程序，“按钮 1”将出现在设定的位置。“按钮 2”还是在屏幕左上角。可以采用同样的方式对“按钮 2”进行操作，也可以采用相对“按钮 1”的位置进行定位。例如，将“按钮 2”定位在“按钮 1”下方 50dp

位置,“按钮 2”右侧离屏幕右侧 100dp。为实现上述定位要求,先选中“按钮 2”上方的圆形句柄并拖动到“按钮 1”(此时“按钮 1”上会出现上、下两个圆形句柄)下方的圆形句柄,调整距离值为 50dp。如果进行此操作,鼠标左键释放时指向“按钮 1”区域而非圆形句柄,会弹出如图 3-25 的上下文菜单,可选择“按钮 2”是对齐“按钮 1”顶部还是底部。

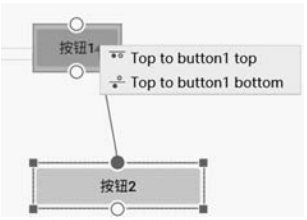

图 3-25 设定“按钮 2”相对“按钮 1”的垂直方向定位

选择“按钮 2”右侧的圆形句柄并拖动到屏幕右侧,调整距离值为 100dp,“按钮 2”的定位如图 3-26 所示。

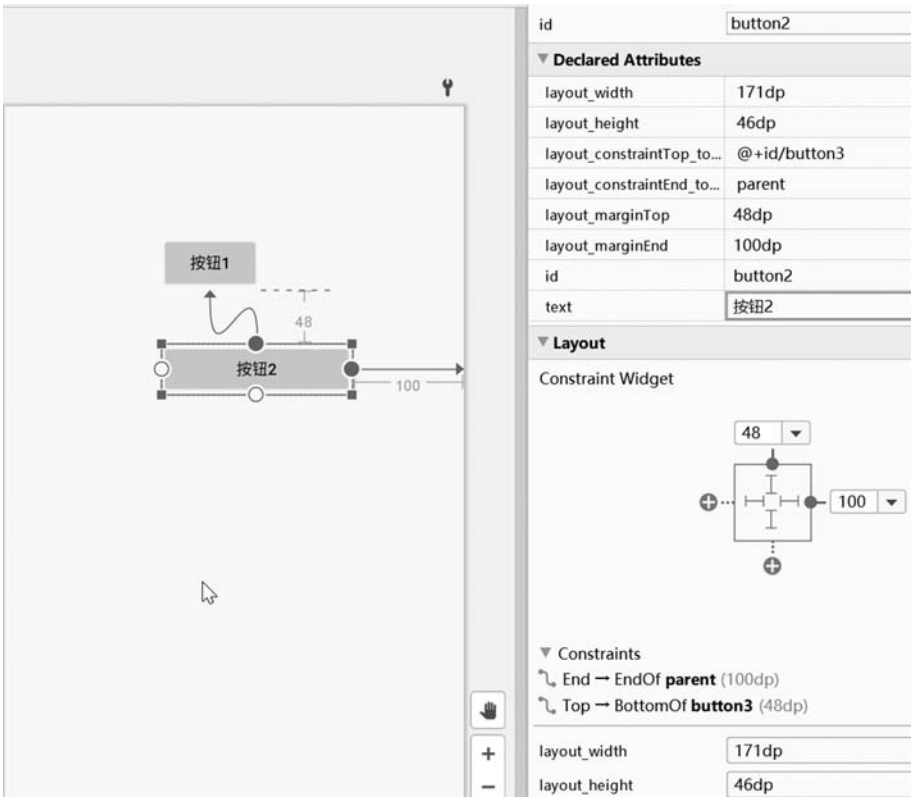

图 3-26 “按钮 2”的定位

约束布局运行结果如图 3-27 所示。

设计图与运行图相比,两个按钮的显示位置还是有差异,这是因为设计图并不是按实际设备的屏幕分辨率来设定的,距离真正的所见即所得还有一定的差距。当控件定位属性不全时,组件树会有红色圆形警告标识,Code 视图的控件标签也会标红,同时会多出一个前缀为“tools:”的属性。例如,tools:layout_editor_absoluteX="180dp"指明当前控件在 Design 视图中的 X 轴坐标是 180dp。此属性只在 Design 视图中起辅助定位时使用,在运行时还是被忽略的,即运行时控件在 X 轴坐标还是 0(对齐屏幕左侧)。

如果对“按钮 1”在水平方向分别将左边缘和右边缘依次定位到屏幕两侧,Android Studio 会将“按钮 1”在水平方向自动置中,定位标尺直线变为折线,其含义是最终定位还需考虑其他定位属性。Design 视图显示水平方向双重定位,如图 3-28 所示。运行程序时,“按

钮 1”也会显示在屏幕水平方向正中央。

图 3-27　约束布局运行结果

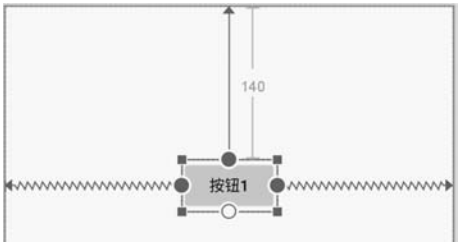

图 3-28　水平方向双重定位

此时“按钮 1”隐含以下属性(此时布局文件中不会显示此属性)：

```
app:layout_constraintHorizontal_bias = "0.5"
```

只要将“按钮 1”水平拖动，如图 3-29 所示，将多出属性 layout_constraintHorizontal_bias，其代表左右定位尺寸(左右圆形句柄到定位基线的距离，定位基线可能是屏幕边缘，也可能是其他控件边缘)的偏离率，0.5 代表按钮左右定位长度相等，小于 0.5 时按钮偏向左边，大于 0.5 时按钮偏向右边。可以在属性栏中直接修改偏离率，也可以在 Inspector(layout_constraintHorizontal_bias 属性，如图 3-29 标注 Constraint Widget 的图形部分)中直接拖动带数字的进度条修改偏离率，还可以通过直接拖动“按钮 1”的方式或在 Code 视图中修改偏离率。

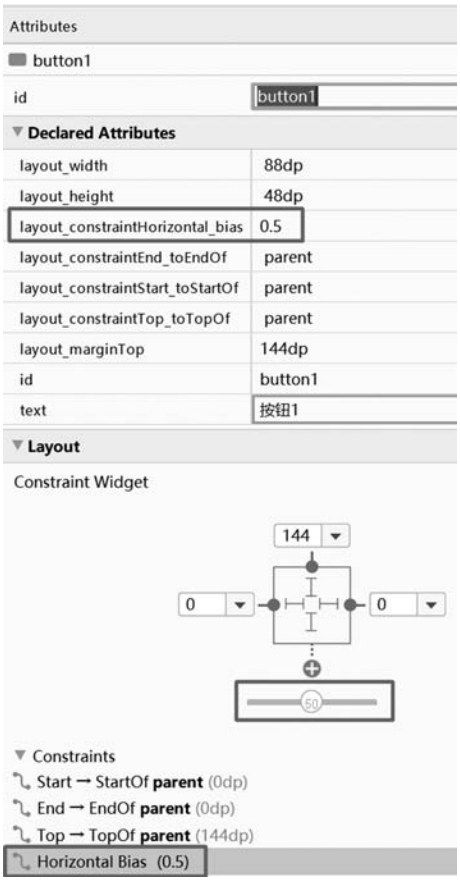

图 3-29　layout_constraintHorizontal_bias 属性

如果希望“按钮 1”无论怎么左右移动，按钮左侧到屏幕左边缘均至少保留 50dp 的间隙，可先选中“按钮 1”，然后在 Inspector 左侧文本框中输入 50，layout_marginStart 属性如图 3-30 所示。“按钮 1”左侧折线连接一段 50dp 的直线，相应的 layout_marginStart 属性为 50dp，此时偏离率是不计算这 50dp 直线长度的，即使 app:layout_constraintHorizontal_bias 属性值等于 0，“按钮 1”左侧还是会保留 50dp 的空白，此时拖动

“按钮 1”到 50dp 时就无法再向左边移动。

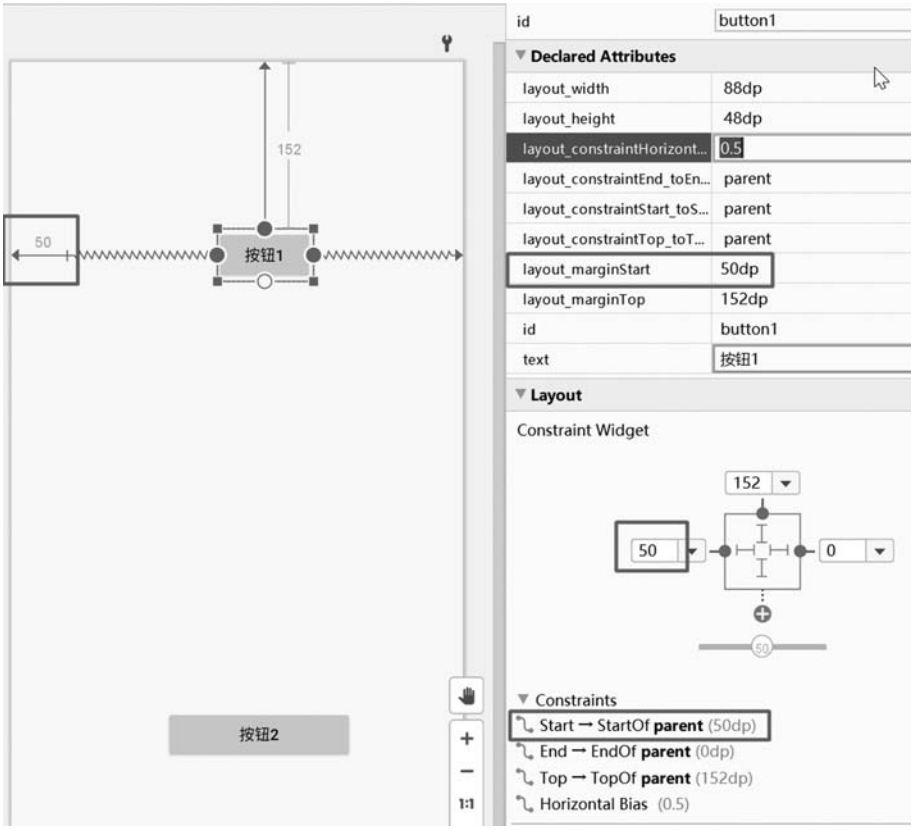

图 3-30　layout_marginStart 属性

选中控件的圆形句柄,按 delete 键可以删除选中的定位。如果选中的是控件,按 delete 键会删除整个控件(含控件定位属性)。

Inspector 如图 3-31 所示。

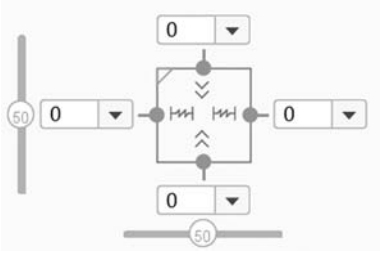

图 3-31　Inspector

>> 表示 wrap_content,Code 视图中的属性为 android:layout_width="wrap_content"。

|— 表示固定值,给控件指定了一个固定的长度或者宽度值。

|—| 表示任意长度或者宽度值。以控件宽度为例,Code 视图中控件的宽度属性变为 android:layout_width="0"。配合其他定位属性,控件宽度可能为 wrap_content(左右圆形句柄只有一个用于定位)、match_parent(左右圆形句柄都用于定位)或任意长度(左右圆形句柄都用于定位,且同时定义了 layout_marginStart 或 layout_marginEnd)。

在“按钮 1”上右击,弹出控件快捷菜单,如图 3-32 所示。

选择 Show Baseline 菜单,会在“按钮 1”上显示 Baseline(即基准线),如图 3-33 所示。单击“按钮 1”的基准线并拖放到“按钮 2”的基准线位置就可实现基准线对齐。基准线对齐主要用在多个高度不同的控件间实现文字对齐(如果按钮中文字显示为多行或两个按钮的

字体大小不同，基准线对齐是将第一行文字底部对齐）。

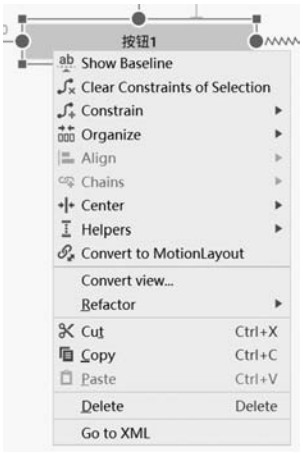

图 3-32　控件快捷菜单

图 3-33　Baseline

选择 Clear Constraints of Selection 菜单，将删除选中控件的所有约束布局属性。

选择 Convert view 菜单，弹出转换控件窗口，如图 3-34 所示。选择想变更的控件类型（Android 中称之为 View），可以在保留定位数据的情况下变更控件类型。

其他菜单项是常用选项，如复制、粘贴等，还有几个菜单项在设计图上方的工具栏中有相同功能按钮。相关功能可参看后续内容。

Transforms 如图 3-35 所示。在 Transforms 中可设置 View 的 X、Y、Z 轴的旋转和坐标参照点的偏移。

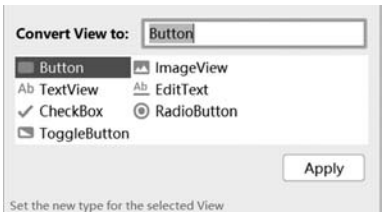

图 3-34　转换控件窗口

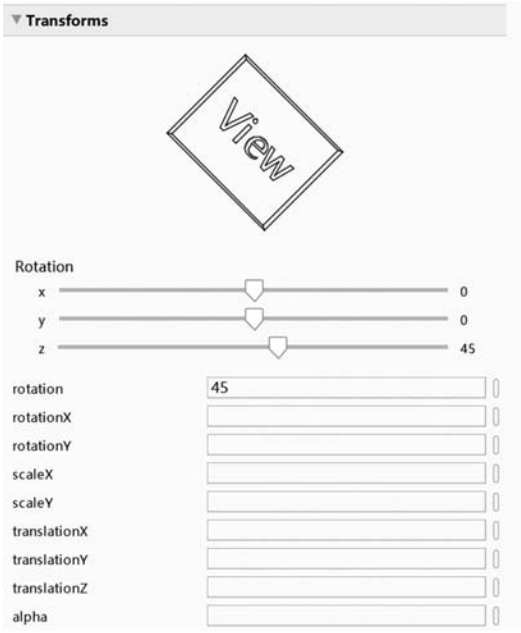

图 3-35　Transforms

图 3-35 中对 Z 轴旋转了 45°,对应属性 android:rotation="45"。如果定义在按钮控件内,则对应按钮控件旋转 45°。如果定义在 ConstraintLayout 标签内,则 ConstraintLayout 标签内的所有控件都会旋转 45°。

【main.xml】

```
<?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"
    android:rotation = " - 45">

    < Button
        android:id = "@ + id/button1"
        android:layout_width = "88dp"
        android:layout_height = "wrap_content"
        android:layout_marginStart = "196dp"
        android:layout_marginTop = "164dp"
        android:text = "随布局旋转按钮 1"
        app:layout_constraintStart_toStartOf = "parent"
        app:layout_constraintTop_toTopOf = "parent" />

    < Button
        android:id = "@ + id/button2"
        android:layout_width = "88dp"
        android:layout_height = "wrap_content"
        android:layout_marginStart = "108dp"
        android:layout_marginTop = "288dp"
        android:rotation = "90"
        android:text = "自定义旋转的按钮 2"
        app:layout_constraintStart_toStartOf = "parent"
        app:layout_constraintTop_toTopOf = "parent" />

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

第 7 行定义约束布局内所有控件都旋转－45°。此时 button1 和 button2 都旋转－45°。

第 25 行定义 button2 旋转 90°,扣除约束布局旋转的－45°,最终效果是 button2 旋转 45°。

android:rotation 属性运行结果如图 3-36 所示。其前缀是 android,属于 android 命名空间,所以也可以用在其他布局中,如线性布局。android:rotation 是以控件中心且垂直于屏幕为轴心的 Z 轴旋转。android:rotationX 和 android:rotationY 分别对应控件 X 方向中心轴和 Y 方向中心轴旋转。

视频讲解

3.10.2 Barrier

在实际布局中可能会遇到 Barrier 定位,如图 3-37 所示。希望“按钮 3”布置在“按钮 1”和“按钮 2”最右侧 40dp 的位置,即如果“按钮 2”比“按钮 1”更靠右,则“按钮 3”左侧距离“按钮 2”右侧 40dp;如果“按钮 1”比“按钮 2”更靠右,则“按钮 3”左侧距离“按钮 1”右侧 40dp。

图 3-36　android:rotation 属性运行结果

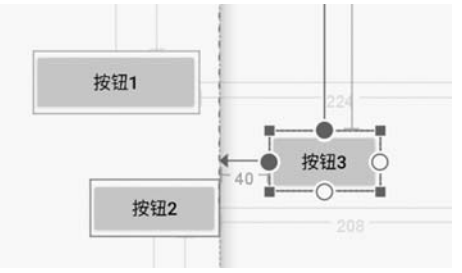

图 3-37　Barrier 定位

为此约束布局引入了 Barrier 的概念，增加了 Barrier 标签，其中定义以下两个属性：

```
app:barrierDirection = "right"
app:constraint_referenced_ids = "button1,button2"
```

以上两行是在“按钮 1”与“按钮 2”右侧建立 Barrier，Barrier 类似一堵墙，两个按钮谁更靠右，这堵墙就以谁为边界。而“按钮 3”左侧定位以这个墙为基准。使用图形化界面建立 Barrier 方法如下：建立“按钮 1”与“按钮 2”并完成相应定位。选中“按钮 1”，单击 按钮，弹出如图 3-38 所示的界面，选择 Add Vertical Barrier 菜单添加一个垂直方向 Barrier。

默认添加的 Barrier 与“按钮 1”左侧对齐。修改 Barrier 属性，如图 3-39 所示，在 barrier 属性栏中修改 barrierDirection 属性为 right(或者是 end)、constraint_referenced_ids 属性为“button1,button2”(默认只有 button1)。如果事先已经明确“按钮 1”和“按钮 2”共同建立 Barrier，也可同时选中“按钮 1”和“按钮 2”，然后再添加垂直方向的 Barrier，constraint_referenced_ids 属性自动填写为“button1,button2”。

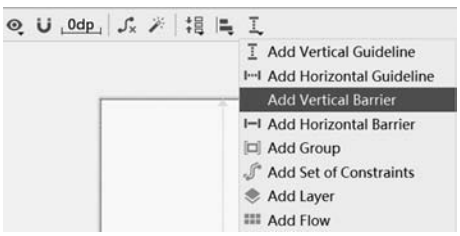

图 3-38　添加垂直方向 Barrier

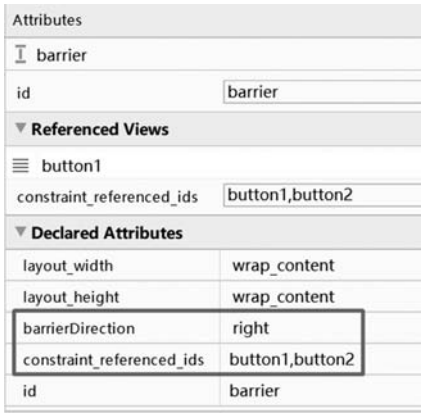

图 3-39　修改 barrier 属性

选中“按钮 3”,添加左侧定位到任意控件右边缘,使“按钮 3”的 Declared Attributes 属性栏中多出一项 layout_constraintStart_toEndOf 属性(也可以在 All Attributes 中查找对应属性),将其改为要对齐的 Barrier,将 layout_marginStart 改为 40dp。Android Studio 自动生成布局文件,代码如下:

【main.xml】

```
<?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">

    < Button
        android:id = "@ + id/button1"
        android:layout_width = "132dp"
        android:layout_height = "50dp"
        android:layout_marginTop = "176dp"
        android:layout_marginEnd = "224dp"
        android:text = "按钮 1"
        app:layout_constraintEnd_toEndOf = "parent"
        app:layout_constraintTop_toTopOf = "parent" />

    < Button
        android:id = "@ + id/button2"
        android:layout_width = "103dp"
        android:layout_height = "45dp"
        android:layout_marginEnd = "208dp"
        android:layout_marginBottom = "400dp"
        android:text = "按钮 2"
        app:layout_constraintBottom_toBottomOf = "parent"
        app:layout_constraintEnd_toEndOf = "parent" />

    < Button
        android:id = "@ + id/button3"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_marginStart = "40dp"
        android:layout_marginTop = "236dp"
        android:text = "按钮 3"
        app:layout_constraintStart_toEndOf = "@id/barrier1"
        app:layout_constraintTop_toTopOf = "parent" />

    < androidx.constraintlayout.widget.Barrier
```

```
39            android:id = "@ + id/barrier1"
40            android:layout_width = "wrap_content"
41            android:layout_height = "wrap_content"
42            app:barrierDirection = "right"
43            app:constraint_referenced_ids = "button1,button2"
44            tools:layout_editor_absoluteX = "55dp" />
45
46  </androidx.constraintlayout.widget.ConstraintLayout >
```

第 35 行定义 button3 左侧对齐到 barrier1 右侧。

第 38～44 行定义 barrier1，其中第 43 行定义 barrier1 阻挡的控件是 button1 和 button2，第 42 行指明 barrier1 阻挡方向是右侧。

3.10.3 Guideline


视频讲解

之前的控件定位都是基于屏幕（更准确的称呼为控件父容器的约束布局）或者是可见控件，约束布局中引入了一种在运行时看不见的 Guideline——定位基准线作为定位补充。添加 Guideline 如图 3-40 所示，分别添加垂直和水平方向的 Guideline。

拖动 Guideline 至所需位置，指定左定位 150dp 的垂直方向 Guideline，如图 3-41 所示。

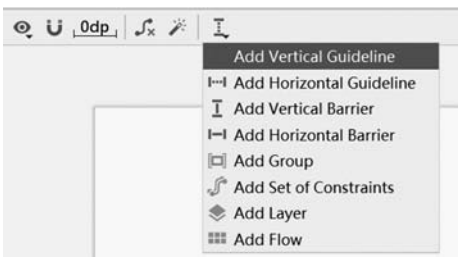

图 3-40　添加 Guideline

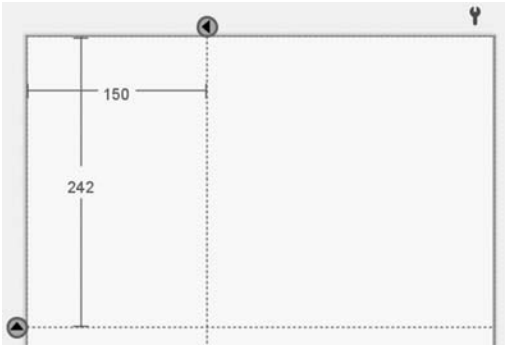

图 3-41　指定左定位 150dp 的垂直方向 Guideline

单击左侧向上箭头将切换为向下箭头，此时 Guideline 按 Bottom 位置定位，再次单击将切换成百分比符号，代表 Guideline 使用位置百分比设置自身定位（Guideline 属性 app:layout_constraintGuide_percent＝"0.33"是将 Guideline 设置在屏幕长度或宽度的 1/3 位置），指定百分比的水平方向 Guideline，如图 3-42 所示。

【注】 目前约束布局版本水平方向的 Guideline 可通过鼠标单击实现 、 、 三种定位方式循环切换，垂直方向的 Guideline 的切换还有问题。约束布局的功能在不停地升级，或许下一版本会将这个问题解决。

添加控件并将其定位指向 Guideline，使用 Guideline 定位如图 3-43 所示。实际运行时 Guideline 是不可见的。

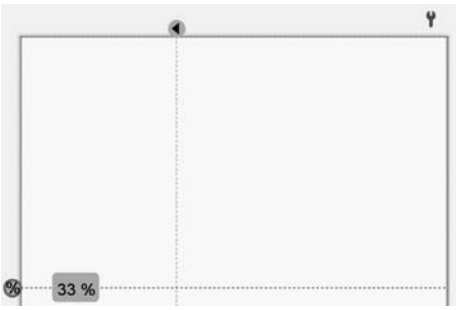

图 3-42　指定百分比的水平方向 Guideline

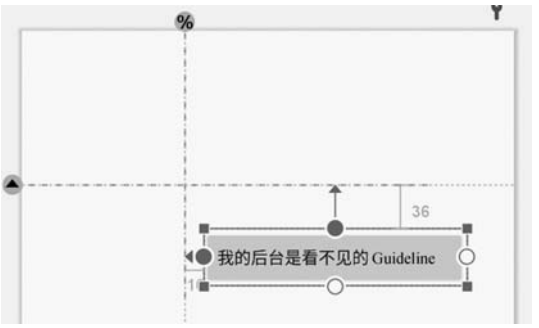

图 3-43　使用 Guideline 定位

布局代码如下：

【main.xml】

```
<?xml version = "1.0" encoding = "UTF - 8"?>
< androidx.constraintlayout.widget.ConstraintLayoutxmlns: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">

    < androidx.constraintlayout.widget.Guideline
        android:id = "@ + id/vertical_guide_line"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:orientation = "vertical"
        app:layout_constraintGuide_begin = "150dp" />

    < androidx.constraintlayout.widget.Guideline
        android:id = "@ + id/percent_guide_Line"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:orientation = "horizontal"
        app:layout_constraintGuide_percent = "0.33" />

    < Button
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_marginStart = "16dp"
        android:layout_marginTop = "36dp"
        android:text = "我的后台是看不见的 Guideline"
        app:layout_constraintStart_toStartOf = "@id/vertical_guide_line"
        app:layout_constraintTop_toTopOf = "@ + id/percent_guide_Line" />

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

第 8～13 行定义了一个距离屏幕左边界 150dp 的垂直方向 Guideline。

第 15～20 行定义了一个距离屏幕上端 1/3 位置的水平方向 Guideline。

第 22～29 行定义了一个按钮，其顶端和左侧分别定位到两个 Guideline。由于 Guideline 是一条线，因此第 28 行 layout_constraintStart_toStartOf 换成 layout_constraintStart_toEndOf 的效果是一样的。

3.10.4 Group

使用约束布局的一个目的是减少布局的嵌套。如果想把多个控件设置为隐藏，就需分别设置各控件的可视化属性。使用 Group 相当于将各控件进行分组，设置 Group 属性等效于将 Group 中各控件设置相同属性。在 Design 视图中添加 Group 时需先选择要包含的控件，然后如图 3-44 所示选择 Add Group 菜单项添加的 Group 对象，新添加 Group 对象取名为 group1。

group1 在程序运行时是不可见的，要将相应的控件放入 group1，最便捷的方式是在组件树中将相应的按钮控件拖放到 group1，如图 3-45 所示。

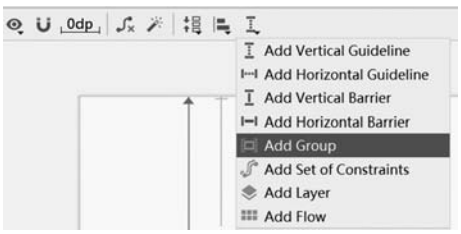

图 3-44　添加 Group

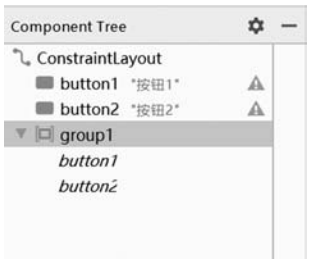

图 3-45　将按钮添加到 group1

此时 group1 中包含了 button1 和 button2。Group 的关键代码如下：

```
01      <androidx.constraintlayout.widget.Group
02          android:id="@+id/group1"
03          android:layout_width="wrap_content"
04          android:layout_height="wrap_content"
05          app:constraint_referenced_ids="button1,button2" />
```

其中，第 5 行定义 group1 中包含 button1 和 button2。运行程序，可以看到屏幕上显示 button1 和 button2。改变 group1 的 visibility 属性为 invisible，再次运行程序，屏幕上不显示 button1 和 button2。需要注意的是，group1 的属性栏中有两个 visibility 属性，如图 3-46 所示。

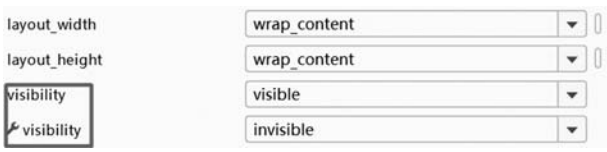

图 3-46　visibility 属性

上方的 visibility 属性在 Code 视图中显示为 android:visibility，其设定的值影响相关控件在 Android 设备上是否显示。下方的 visibility 属性前有一个 符号，在 Code 视图中显

示为 tools:visibility,其设定只影响在 Android Studio 的 Design 视图中是否显示,并不影响在 Android 设备上的运行显示。

3.10.5 Circle


视频讲解

Circle 方式定位目前还无法使用图形界面操作,可以在 Code 视图中输入代码。Circle 定位示意图如图 3-47 所示。以参考定位控件 A 中心为圆点,以控件 B 中心到控件 A 中心为半径,与垂直向上方向的夹角来定位控件 B 的位置。

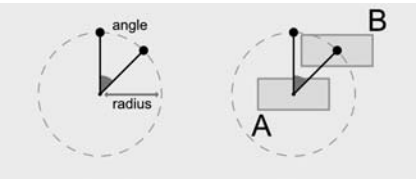

图 3-47 Circle 定位示意图

```
01  <?xml version = "1.0" encoding = "UTF - 8"?>
02  < androidx.constraintlayout.widget.ConstraintLayout xmlns:android = "http://schemas.
    android.com/apk/res/android"
03      xmlns:app = "http://schemas.android.com/apk/res - auto"
04      xmlns:tools = "http://schemas.android.com/tools"
05      android:layout_width = "match_parent"
06      android:layout_height = "match_parent">
07
08      < Button
09          android:id = "@ + id/button1"
10          android:layout_width = "132dp"
11          android:layout_height = "50dp"
12          android:layout_marginStart = "84dp"
13          android:layout_marginTop = "160dp"
14          android:text = "按钮 1"
15          app:layout_constraintStart_toStartOf = "parent"
16          app:layout_constraintTop_toTopOf = "parent" />
17
18      < Button
19          android:id = "@ + id/button2"
20          android:layout_width = "103dp"
21          android:layout_height = "45dp"
22          android:text = "按钮 2"
23          app:layout_constraintCircle = "@id/button1"
24          app:layout_constraintCircleAngle = "45"
25          app:layout_constraintCircleRadius = "150dp" />
26
27  </androidx.constraintlayout.widget.ConstraintLayout >
```

第 23 行定义 button2 按照 Circle 方式来定位,定位基准为 button1。

第 24 行定义夹角为 45°。

第 25 行定义 button1 与 button2 中心点的距离为 150dp。

目前,Circle 方定位方式还不完善,除了不支持图形化拖曳设计以外,在 Code 视图下 Button 标签也会显示为红色,组件树中 button2 也会用红色惊叹号标识,提示相关约束属性不完备。

3.10.6 Chain

Chain 用于指定链式约束。Chain 示意图如图 3-48 所示。图 3-48 中，控件 A 与控件 B 相互约束形成一个简单的链式约束。

链式约束中的第一个控件称为 Chain Head。在整个约束链中只要在 Chain Head 中定义链式约束类型即可。Chain Head 示意图如图 3-49 所示。

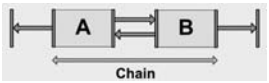

图 3-48 Chain 示意图

图 3-49 Chain Head 示意图

在 Head 控件中添加 layout_constraintHorizontal_chainStyle 属性，ChainStyle 属性及示意图如图 3-50 所示。5 种链式约束类型的主要区别是控件间的间隔或控件自身宽度比例。如果 Head 控件中未设置 layout_constraintHorizontal_chainStyle 属性，则默认为 Spread Chain 类型。

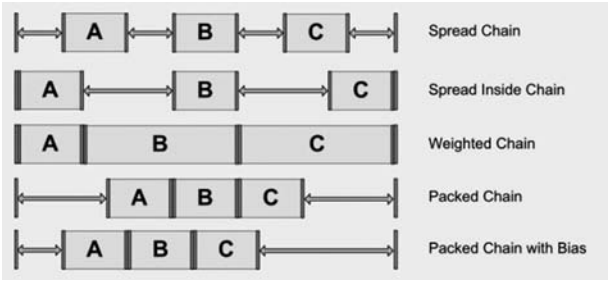

图 3-50 ChainStyle 属性及示意图

以下是根据官方代码编写的案例，分别实现图 3-50 的 5 种链式约束类型。

【main.xml】

```
<?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"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent">

    <!-- 默认为 app:layout_constraintVertical_chainStyle = "spread" -- >
    < Button
        android:id = "@ + id/spread1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "1"
        app:layout_constraintLeft_toLeftOf = "parent"
        app:layout_constraintRight_toLeftOf = "@ + id/spread2"
        app:layout_constraintTop_toTopOf = "parent" />

```

```
17      <Button
18          android:id="@+id/spread2"
19          android:layout_width="wrap_content"
20          android:layout_height="wrap_content"
21          android:text="2"
22          app:layout_constraintLeft_toRightOf="@+id/spread1"
23          app:layout_constraintRight_toLeftOf="@+id/spread3"
24          app:layout_constraintTop_toTopOf="@+id/spread1" />
25
26      <Button
27          android:id="@+id/spread3"
28          android:layout_width="wrap_content"
29          android:layout_height="wrap_content"
30          android:text="3"
31          app:layout_constraintLeft_toRightOf="@+id/spread2"
32          app:layout_constraintRight_toRightOf="parent"
33          app:layout_constraintTop_toTopOf="@+id/spread1" />
34
35      <!--app:layout_constraintHorizontal_chainStyle="spread_inside"-->
36      <Button
37          android:id="@+id/spread_inside1"
38          android:layout_width="wrap_content"
39          android:layout_height="wrap_content"
40          android:text="spread_in1"
41          app:layout_constraintHorizontal_chainStyle="spread_inside"
42          app:layout_constraintLeft_toLeftOf="parent"
43          app:layout_constraintRight_toLeftOf="@+id/spread_inside2"
44          app:layout_constraintTop_toBottomOf="@+id/spread1" />
45
46      <Button
47          android:id="@+id/spread_inside2"
48          android:layout_width="wrap_content"
49          android:layout_height="wrap_content"
50          android:text="spread_in2"
51          app:layout_constraintLeft_toRightOf="@+id/spread_inside1"
52          app:layout_constraintRight_toLeftOf="@+id/spread_inside3"
53          app:layout_constraintTop_toTopOf="@+id/spread_inside1" />
54
55      <Button
56          android:id="@+id/spread_inside3"
57          android:layout_width="wrap_content"
58          android:layout_height="wrap_content"
59          android:text="spread_in3"
60          app:layout_constraintLeft_toRightOf="@+id/spread_inside2"
61          app:layout_constraintRight_toRightOf="parent"
62          app:layout_constraintTop_toTopOf="@+id/spread_inside1" />
63
64      <!--app:layout_constraintHorizontal_weight 设置各控件比例-->
65      <Button
66          android:id="@+id/weight1"
67          android:layout_width="wrap_content"
68          android:layout_height="wrap_content"
69          android:text="weight1"
```

```
70          app:layout_constraintLeft_toLeftOf = "parent"
71          app:layout_constraintRight_toLeftOf = "@ + id/weight2"
72          app:layout_constraintTop_toBottomOf = "@ + id/spread_inside1" />
73
74      < Button
75          android:id = "@ + id/weight2"
76          android:layout_width = "0dp"
77          android:layout_height = "wrap_content"
78          android:text = "weight2"
79          app:layout_constraintHorizontal_weight = "2"
80          app:layout_constraintLeft_toRightOf = "@ + id/weight1"
81          app:layout_constraintRight_toLeftOf = "@ + id/weight3"
82          app:layout_constraintTop_toTopOf = "@ + id/weight1" />
83
84      < Button
85          android:id = "@ + id/weight3"
86          android:layout_width = "0dp"
87          android:layout_height = "wrap_content"
88          android:text = "weight3"
89          app:layout_constraintHorizontal_weight = "3"
90          app:layout_constraintLeft_toRightOf = "@ + id/weight2"
91          app:layout_constraintRight_toRightOf = "parent"
92          app:layout_constraintTop_toTopOf = "@ + id/weight1" />
93
94
95      <!-- app:layout_constraintHorizontal_chainStyle = "packed" -- >
96      < Button
97          android:id = "@ + id/packed1"
98          android:layout_width = "wrap_content"
99          android:layout_height = "wrap_content"
100          android:text = "packed1"
101          app:layout_constraintHorizontal_chainStyle = "packed"
102          app:layout_constraintLeft_toLeftOf = "parent"
103          app:layout_constraintRight_toLeftOf = "@ + id/packed2"
104          app:layout_constraintTop_toBottomOf = "@ + id/weight1" />
105
106      < Button
107          android:id = "@ + id/packed2"
108          android:layout_width = "wrap_content"
109          android:layout_height = "wrap_content"
110          android:text = "packed2"
111          app:layout_constraintLeft_toRightOf = "@ + id/packed1"
112          app:layout_constraintRight_toLeftOf = "@ + id/packed3"
113          app:layout_constraintTop_toTopOf = "@ + id/packed1" />
114
115      < Button
116          android:id = "@ + id/packed3"
117          android:layout_width = "wrap_content"
118          android:layout_height = "wrap_content"
119          android:text = "packed3"
120          app:layout_constraintLeft_toRightOf = "@ + id/packed2"
121          app:layout_constraintRight_toRightOf = "parent"
122          app:layout_constraintTop_toTopOf = "@ + id/packed1" />
```

```
123
124       <!-- app:layout_constraintHorizontal_bias = "0.2" -->
125       <!-- app:layout_constraintHorizontal_chainStyle = "packed" -->
126       < Button
127            android:id = "@ + id/bias1"
128            android:layout_width = "wrap_content"
129            android:layout_height = "wrap_content"
130            android:text = "bias1"
131            app:layout_constraintHorizontal_bias = "0.2"
132            app:layout_constraintHorizontal_chainStyle = "packed"
133            app:layout_constraintLeft_toLeftOf = "parent"
134            app:layout_constraintRight_toLeftOf = "@ + id/bias2"
135            app:layout_constraintTop_toBottomOf = "@ + id/packed1" />
136
137       < Button
138            android:id = "@ + id/bias2"
139            android:layout_width = "wrap_content"
140            android:layout_height = "wrap_content"
141            android:text = "bias2"
142            app:layout_constraintLeft_toRightOf = "@ + id/bias1"
143            app:layout_constraintRight_toLeftOf = "@ + id/bias3"
144            app:layout_constraintTop_toTopOf = "@ + id/bias1" />
145
146       < Button
147            android:id = "@ + id/bias3"
148            android:layout_width = "wrap_content"
149            android:layout_height = "wrap_content"
150            android:text = "bias3"
151            app:layout_constraintLeft_toRightOf = "@ + id/bias2"
152            app:layout_constraintRight_toRightOf = "parent"
153            app:layout_constraintTop_toTopOf = "@ + id/bias1" />
154
155  </androidx.constraintlayout.widget.ConstraintLayout >
```

5 种链式约束类型运行结果如图 3-51 所示。

图 3-51　5 种链式约束类型运行结果