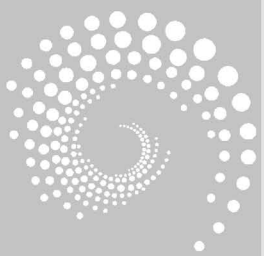
第3章

Android应用程序页面的组织者

CHAPTER 3



学习目标：

熟悉使用线性布局，实现相应的页面效果。

熟悉使用相对布局，实现相应的页面效果。

熟悉使用表格布局，实现相应的页面效果。

熟悉使用网格布局，实现相应的页面效果。

了解帧布局、绝对布局，在适当的时候能够选择相应的布局方式实现相应的页面效果。

职业技能目标：

能够根据页面效果图，选择恰当的布局实现相应的效果。

课程思政育人目标：

培养学生的大局意识和工匠精神，锻炼学生的毅力和组织能力，唤醒学生的自重能力。

学习导读：

为了更好地掌握本章的内容，请读者按照本章导读进行学习。

首先，在进行课堂学习之前，请先完成课前学习任务，熟悉Android应用程序中常用的布局控件，并学会根据界面设计图选择恰当的布局实现相应的页面效果。

其次，在课堂上通过完成课堂任务，深入学习Android项目中各个布局类的使用技巧，使学生能够灵活使用Android项目中的布局来实现相应的页面效果。

最后，在课后独立设计一个页面，并使用已学习的知识，实现页面的效果。

3.1　课前学习任务：掌握常用布局的定义及其常用属性的使用

设计实现 Android 布局是应用界面开发的重要环节，布局是 Android 应用程序的界面框架。布局中所有界面元素都是视图(View)对象或视图组(ViewGroup)对象，一个布局首先是一个视图组对象，然后在视图组对象中添加子视图组对象或者视图对象。简而言之，布局就是把界面的控件按照某种规律摆放在指定的位置，为了解决应用程序在不同手机中的显示问题。Android 中实现布局有两种方式，一种是通过 Java 源代码，另一种是 XML 配置文件，在本章将重点学习用 XML 配置文件实现布局。在学习的过程中，请思考以下问题。

(1) 线性布局在布局文件中定义的标签名称是什么？常用属性有哪些？
(2) 相对布局在布局文件中定义的标签名称是什么？常用属性有哪些？
(3) 表格布局在布局文件中定义的标签名称是什么？常用属性有哪些？
(4) 网格布局在布局文件中定义的标签名称是什么？常用属性有哪些？
(5) 帧布局在布局文件中定义的标签名称是什么？常用属性有哪些？
(6) 绝对布局在布局文件中定义的标签名称是什么？常用属性有哪些？

如果学习完本章内容还不能解决上述问题，请重新学习相关章节，或查阅相关资源。

3.1.1　布局的介绍

在 Android 应用程序中，所有的界面元素都是由 View 和 ViewGroup 的对象构成的。View 是绘制在屏幕上能与用户交互的一个对象，而 ViewGroup 则是一个用于存放其他 View 和 ViewGroup 对象的布局容器。Android 还提供了一个 View 和 ViewGroup 子类的集合，集合中提供了一些常用的控件和各种各类的布局模式，布局的实现是本章的重点，而控件的使用则是第 4 章的重点。View 和 ViewGroup 的关系与玻璃和窗框的关系相似，窗框用于控制玻璃的摆放位置，ViewGroup 用于控制 View 的位置。ViewGroup 继承 View 类，是 View 类的扩展，是用来容纳其他控件的容器，由于 ViewGroup 是个抽象类，所以一般使用 ViewGroup 的子类来作为容器。Android 视图元素相关类之间的关系层次结构如图 3-1 所示，从图中可以看出，Android 视图元素类的顶层是一个 ViewGroup 对象，在该对象下面有子类，子类可以是 View 对象和 ViewGroup 对象。

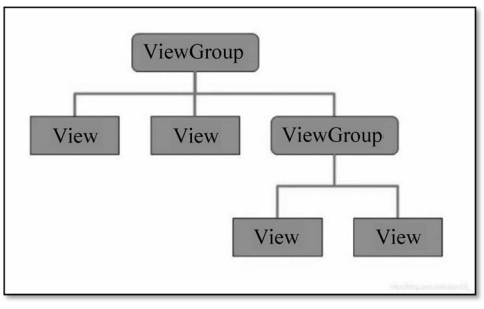


图 3-1　Android 视图元素顶层类结构

通过 ViewGroup 类直接派生出来的布局类有绝对布局(AbsoluteLayout)、层布局(FrameLayout)、线性布局(LinearLayout)、网格布局(GridLayout)、相对布局(RelativeLayout)和表格布局(TableLayout),表格布局是线性布局的子类,其类图如图 3-2 所示。GridLayout 是 Android 4.0 引入的一种新的布局,使用的方法与 TableLayout 相似,但加入了更多好用的属性,后面会详细讲解各种布局的使用以及它们的重点属性。

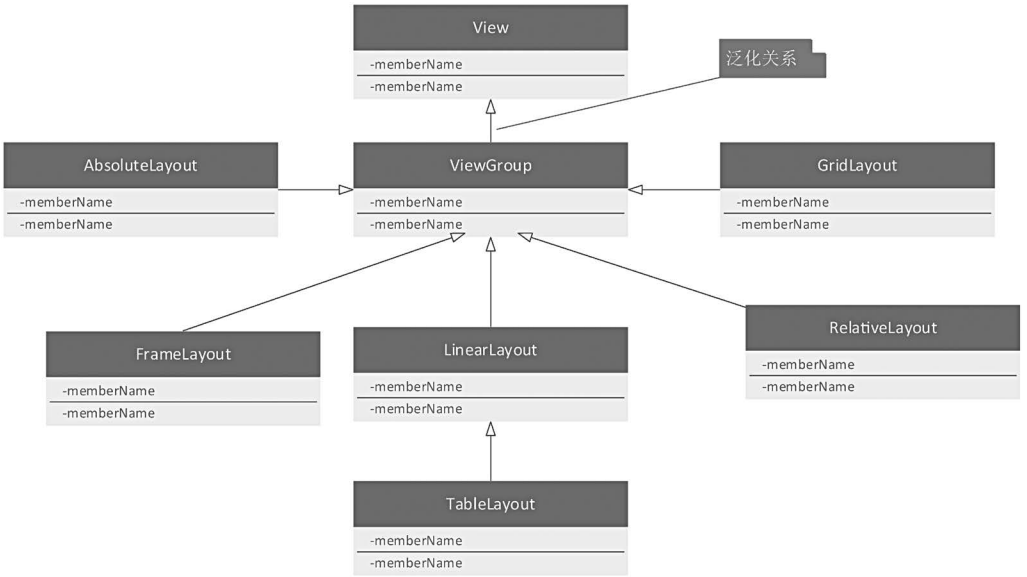


图 3-2 Android 布局类之间的类图

1. 使用代码创建布局

Android 应用程序一般推荐使用 XML 来设置页面框架,但也可以使用 Java 源代码来实现整个页面的框架,使用 Java 源代码创建页面框架的步骤如下。

(1) 使用恰当的布局类创建布局对象,代码如下。

```
//上下文一般布局所在的 Activity 的 this 对象
布局类 对象名 = new 布局类(上下文);
```

如在 Activity 类中创建一个线性布局对象,其代码格式如下。

```
LinearLayout linearLayout = new LinearLayout(this);
```

(2) 根据页面设计图选择合适的布局,并设置相应的布局属性,具体设置哪些布局属性,可以根据页面效果需要来设置。例如页面中的组件是以列的形式进行排列的,则需要设置线性布局的线性方向为垂直,代码如下。

```
linearLayout.setOrientation(LinearLayout.VERTICAL);
```

(3) 根据页面效果创建相应的控件对象,代码格式如下。

```
控件类 控件对象名 = new 控件类(this);
```

(4) 设置控件的宽高参数,定义参数对象的代码格式如下。

```
LayoutParams 宽高参数对象 = new LayoutParams(宽,高);
```

(5) 根据页面效果设置控件的属性,例如设置控件中内容位置的属性,代码如下。

```
//设置控件在布局里居中对齐
宽高参数对象.gravity = Gravity.CENTER;
```

(6) 在布局中添加控件,代码如下。

```
布局对象.addView(控件对象,宽高参数对线下);
```

(7) 将整个布局的内容显示在手机屏幕中,代码如下。

```
setContentView(布局对象);
```

上述代码量不多,难点是如何选择布局,确定相应的布局类,选择相应的控件,实现界面的效果和功能,下面演示使用纯 Java 代码实现一个检索界面的效果。

步骤 1,打开 Android Studio 软件,然后导入上一次课的 Android 项目或者新建一个 Android 项目,将项目结构栏的视图模式切换为 Android 模式。

步骤 2,打开项目结构中的 java 目录,然后选中项目主包名,右击→New→Java Class,具体操作如图 3-3 所示。

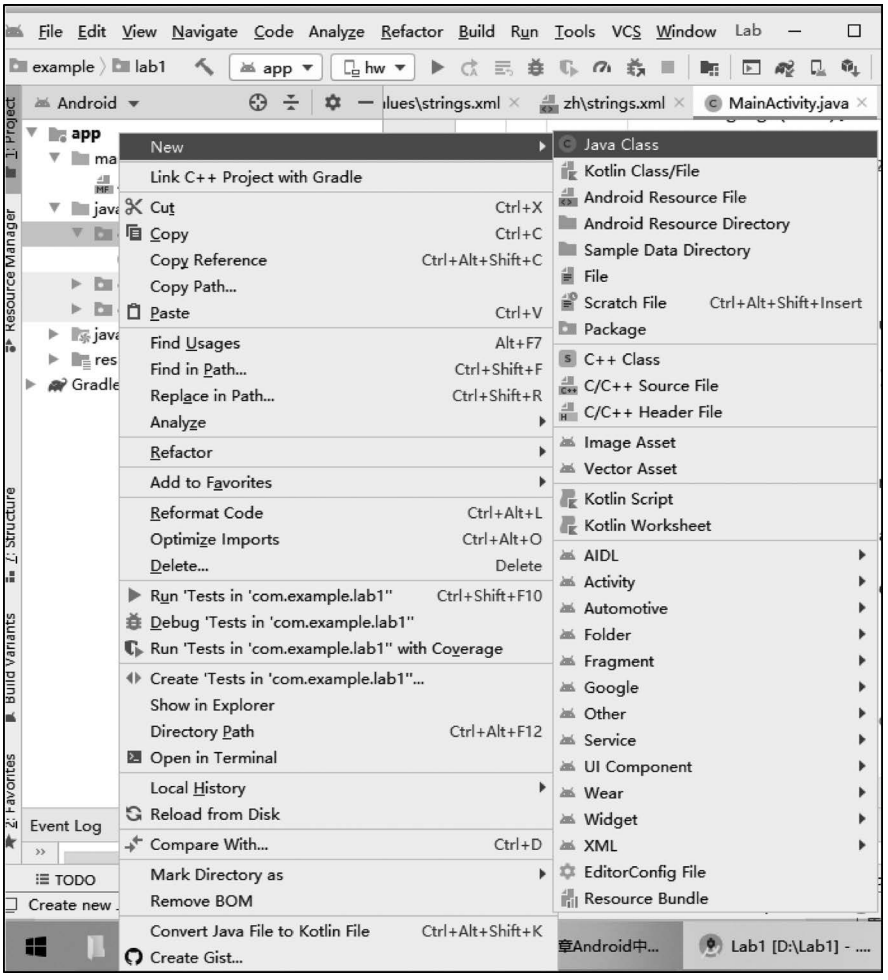


图 3-3　新建 Java Class 类的过程

步骤 3，在弹窗中输入 Java Class 的名称，类名一般为页面的名称，因为要实现一个检索页面的效果，所以类名为 SearchActivity，具体操作如图 3-4 所示。填写好类名后，选中 Class 选项双击或者按下 Enter 键，就会打开新建的类。

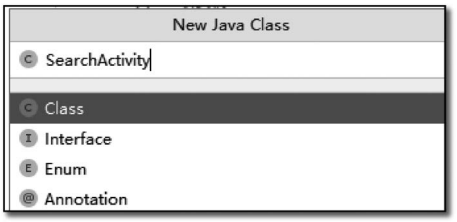


图 3-4　填写类名窗口

步骤 4，让新建的类继承 Activity 类，并重写 onCreate()方法，代码如图 3-5 所示。

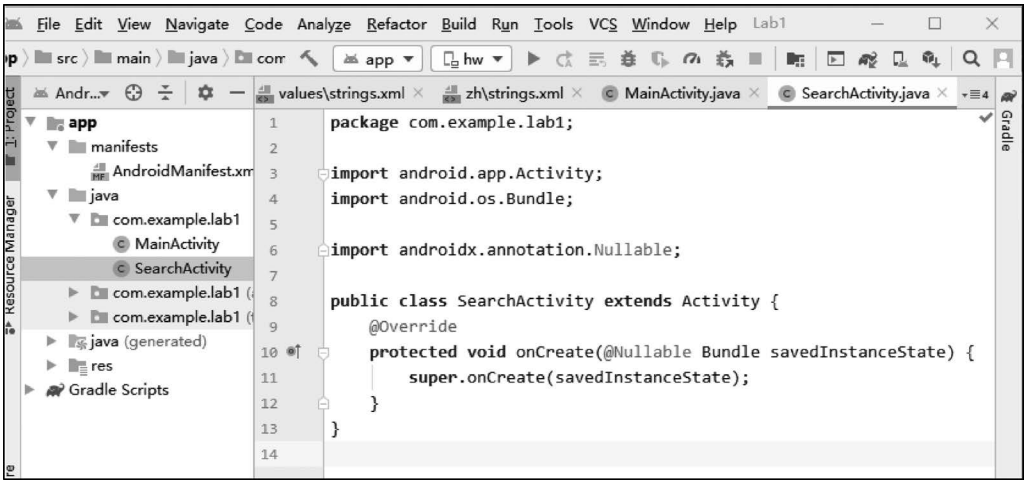


图 3-5　SearchActivity 类的代码

步骤 5，按照使用 Java 代码实现布局的步骤，在 SearchActivity 中编写 Java 代码实现线性布局的界面效果，并将它封装成一个方法，其代码如图 3-6 所示。

步骤 6，在 onCreate()方法中调用步骤 5 中编写的方法，其代码如图 3-7 所示。

步骤 7，打开清单文件 AndroidManifest. xml，将 SearchActivity 修改为应用程序的启动页面，其代码如图 3-8 所示。

步骤 8，运行测试应用程序，其效果如图 3-9 所示。

其他布局也可以使用 Java 代码去实现相应的界面效果，代码类似，就不重复讲述，读者可以自行使用其他布局的 Java 代码实现检索页面的效果。

2. 使用 XML 实现布局

上一节内容讲解了使用线性布局的 Java 源代码来实现一个检索页面，页面中只有两个控件，却需要接近 50 行代码才能实现，由此可见单纯使用 Java 源代码实现界面效果是非常麻烦的。Android 项目的结构是典型的 MVC 结构，在 res 里有专门的 layout 目录用于存放应用程序静态的页面效果或者框架。可以通过以下步骤使用 XML 文件实现界面效果。

(1) 根据界面效果分析界面的布局及其包含的控件。

```
18      //利用代码设置线性布局
19      private void setLinearLayout(){
20          //步骤1：创建线性布局对象，并设置布局的属性
21          LinearLayout layout1 = new LinearLayout( context: this);//创建页面的总布局模式
22          //步骤2：设置布局的属性
23          layout1.setOrientation(LinearLayout.VERTICAL);//设置线性布局的方向为垂直布局
24          //步骤3：创建控件对象
25          LinearLayout layout2 = new LinearLayout( context: this);
26          layout2.setOrientation(LinearLayout.HORIZONTAL);
27          layout2.setWeightSum(5);//将容器一行分为5份
28          //步骤3：创建控件对象
29          EditText editText = new EditText( context: this);//创建一个输入框
30          //步骤4：设置控件的宽高参数，
31          LinearLayout.LayoutParams params1 = new LinearLayout.LayoutParams(
32                  width: 0, ViewGroup.LayoutParams.WRAP_CONTENT);
33          params1.weight = 4;//设置输入框占一行的4份
34          //步骤5：根据功能设置控件的属性
35          editText.setHint("请输入检索关键字");//输入输入框中的输入提示
36          //步骤6：将控件添加到线性布局中
37          layout2.addView(editText,params1);
38          //步骤3：创建控件对象
39          Button btn = new Button( context: this);
40          btn.setText("搜索");
41          LinearLayout.LayoutParams params2 = new LinearLayout.LayoutParams(
42                  width: 0, ViewGroup.LayoutParams.WRAP_CONTENT);
43          params2.weight = 1;//设置按钮占一行的1份
44          layout2.addView(btn,params2);
45          layout1.addView(layout2);
46          setContentView(layout1);
47      }
48  }
```

图 3-6　通过线性布局实现检索页面的 Java 源代码

```
13      @Override
14      protected void onCreate(@Nullable Bundle savedInstanceState) {
15          super.onCreate(savedInstanceState);
16          setLinearLayout();
17      }
```

图 3-7　在 onCreate()方法中调用 setLinearLayout()方法

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3       package="com.example.lab1">
4       <application
5           android:allowBackup="true"
6           android:icon="@mipmap/main_index_my_pressed"
7           android:roundIcon="@mipmap/main_index_my_pressed"
8           android:label="Lab1"
9           android:configChanges="locale"
10          android:supportsRtl="true"
11          android:theme="@style/AppTheme">
12          <activity android:name=".SearchActivity">
13              <intent-filter>
14                  <action android:name="android.intent.action.MAIN" />
15                  <category android:name="android.intent.category.LAUNCHER"/>
16              </intent-filter>
17          </activity>
18      </application>
19  </manifest>
```

页面的逻辑控制类

启动页面代码

图 3-8　在清单文件中将 SearchActivity 设置为应用程序的启动页面

(2) 新建一个布局文件。

(3) 修改布局文件的根标签即页面的布局。

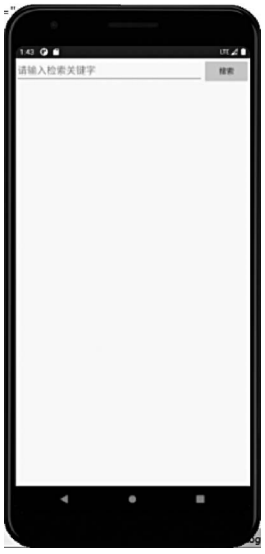


图 3-9 检索页面效果

（4）根据第一步的分析，在布局中自上而下地添加控件。

（5）在该布局的页面逻辑控制类 XXXActivity 的 onCreate() 方法中，使用 setContentView() 方法，将布局和页面逻辑控制类绑定在一起，就可以将布局文件里的内容在手机屏幕中显示出来。

接下来讲解使用 XML 实现图 3-9 所示的检索页面效果的步骤。

（1）根据页面的效果可知，该界面有两个控件，一个是输入框（EditText），一个是按钮（Button），它们是水平排列，因此布局文件里根标签可以使用线性布局，并设置线性布局的方向为水平方向（horizontal）。

（2）新建检索页面的布局文件，创建布局文件步骤是先选中项目中 res/layout 目录，接着右键选择 Layout Resource File，然后在弹出的窗口中填写布局文件名称等信息，弹窗效果如图 3-10 所示，填写好文件名后单击 OK 按钮，会在 Android Studio 的编辑区打开新建的布局文件，效果如图 3-11 所示。

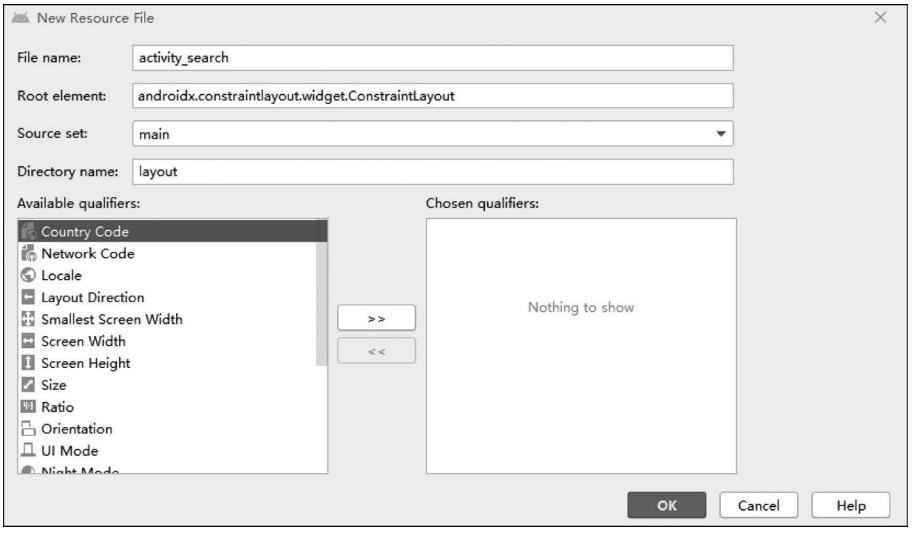


图 3-10 填写布局文件信息的弹窗

图 3-11 新建的布局文件的代码

(3) 将图 3-11 中的根标签修改为线性布局,在布局文件中使用标签定义布局,线性布局的标签为< LinearLayout >,需要将图 3-11 中的第一个红框中的根标签代码改为 LinearLayout,记得保留左边的尖括号(<),并在 LinearLayout 标签中添加 orientation 属性,属性值设置为 horizontal,修改后布局文件中的代码如图 3-12 所示。

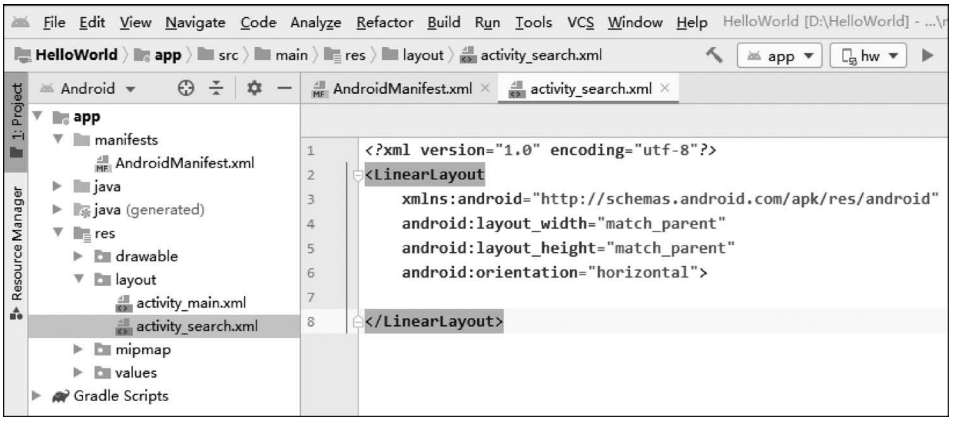


图 3-12　修改根标签后的布局文件代码

(4) 在根标签中添加控件,添加控件后布局文件中的代码和预览效果如图 3-13 所示。

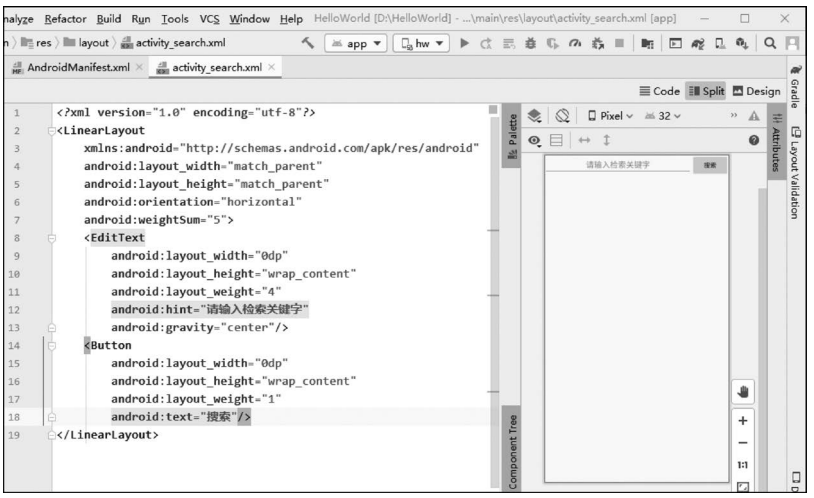


图 3-13　添加控件后布局文件中的代码及预览效果

(5) 为 activity_search. xml 布局文件创建一个页面逻辑控制类 SearchActivity,并重写其 onCreate( ) 方法,在 onCreate( ) 方法中使用 setContentView( ) 方法将布局文件(R. layout. 布局文件名)和页面逻辑控制类绑定在一起,创建页面逻辑控制类的步骤是先选中 java/应用程序主包名,然后右键选择 New,接着选择 Java Class,在弹窗中填写类名为 SearchActivity,类名可以按照 Java 代码的编程风格自己进行命名,填写完后选择 Class,双击或者按下 Enter 键,就会在编辑区打开新建的页面逻辑控制类,其代码如图 3-14 所示。接着让新建的 SearchActivity 继承 Activity 类,并重写 onCreate()方法,并在该方法中将布局文件与页面逻辑控制类进行绑定,其代码如 3-15 所示。

(6) 在清单文件 AndroidManifest. xml 中将页面逻辑控制类设置为启动页面,即将清

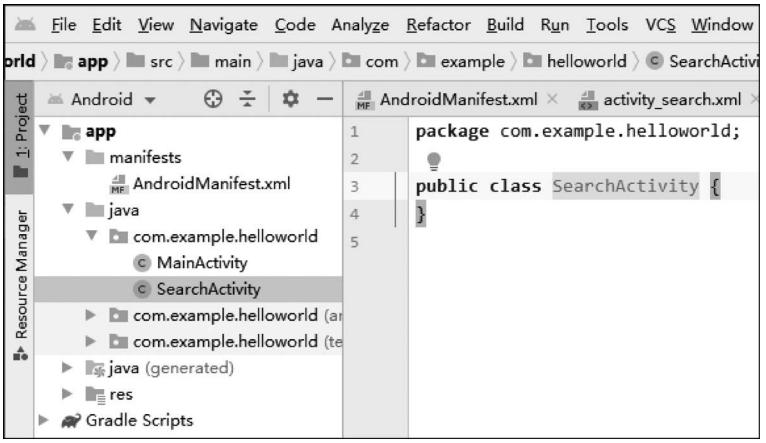


图 3-14 为 activity_search. xml 布局文件新建的页面逻辑控制类

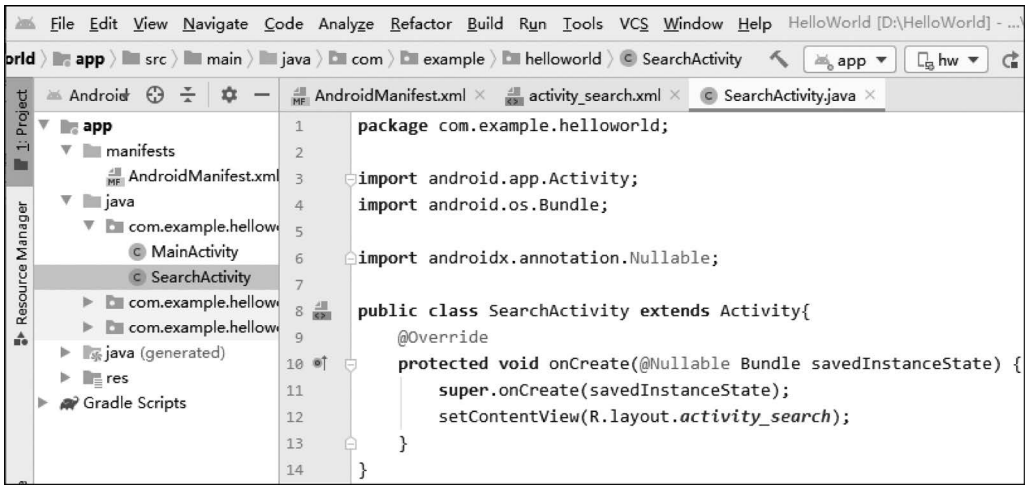


图 3-15 页面逻辑控制类中的代码

单文件中 activity 标签的 name 属性值. MainActiviy 改为. SearchActivity，其代码如图 3-16 所示，修改完即可按“运行”按钮，将应用程序安装和运行在手机模拟器中，其运行效果如图 3-17 所示。

通过使用源代码和布局文件实现相同界面效果，会发现使用布局文件实现界面应用程序的结构更加清晰，在布局文件中实现基本的页面框架，使用页面逻辑控制类实现页面的逻辑，这种页面和功能相分离的结构会更加有利于后期团队合作开发、代码修改和维护。

3. 在布局中添加控件的步骤

在布局中添加控件是 Android 应用程序开发成功的关键步骤，在多年的教学中，发现很多学生在开发 Android 应用程序的界面时都无法在布局中正确地定义控件，经过总结和整理，得到一些在布局中添加控件的技巧。

首先，要掌握在布局中定义控件的格式。布局里的控件可以分为容器控件和内容控件，内容控件是以单标签的形式进行定义，以下是其定义格式。

AndroidManifest.xml　activity_search.xml　SearchActivity.java　Gradle

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.helloworld">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/main_index_my_pressed"
        android:label="HelloWorld"
        android:roundIcon="@mipmap/main_index_my_pressed"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".SearchActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

图 3-16　清单文件中的代码

```
<控件名
    属性/>
```

例如：定义一个文本框控件，以下是其参考代码。

```
< TextView
    属性.../>
```

另一种是容器控件，是以双标签形式定义，在开始标签中定义属性，在标签里可以定义子标签，最后必须有一个结束标签，布局控件或容器控件一般用成对标签进行定义，以下是其定义格式。

```
<控件名
  属性>
  <子控件
    属性/>
</控件名>
```

例如：定义一个线性布局，并在线性布局中添加控件，以下是其参考代码。

```
< LinearLayout
    属性...>
    < TextView
        属性.../>
</ LinearLayout >
```

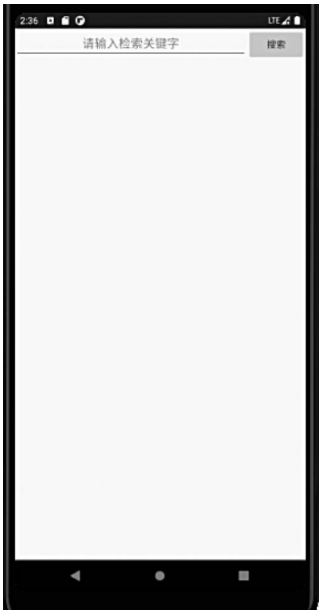


图 3-17　应用程序运行到手机模拟器的效果

以上两种定义控件的格式，读者一定要熟记掌握，并能够灵活应用，才能实现 Android 应用程序的界面效果。

接着需要学会在控件中添加恰当的属性实现界面的效果，每个控件都拥有很多属性，在定义控件时要根据需求在布局中定义相应的属性，控件的属性可以分为必要属性和可选属性，必要属性是定义控件时必须在布局中显式定义出来的，可选属性是根据界面效果的需求选择性使用。

1）必要属性

在布局中定义控件时必须声明控件的宽高，否则系统会出现错误，导致应用程序不能运行，以下是控件宽高属性的定义格式。

宽：layout_width = "值"

高：layout_height = "值"

设置控件的宽高值有三种方法，具体设置要求如下。

（1）宽高的值可以使用系统自带的三个值，以下是这三个值的具体含义。

match_parent 表示填充父容器，wrap_content 可以根据内容自适应组件大小，fill_parent 表示填充父容器，但是从 API 级别 8 开始被弃用并被替换为 match_parent。

（2）直接设置控件宽高，格式为数字＋单位，例如设置宽度为 50dp，其代码为 layout_width="50dp"。

（3）使用 weight 权重值来设置控件的宽高，其步骤如下。

步骤 1，在父容器中使用 android:weightSum 属性将容器的宽或高分成若干份，该属性值可以是小数也可以是整数。

步骤 2，将控件的宽或高属性值设为 0dp。

步骤 3，在控件中添加 android:layout_weight 属性，并设置该属性值，该值要小于 weightSum 的值。

2）可选属性

控件中除了宽高属性外其他的属性为可选属性，可选属性是根据界面效果的需求来进行选择设置，这些属性根据其内容性质分为内容属性、内容美化属性、位置属性和标识属性，一般在编写控件属性时，先编写控件的内容属性，再编写美化属性，再编写控件的位置属性，最后编写控件的标识属性。下面介绍各类属性的情况。

（1）内容属性决定着控件中显示的内容，控件的内容一般包含文字、颜色和图片，这些内容包含的属性有

设置文字属性：android:text、android:label 等。

设置图片属性：android:background、android:icon、android:src 等。

（2）修饰属性，一般用于美化控件或者控件里的内容，使控件及其组成的界面更加美观，添加控件内容美化属性需要根据页面效果图的需要来确定，不同控件会有不同的属性，例如文本框控件 TextView 常用的内容美化属性有 android:textSize、android:textColor 等，不同的控件有不同的美化属性，通用的美化属性有 android:background 设置控件背景，android:style 设置控件的样式等。

（3）位置属性，一般用于设置控件在父容器或者在手机屏幕上的位置，设置位置属性时一般要根据父容器的布局方式来设置，通用的位置属性有 android:layout_margin、android:layout_marginLeft、android:layout_marginRight、android:layout_marginTop、android:layout_marginBottom 这五个属性用于设置控件间的间隙，而 android:padding、android:paddingLeft、android:paddingRight、android:paddingTop、android:paddingBottom 这五个属性用于设置控件内容到控件边缘的空隙。

（4）标识属性 android:id 用于标记控件在项目中的名字，方便后续使用源代码控制控件。初学者会产生疑问，是否需要给控件设置标识属性，先判断该控件的内容是否需要被修

改。如果需要被改变，则需要给控件设置标识属性。

例如需要在界面上添加一个显示公司名的文本框控件，就可以在布局中添加文本框控件的代码，以下是其参考代码。

```
<TextView
    android:id = "@ + id/tv_pagename_company"
    android:layout_width = "wrap_content"
    android:layout_height = "match_parent"
    android:text = "公司名"
    android:textSize = "20sp"
    android:textColor = "#000"/>
```

由上述代码可见，定义控件时需要加入控件宽高属性、内容属性、内容修饰属性、控件修饰属性、标识属性，标识属性值的命名规范一般为控件类名缩写_页面名_控件的显示内容名称。需要注意的是，在写标识属性值前需要将标识属性的前缀@＋id 要写出来，标识属性值的命名格式为：@＋id/标识属性名。

了解完布局的基本情况和在布局中添加控件的步骤后，接下来逐个讲解各个布局的常用属性和使用技巧。

3.1.2　线性布局

线性布局主要是以水平或者垂直的方式来组织界面中的控件，由 orientation 来指定，若 orientation 属性值为 vertical，表示垂直方式来设置控件，控件的显示顺序是从上到下；若 orientation 属性值为 horizontal，表示水平方式来设置控件，控件显示的顺序是从左到右。线性布局中常用属性如表 3-1 所示。

表 3-1　LinearLayout 常用属性

属　性　名	属 性 说 明
orientation	设置布局中的控件的排列方向，有两个可选值：vertical（垂直）、horizontal（水平）
android:gravity	控制控件所包含的子元素或者内容的对齐方式，可多个组合，如(left|buttom)
android:layout_gravity	控制该控件在父容器里的对齐方式
android:divider	为 LinearLayout 设置分割线的图片
android:showDividers	设置分割线所在的位置，有四个可选值：none，middle，beginning，end
android:dividerPadding	设置分割线的空隙

了解完线性布局的属性后，接下来尝试一下在布局文件中使用线性布局的实现效果和过程。

首先，打开上一节课的项目或新建一个项目，在项目的 layout 目录中新建一个布局文件，布局文件名为 activity_lab1，并将布局文件中的根标签修改为 LinearLayout，在布局中添加 orientation 属性，并将属性值修改为 vertical。

其次，在布局中添加两个文本框控件，文本框分别显示文本框 1、文本框 2，文字大小设置为 40sp，其参考代码及预览效果如图 3-18 所示。

最后，在该布局中加入一个线性布局，将线性布局的 orientation 属性值设置为 horizontal，并在该标签里继续添加两个文本控件，分别显示文本框 3、文本框 4，文字大小为 40sp，其参考代码及预览效果如 3-19 所示。

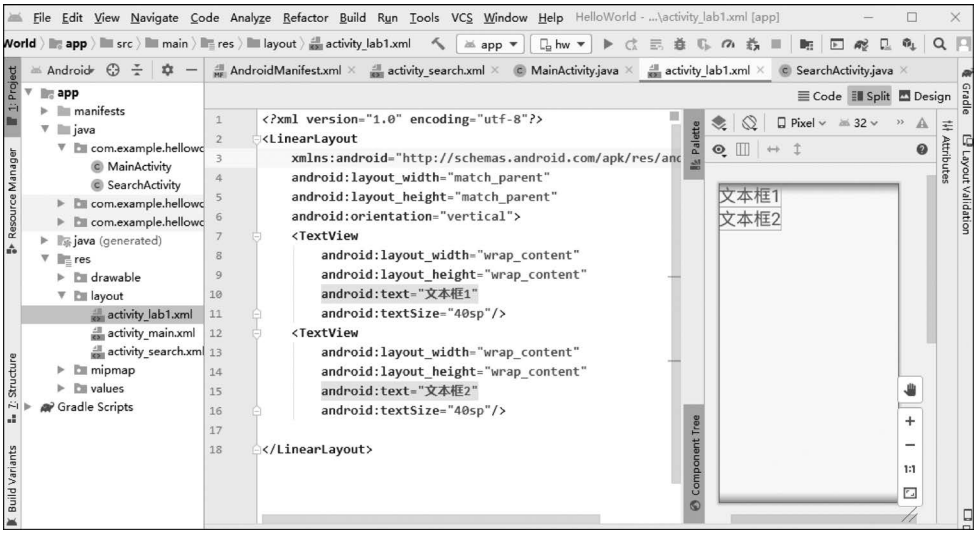


图 3-18　线性布局作为根标签 orientation 属性值为 vertical 的默认效果

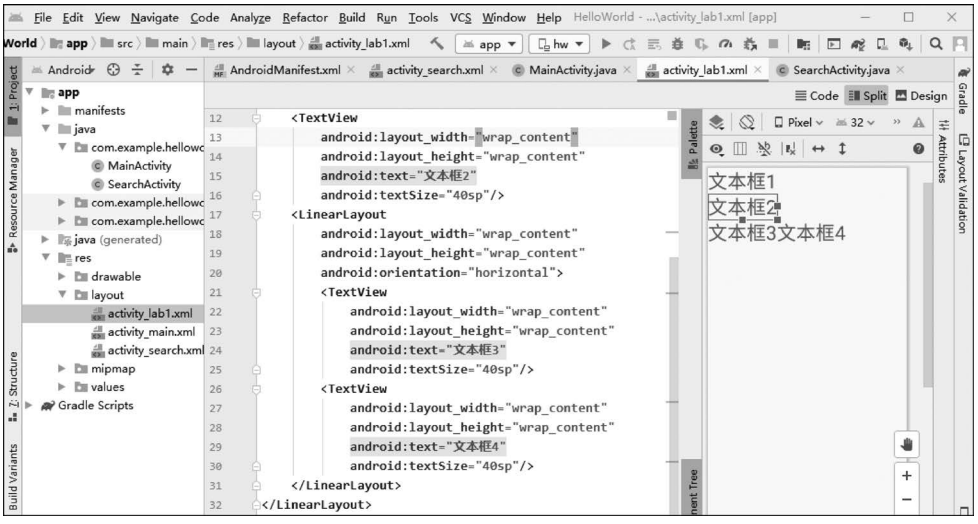


图 3-19　线性布局中 orientation 属性值设为 horizontal 的默认效果

请读者思考一下，如果现在需要在 30 行的下一行添加一个文本框，会在预览效果图的哪个位置呢？如果在第 31 行的下一行添加一个文本框，会在预览效果图的哪个位置呢？

3.1.3　相对布局

相对布局是 Android 布局控件中最灵活的一种结构，比较适合一些复杂界面。相对布局的标签名称是 RelativeLayout，相对布局容器里子控件的位置是由兄弟控件或父容器决定的，因此这种布局方式被称为相对布局。初学者在相对布局里设置控件的位置时，会感到很无助，因为不清楚在相对布局里设置控件位置的技巧。在相对布局里设置控件位置有两种方式，一种是根据父容器设置，另一种是根据相对布局里已有的控件来设置，下面将介绍这两种方式的主要属性。

1. 根据父容器设置控件位置的属性

如表 3-2 所示，这组属性用于设置相对布局中控件所在父容器的位置，其属性值为 boolean 类型，只能设置 true 或者 false。根据父容器设置定位属性示意图如图 3-20 所示，在相对布局中，如果不设置控件的位置属性，那么控件默认从屏幕的(0,0)点(即左上角)开始绘制，如果新控件中只设置了位置属性 **android:layout_centerInParent="true"**，那么该控件将设置在图 3-20 中的中央位置，以此类推，如果控件设置其中一个根据父容器定位控件属性的值为 true，会在图 3-20 所示的相应位置。

表 3-2　根据父容器设置控件位置的属性

属　性　名	属 性 说 明
android:layout_centerHorizontal	控制该子组件是否位于布局容器的水平居中
android:layout_centerVertical	控制该子组件是否位于布局容器的垂直居中
android:layout_centerInParent	控制该子组件是否位于布局容器的中央位置
android:layout_alignParentBottom	控制该子组件是否与布局容器底端对齐
android:layout_alignParentLeft	控制该子组件是否与布局容器左边对齐
android:layout_alignParentRight	控制该子组件是否与布局容器右边对齐
android:layout_alignParentTop	控制该子组件是否与布局容器顶端对齐

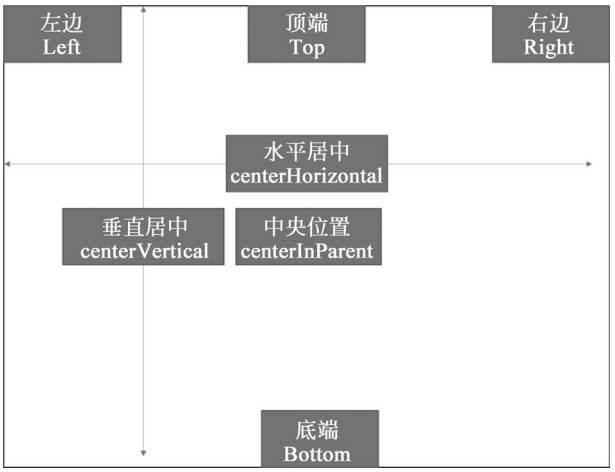


图 3-20　根据父容器定位属性示意图

2. 根据已有控件设置新控件位置的属性

如表 3-3 所示，这些属性值需要根据已给定控件的 id 值设置。根据已给定的控件设置新控件的位置示意图如图 3-21 所示。如果已经放置了一个控件 1 在靠近屏幕中心的位置，其 id 属性值为 id1，如果在新控件中设置位置属性为 **android:layout_toLeftOf="@id/id1"**，那么这个新控件会被放在区域 1 的位置，新控件靠着顶部左上角的(0,0)点开始绘制。如果设置的位置属性为 **android:layout_above="@id/id1"**，那么这个新控件会被放在区域 2，新控件的底部线与控件 1 的顶部线重合，新控件的左边会靠着屏幕的最左边，其绘制起点是控

件 1 顶部线的延长线与屏幕左边线的交叉点。如果新控件的位置属性设置为 **android:layout_toRightOf="@id/id1"**，那么这个控件会被放在区域 4，控件的绘制起点是右上角。如果该控件的位置属性设置为 **android:layout_below="@id/id1"**，那么这个控件会被放在区域 3 的位置，新控件的顶部线与控件 1 的底部线重合，新控件的左边线条与屏幕左边线条重合，其绘制起点为控件 1 底部线的延长线与屏幕左边线的交叉点。

表 3-3　根据已有控件来设置新控件位置的属性

属　性　名	属 性 说 明	属性值格式说明
android:layout_above	将该控件置于给定 id 控件的上面	="@id/控件 id"，@id 表示使用 id 资源，控件 id 表示某个已有控件设置的标识属性
android:layout_below	将该控件置于给定 id 控件的下面	
android:layout_toLeftOf	将该控件置于给定 id 控件的左边	
android:layout_toRightOf	将该控件置于给定 id 控件的右边	
android:layout_alignBaseLine	将控件的 baseline 与给定 id 控件的 baseline 对齐	
android:layout_alignTop	将控件的顶部与给定 id 控件的顶部对齐	
android:layout_alignBottom	将控件的底部与给定 id 控件的底部对齐	
android:layout_alignLeft	将控件的左边与给定 id 控件的左边对齐	
android:layout_alignRight	将控件的右边与给定 id 控件的右边对齐	

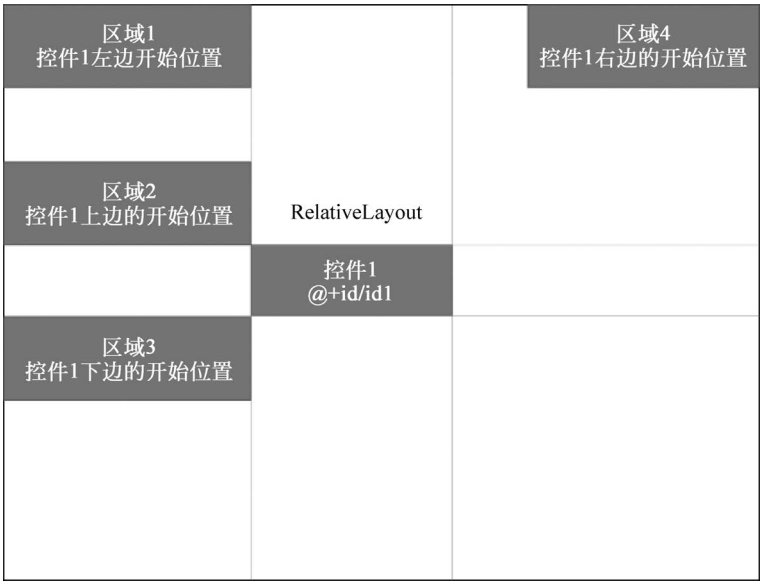


图 3-21　根据已给定的控件设置控件的位置示意图

上面讲述了在相对布局中设置控件位置的常用属性，现在尝试一下在布局文件中使用相对布局组织控件的效果和步骤。

首先，打开上一次 Android 项目或新建一个 Android 项目，在项目的 layout 目录中新建一个布局文件，布局文件名为 activity_lab2，使用 RelativeLayout 作为该布局文件的根标签。

接着在该布局中添加 5 个文本框，分别显示文本框 1、文本框 2、文本框 3、文本框 4、文本框 5，文本框 1 位于页面的中间，文本框 2 在文本框 1 的上方，文本框 3 位于文本框 1 的左边，文本框 4 位于文本框 1 的右边，文本框 5 位于文本框 1 的下方，其参考代码如代码段 3-1 所示，预览效果如图 3-22 所示。

如果现需要将文本框 2 移动到文本框 1 的正上方，文本框 3 移动到文本框 1 的左边与文本框 1 对齐，文本框 4 移动到文本框 1 的右边与文本框 1 对齐，文本框 5 移动到文本框 1 正下方，需要分别在文本框 2、文本框 3、文本框 4 和文本框 5 中添加什么属性呢？如果需要将文本框 6 放在文本框 5 的正下方，需要添加什么属性呢？

图 3-22　RelativeLayout 实现界面的运行效果

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView  android:id="@+id/tv1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="文本框 1"
        android:textSize="40sp"
        android:layout_centerInParent="true" />
    <TextView  android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="文本框 2"
        android:textSize="40sp"
        android:layout_above="@id/tv1" />
    <TextView  android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="文本框 3"
        android:textSize="40sp"
        android:layout_toLeftOf="@id/tv1" />
    <TextView  android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="文本框 4"
        android:textSize="40sp"
        android:layout_toRightOf="@id/tv1" />
    <TextView  android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="文本框 5"
        android:textSize="40sp"
        android:layout_below="@id/tv1" />
</RelativeLayout>
```

代码段 3-1　使用相对布局的参考代码

3.1.4　表格布局

表格布局(TableLayout)是以行和列的形式对控件进行管理，每一行为一个 TableRow 对象或者一个 View 控件。如果一行是一个 TableRow 对象，在 TableRow 标签中添加子控件，默认情况下，每个子控件占据一列。如果是直接放控件，一个控件就独占一行。

在使用 TableLayout 时，首先要确定页面有几行几列，每一行有几个控件，如果每行只有一个控件，那就直接在 TableLayout 中定义该控件。如果一行有多个控件，就需要在 TableRow 中定义这些控件，这些控件的宽是固定平分一行的宽度，其 layout_width 属性值为 match_parent，不能修改，即使修改为其他值也无效，其 layout_height 属性默认值为

wrap_content，可以修改为其他值。如果需要修改控件的宽度，则需要在控件中设置stretchColumns 属性，让控件可以横向伸展，或者使用 layout_span 属性合并单元格。表格布局中设置控件位置的常用属性如表 3-4 所示。

表 3-4　在 TableLayout 中设置控件位置的常用属性

属　性　名	属 性 说 明	属性值说明
android:stretchColumns	设置允许被拉伸的列的序号，该列可以行向伸展，最多占据一整行	这三个属性都是写在 TableLayout 的开始标签里，属性值的取值范围从 0 开始，0 表示第一列，如果同时设置两列则可以用逗号隔开
android:shrinkColumns	设置允许被收缩的列的序号，该列子控件的内容太多，已挤满所在行，那么该子控件的内容将往列方向显示	
android:collapseColumns	设置需要被隐藏的列的序号	
android:layout_column	设置组件所在列的列号	用在控件里的，设置组件所在的列，列号取值范围从 0 开始
android:layout_span	设置合并的单元格数	用于控件里，设置控件所占单元格数量

根据以上表格布局的特性和属性，如果需要实现一个九宫格菜单的页面效果，可以根据以下的步骤实现。

步骤 1，打开 HelloWorld 项目，在 layout 目录中新建一个布局文件，布局文件名为 activity_lab3，并将布局文件的根标签修改为 TableLayout。

步骤 2，九宫格共三行，每行有三个控件，因此在 TableLayout 中添加三个 TableRow 标签。

步骤 3，在每一个 TableRow 标签中添加三个 TextView 控件，参考代码如代码段 3-2 所示，其预览效果如图 3-23 所示。

请各位读者思考，如果需要去掉九宫格中的菜单 5 按钮，让 8 个菜单按钮围绕着中间空白处，如何修改布局文件才能实现这样的效果呢？

3.1.5　网格布局

网格布局(GridLayout)与 TableLayout 类似，但在设置子控件的位置和宽高等方面比 TableLayout 更灵活。GridLayout 是 Android 4.0 引入的一个新布局，是以行列单元格的形式排列展示控件，可以实现类似计算机键盘效果，也可以实现自动变行的标签群效果。使用 GridLayout 可以有效减少布局的深度，加快渲染速度。

在使用 GridLayout 时需要先确定整个页面的子控件的排列顺序是水平方向还是垂直方向，GridLayout 的 orientation 属性用于确定布局的方向，然后确定整个页面一共有几行几列，在 GridLayout 开始标签中使用属性 rowCount 设置页面子控件的行数，使用属性 columnCount 设置页面子控件的列数。在子控件中可以使用 layout_row 和 layout_column 来设置子控件的位置，使用 layout_rowSpan 和 layout_columnSpan 来设置子控件的宽高，使用 GridLayout 时重点的属性如表 3-5 所示。

```
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TableRow
        android:weightSum="3">
        <TextView
            android:layout_width="0dp"
            android:layout_weight="1"
            android:layout_height="wrap_content"
            android:text="菜单 1"
            android:drawableTop="@mipmap/ic_launcher"
            android:gravity="center"/>
        <TextView
            android:layout_width="0dp"
            android:layout_weight="1"
            android:layout_height="wrap_content"
            android:text="菜单 2"
            android:drawableTop="@mipmap/ic_launcher"
            android:gravity="center"/>
        <TextView
            android:layout_width="0dp"
            android:layout_weight="1"
            android:layout_height="wrap_content"
            android:text="菜单 3"
            android:drawableTop="@mipmap/ic_launcher"
            android:gravity="center"/>
    </TableRow>
    <TableRow
        android:weightSum="3">
        <TextView
            android:layout_width="0dp"
            android:layout_weight="1"
            android:layout_height="wrap_content"
            android:text="菜单 4"
            android:drawableTop="@mipmap/ic_launcher"
            android:gravity="center"/>
        <TextView
            android:layout_width="0dp"
            android:layout_weight="1"
            android:layout_height="wrap_content"
            android:text="菜单 5"
            android:drawableTop="@mipmap/ic_launcher"
            android:gravity="center"/>
        <TextView
            android:layout_width="0dp"
            android:layout_weight="1"
            android:layout_height="wrap_content"
            android:text="菜单 6"
            android:drawableTop="@mipmap/ic_launcher"
            android:gravity="center"/>
    </TableRow>
    <TableRow
            android:weightSum="3">
            <TextView
                android:layout_width="0dp"
                android:layout_weight="1"
                android:layout_height="wrap_content"
                android:text="菜单 7"
                android:drawableTop="@mipmap/ic_launcher"
                android:gravity="center"/>
            <TextView
                android:layout_width="0dp"
                android:layout_weight="1"
                android:layout_height="wrap_content"
                android:text="菜单 8"
                android:drawableTop="@mipmap/ic_launcher"
                android:gravity="center"/>
            <TextView
                android:layout_width="0dp"
                android:layout_weight="1"
                android:layout_height="wrap_content"
                android:text="菜单 9"
                android:drawableTop="@mipmap/ic_launcher"
                android:gravity="center"/>
        </TableRow>
    </TableLayout>
```

代码段 3-2　TableLayout 实现九宫格菜单参考代码

图 3-23　使用 TableLayout 实现九宫格菜单效果

表 3-5 在 GridLayout 里设置控件位置的常用属性

属 性 名	属 性 说 明	属性值说明
android:orientation	设置排列方式	用于 GridLayout 的开始标签，可选填 vertical 或者 horizontal，默认值为 vertical
android:rowCount	设置行数	用于 GridLayout 的开始标签，用于设置最大的行数
android:columnCount	设置列数	用于 GridLayout 的开始标签，用于设置最大的列数
android:layout_row	设置子控件所在行的序号	用在子控件里，设置子控件所在的行，行号取值范围从 0 开始
android:layout_column	设置子控件所在列的列号	用在子控件里，设置子控件所在的列，列号取值范围从 0 开始
android:layout_rowSpan	设置子控件纵跨几行	用于子控件，子控件高度占几行
android:layout_columnSpan	设置子控件横跨几列	用于子控件，子控件宽度占几列

接着讲解在布局文件里使用 GridLayout 实现九宫格菜单页面的效果，其步骤如下。

步骤 1，在项目的 layout 目录中创建一个新的布局文件，文件名为 activity_lab4，并将布局文件里的根标签修改为 GridLayout，通过 rowCount 属性设置九宫格的总行数，通过 columnCount 属性设置九宫格的总列数。

步骤 2，在 GridLayout 标签里添加控件，并使用 layout_row 设置控件所在行号，layout_column 设置控件所在列号，若不设置行号和列号，控件会先按行排列，一行控件的数量等于总列数时，继续添加控件，控件会自动换行。添加完 9 个控件后的参考代码如代码段 3-3 所示，其预览效果如图 3-24 所示。

请读者们思考，如果要求去掉菜单 5 按钮，让其余 8 个菜单按钮围绕着中间空白处，需要如何修改布局中的代码呢？

3.1.6 帧布局

帧布局(FrameLayout)是以层次堆叠的方式排列子控件，在帧布局中子控件是从父容器的左上角开始绘制，后面绘制的子控件会把前面的子控件覆盖掉。在帧布局中子控件没有任何的定位方式，因此它的应用场景并不多。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的话，那么同一时刻只能看到最上方的控件，后面添加的控件会覆盖前一个，虽然默认会将控件放在左上角，但是我们可以通过 **android:foregroundGravity** 属性指定它的位置。在帧布局里设置控件位置的常用属性如表 3-6 所示。

表 3-6 在 FrameLayout 中设置控件位置的常用属性

属 性 名	属 性 说 明	属性值说明
android:foreground	设置修改帧布局容器的前景图像	
android:foregroundGravity	设置前景图像显示的位置	与 layout_gravity 的取值类型一样，如果是两个方向的可以用|分割开

接下来讲解在帧布局中添加控件的默认效果，请各位读者按照以下的步骤进行操作。

```
<GridLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:rowCount="3"
    android:columnCount="3">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="菜单 1"
        android:drawableTop="@mipmap/ic_launcher"
        android:gravity="center"
        android:layout_margin="30dp" />
    <TextView
        android:layout_width="wrap_co ntent"
        android:layout_height="wrap_content"
        android:text="菜单 2"
        android:drawableTop="@mipmap/ic_launcher"
        android:gravity="center"
        android:layout_margin="30dp" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="菜单 3"
        android:drawableTop="@mipmap/ic_launcher"
        android:gravity="center"
        android:layout_margin="30dp" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="菜单 4"
        android:drawableTop="@mipmap/ic_launcher"
        android:gravity="center"
        android:layout_margin="30dp" />
<TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="菜单 5"
        android:drawableTop="@mipmap/ic_launcher"
        android:gravity="center"
        android:layout_margin="30dp" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="菜单 6"
        android:drawableTop="@mipmap/ic_launcher"
        android:gravity="center"
        android:layout_margin="30dp" />
     <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="菜单 7"
        android:drawableTop="@mipmap/ic_launcher"
        android:gravity="center"
        android:layout_margin="30dp" />
     <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="菜单 8"
        android:drawableTop="@mipmap/ic_launcher"
        android:gravity="center"
        android:layout_margin="30dp" />
     <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="菜单 9"
        android:drawableTop="@mipmap/ic_launcher"
        android:gravity="center"
        android:layout_margin="30dp" />
</GridLayout>
```

代码段 3-3　使用 GridLayout 实现九宫格菜单效果参考代码

图 3-24　使用 GridLayout 实现九宫格菜单预览图

步骤 1，在项目的 layout 目录里新建一个布局文件，其文件名为 activity_lab5，并将布局文件里的根标签修改为 FrameLayout。

步骤 2，在 FrameLayout 标签里添加三个 TextView 控件，三个控件宽高分别为 200dp、150dp、100dp，背景颜色分别为红色、绿色、蓝色，其预览效果如图 3-25 所示，其参考代码如代码段 3-4 所示。

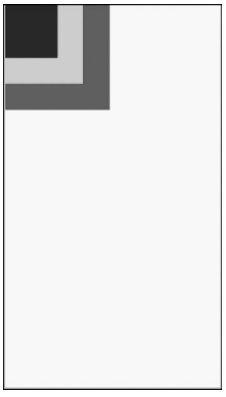


图 3-25　使用 FrameLayout 的效果

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:background="#ff0000" />
    <TextView
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:background="#00ff00" />
    <TextView
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:background="#0000ff" />
</FrameLayout>
```

代码段 3-4　使用 FrameLayout 的参考代码

3.1.7　绝对布局

绝对布局(AbsoluteLayout)通过 X、Y 坐标来设置子控件的位置，其坐标的单位是 dp，这样很容易造成换一个设备界面就不能正常显示的现象，因此，不推荐使用这个布局方式来实现界面效果，在绝对布局中设置控件位置的常用属性如表 3-7 所示。

表 3-7　在 AbsoluteLayout 中设置控件位置的常用属性

属　性　名	属 性 说 明	属性值说明
android:layout_x	设置控件的 X 坐标	一般使用数值和单位设置
android:layout_y	设置控件的 Y 坐标	

接下来讲解使用绝对布局的步骤和效果，请读者按照以下的步骤进行操作。

步骤 1，在项目的 layout 目录里新建布局文件，文件名为 activity_lab6，并将布局文件里的根标签改为 AbsoluteLayout。

步骤 2，在布局文件里添加三个 TextView 控件，并使用 layout_x、layout_y 设置控件在界面的位置，其预览效果如图 3-26 所示，其参考代码如代码段 3-5 所示。

3.1.8　约束布局

约束布局(ConstraintLayout)是一个 ViewGroup，可以在 API 9 以上的 Android 系统中使用，它的出现主要是为了解决布局嵌套过多的问题，以灵活的方式定位和调整小部件。从 Android Studio 2.3 起官方的模板默认使用 ConstraintLayout 作为布局文件的根标签。在开发过程中经常会遇到一些复杂的 UI，可能会出现布局嵌套过多的问题，嵌套得越多，设备绘制视图所需的时间和计算功耗也就越多。在使用过程中，可以把 ConstraintLayout 看

作一个更强大的 RelativeLayout，它提供了更多的 API 来约束控件的相对关系，更容易满足复杂的页面布局，在约束布局里设置控件位置的属性有以下几类。

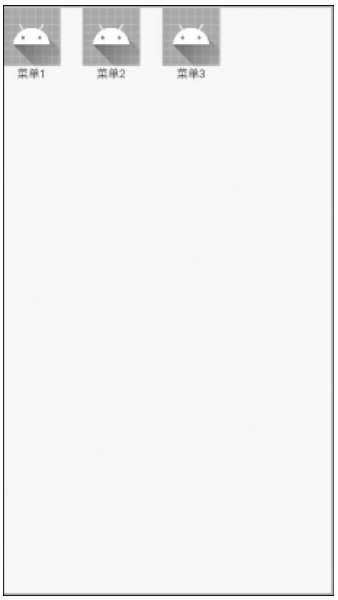


图 3-26　使用 AbsoluteLayout 的预览效果

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="菜单 1"
        android:drawableTop="@mipmap/ic_launcher"
        android:gravity="center"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="菜单 2"
        android:drawableTop="@mipmap/ic_launcher"
        android:gravity="center"
        android:layout_x="100dp"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="菜单 3"
        android:drawableTop="@mipmap/ic_launcher"
        android:gravity="center"
        android:layout_x="200dp"/>
</AbsoluteLayout>
```

代码段 3-5　使用 AbsoluteLayout 的参考代码

1. 相对定位

相对定位设置控件的位置，即使用某个已在约束布局中的控件作为参照物来设置新添加控件的位置，其常用属性如表 3-8 所示。

表 3-8　在相对布局中设置新控件相对位置的常用属性值

属　性　名	属 性 说 明	属性值说明
layout_constraintLeft_toLeftOf	设置新控件的左边在参照控件的左边	属性值为使用已有控件的 id 或者 parent，使用已有控件 id 的格式为@id/控件 id，将 layout_constraintLeft_toLeftOf、layout_constraintRight _ toRightOf、layout _ constraintTop_toTopOf、layout_constraintBottom_toBottomOf 四个属性值同时设置为 parent 即可把控件设置为屏幕居中
layout_constraintLeft_toRightOf	设置新控件的左边在参照控件的右边	
layout_constraintRight_toLeftOf	设置新控件的右边在参照控件的左边	
layout_constraintRight_toRightOf	设置新控件的右边在参照控件的右边	
layout_constraintTop_toTopOf	设置新控件的顶部在参照控件的顶部	
layout_constraintTop_toBottomOf	设置新控件的顶部在参照控件的底部	
layout_constraintBottom_toTopOf	设置新控件的底部在参照控件的顶部	
layout_constraintBottom_toBottomOf	设置新控件的底部在参照控件的底部	
layout_constraintBaseline_toBaselineOf	设置新控件的基线根据参照控件设置	
layout_constraintStart_toEndOf	设置新控件文本的开始位置在参照控件文本结束的位置	
layout_constraintStart_toStartOf	设置新控件文本的开始位置在参照控件文本开始的位置	
layout_constraintEnd_toStartOf	设置新控件文本的结束位置在参照控件文本开始的位置	
layout_constraintEnd_toEndOf	设置新控件文本的结束位置在参照控件文本结束的位置	

2. 居中偏移(bias)

将控件设为屏幕居中后,可以再为控件设置偏移量,其具体属性如表 3-9 所示。

表 3-9 在约束布局中设置控件居中偏移量的常用属性

属 性 名	属 性 说 明	属性值说明
layout_constraintHorizontal_bias	水平偏移	取值范围为 0～1,0 表示最左,1 表示最右
layout_constraintVertical_bias	垂直偏移	取值范围为 0～1,0 表示最上方,1 表示最下方

3. 圆形定位(角度定位)

圆形定位可以让一个控件以另一个控件的中心为中心,使用其相对于该中心点的距离和角度来设置控件的位置,其常用的属性如表 3-10 所示,这三个常用属性一般需要同时使用才能有效果。

表 3-10 圆形定位常用属性

属 性 名	属 性 说 明	属性值说明
app:layout_constraintCircle	设置控件的参照物	属性值一般使用参照控件的 id
app:layout_constraintCircleAngle	设置控件选择的角度	取值范围为 0～360,最上方为 0 度,默认就是 0 度,顺时针开始算
app:layout_constraintCircleRadius	设置两个控件中心点的距离	一般使用数值和单位设置

4. 边距

在约束布局中设置控件间的边距前必须先为控件设置一个相对位置,其边距值一般大于或等于 0,其常用的属性如表 3-11 所示。

表 3-11 在约束布局中设置控件间边距的常见属性

属 性 名	属 性 说 明	属性值说明
android:layout_marginStart	设置控件开始位置的边距	一般使用数值和单位作为边距值
android:layout_marginEnd	设置控件结束位置的边距	
android:layout_marginLeft	设置控件左边位置的边距	
android:layout_marginTop	设置控件顶部位置的边距	
android:layout_marginRight	设置控件右边位置的边距	
android:layout_marginBottom	设置控件底部位置的边距	
layout_goneMarginStart	约束控件的可见性被设置为 gone 时使用的 margin 值	一般使用数值和单位作为边距值
layout_goneMarginEnd		
layout_goneMarginLeft		
layout_goneMarginTop		
layout_goneMarginRight		
layout_goneMarginBottom		

5. 尺寸约束

在约束布局中可以对控件的宽高进行尺寸约束,在约束布局中设置控件的宽高时一般

推荐直接设置指定的尺寸，或者使用 wrap_content 让控件自己计算大小，当控件的高度或宽度为 wrap_content 时，可以使用表 3-12 中的属性来控制最大、最小的高度或宽度。需要注意的是在约束布局中不推荐使用 match_parent，若需要将控件宽或高设置为填满父容器的效果，可以先将宽或高设置为 0dp，然后使用 app:layout_constraintDimensionRatio＝"1：1"属性设置控件宽高比。

表 3-12　设置控件尺寸约束的常用属性

属　性　名	属 性 说 明	属性值说明
android:minWidth	最小的宽度	一般使用数值和单位设置控件宽高的取值范围
android:minHeight	最小的高度	
android:maxWidth	最大的宽度	
android:maxHeight	最大的高度	
app:constrainedWidth	设置控件的宽度值为受约束的	当 ConstraintLayout 为 1.1 版本以下时，使用上面四个属性时需要加上这两个属性强制约束
app:constrainedHeight	设置控件的高度值为受约束的	
app:layout_constraintDimensionRatio	设置控件宽高比	属性值设置为类似 1：1 的比值

6. 约束链

控件自己能够在水平或者垂直方向相互约束而组成一条链，这条链就是约束链，约束链是由开头的控件进行属性控制，其常用的属性如表 3-13 所示。

表 3-13　约束链常用属性值

属　性　名	属 性 说 明	属性值说明
app:layout_constraintHorizontal_chainStyle	设置约束链的样式	该属性的可选属性值含义如下：packed：控件紧挨在一起。还可以通过 bias 属性设置偏移量。spread：均匀分布控件。spread_inside：均匀分布控件，但是两边控件贴边
app:layout_constraintHorizontal_weight	水平权重	当宽度设为 0dp 时使用
app:layout_constraintVertical_weight	垂直权重	当长度设为 0dp 时使用

7. 约束布局的辅助工具

约束布局还提供了一些辅助工具帮我们更好地设置约束布局中控件的位置，常用的辅助工具如下。

1）Group

Group 可以把多个控件归为一组，方便隐藏或显示一组控件，以下是其使用的格式。

```
<android.support.constraint.Group
android:id="@+id/group"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:visibility="invisible"
app:constraint_referenced_ids="TextView1,TextView3"/>
```

上述代码中 visibility 属性用于设置控件是否可见，如果其值为 invisible，则表示控件是隐藏但存在，如果其值为 gone，则表示控件既是隐藏，也是不存在。

2）Guideline

Guideline 是约束布局中一个特殊的辅助布局类，可以创建水平或垂直的参考线，其他的控件可以根据这个参考线来进行布局，它本质是不可见的控件，参考线的位置属性如表 3-14 所示。

表 3-14　Guideline 常用属性

属　性　名	属 性 说 明	属性值说明
orientation	设置控件的参照物	vertical/horizontal
layout_constraintGuide_begin	指定距离左/上边开始的固定位置	一般使用数值和单位设置其位置
layout_constraintGuide_end	指定距离右/下边开始的固定位置	
layout_constraintGuide_percent	指定位于布局中所在的百分比	只需要数值

3）Barrier

Barrier 可以将多个控件看作一个整体，添加另一个控件限制最大宽/高的约束，例如有 3 个控件 A、B、C，C 在 AB 的右边，但是 AB 的宽是不固定的，这个时候 C 无论约束在 A 的右边或者 B 的右边都不对，当出现这种情况时可以用 Barrier 来解决。Barrier 可以在多个控件的一侧建立一个屏障，用于放置新的控件，Barrier 常用属性如表 3-15 所示。

表 3-15　Barrier 常用属性

属　性　名	属 性 说 明	属性值说明
constraint_referenced_ids	使用多个控件，看作一个整体	使用已有控件的 id，多个控件的 id 用逗号隔开
app:barrierDirection	设置 Barrier 的方向	可设置的值有 bottom、end、left、right、start、top

接下来使用约束布局实现如图 3-27 的界面效果，A 控件的宽为 100dp，B 控件的宽为 50dp，C 控件的宽为 60dp，读者可以根据以下步骤进行操作。

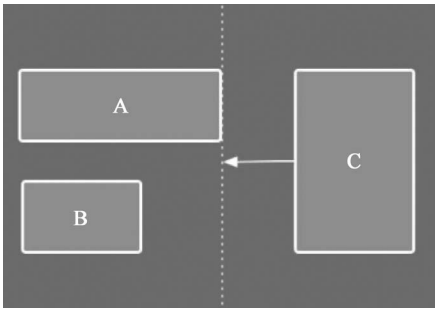


图 3-27　使用约束布局所需实现的设计图效果

步骤 1，在项目的 layout 目录里新建一个布局文件，布局文件名为 activity_lab7。

步骤 2，在约束布局中添加控件 A（菜单 1）、控件 B（菜单 2），然后再添加一个 Barrier，最后添加控件 C（菜单 3），其最终的代码如代码段 3-6 所示，其预览效果如图 3-28 所示。

如果现在需要在约束布局中添加控件 D，控件 D 的宽为 60dp，将控件 D 放在控件 C 的上方，需要对现在布局文件的代码进行怎样的修改呢？

```
<? xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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"
    xmlns:APP="http://schemas.android.com/apk/res -auto">
    <TextView
        android:id="@+id/tv_itme1"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:text="菜单 1"
        android:drawableTop="@mipmap/ic_launcher"
        android:background="#ff0000"
        android:gravity="center"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintHorizontal_bias="0.2" />
    <TextView
        android:id="@+id/tv_itme2"
        android:layout_width="50dp"
        android:layout_height="wrap_content"
        android:text="菜单 2"
        android:drawableTop="@mipmap/ic_launcher"
        android:background="#00ff00"
        android:gravity="center"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@id/tv_itme1"
        android:layout_marginLeft="60dp"
        android:layout_marginTop="20dp" />
    <androidx.constraintlayout.widget.Barrier
        android:id="@+id/bar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:constraint_referenced_ids="tv_itme1,tv_itme2"
        android:orientation="horizontal"
        app:barrierDirection="right" />
    <TextView
        android:id="@+id/tv_itme3"
        android:layout_width="80dp"
        android:layout_height="wrap_content"
        android:text="菜单 3"
        android:drawableTop="@mipmap/ic_launcher"
        android:background="#0000ff"
        android:gravity="center"
        app:layout_constraintLeft_toRightOf="@id/bar"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        android:layout_marginLeft="50dp" />
</androidx.constraintlayout.widget.ConstraintLayout >
```

代码段 3-6　使用约束布局的参考代码

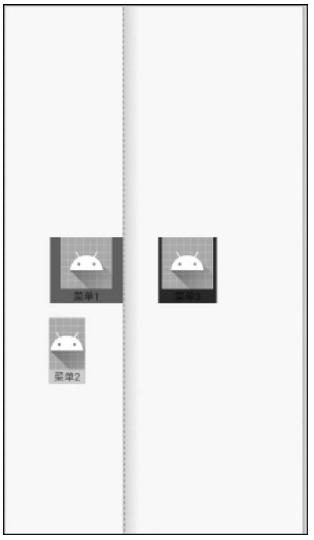
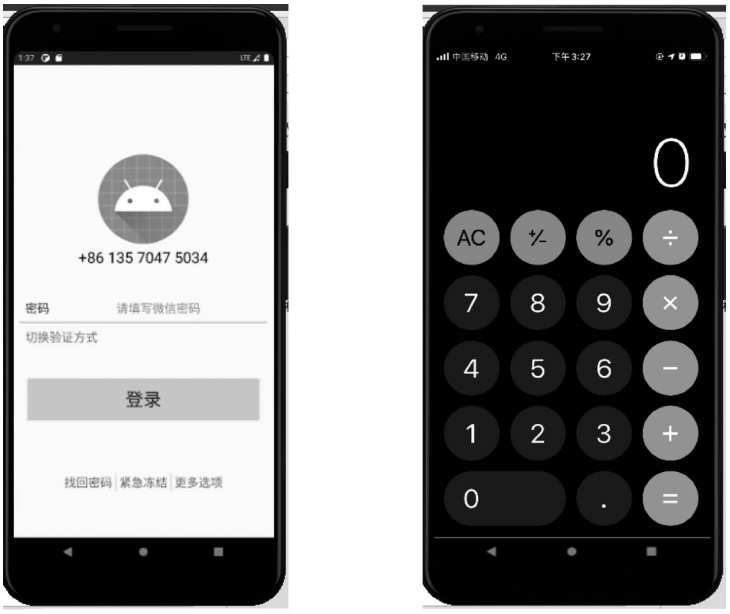


图 3-28　约束布局实现的页面预览效果

3.2　课堂学习任务：使用相应布局实现页面效果

相信读者在课前学习任务中已经初步掌握了 Android 中常用布局的使用技巧了，接着通过实现登录页面和计算器页面的效果来检测和拓展课前学习的知识，登录页面和计算器页面的效果如图 3-29 所示。在动手实现界面效果之前，需要分析出界面中包含哪些控件、控件的排列规则是什么。

由图 3-29 可知，登录页面中包含了图像控件、文本控件、文本输入控件、按钮控件，控件总体呈线性垂直排列，因此可以考虑使用线性布局或者相对布局来实现该页面的效果。计算器页面中包含了文本框控件或者输入文件框控件、按钮控件，呈现网格型排列。接下来请



(a) 登录页面效果　　(b) 计算器页面效果

图 3-29　登录页面和计算器页面效果

各位读者使用相应的布局来实现这两个页面的效果，并总结使用这些布局的技巧。

3.2.1　使用线性布局实现登录页面

线性布局的常用属性已经在课前学习任务里讲解过了，在此不再累述，在实现登录页面的过程中不清楚实现效果所需属性的读者，可以自行从课前学习任务中进行查询，接下来请读者参考以下的步骤来实现登录页面的效果。

第一步，分析界面结构，确定界面的布局方式和界面所包含的控件。

通过观察，这个界面的控件是从上到下排列的，第一个控件是显示头像的 ImageView 控件，接着是一个显示电话号码的 TextView 控件，接下来一行是用户填写密码的 TextView 控件和 EditText 控件，接下来是切换验证方式的控件，紧接着是登录的按钮，最后一行是找回密码、紧急冻结、更多选项的底部按钮。

第二步，新建布局文件，并将布局文件里的根标签修改为 LinearLayout。新建布局文件的步骤是选中 layout 文件夹，单击右键，接着选择 New，再选择 Layout Resource File，在弹框中填写布局文件的信息，如图 3-30 所示。只需要填写文件名这一项信息，接着单击 OK 按钮，就会打开新建的布局文件，代码如图 3-31 所示。然后需要将新建布局文件里的根标签和结束标签修改为 LinearLayout，并在根标签中添加 orientation 属性，并设置该属性值为 vertical，其最终代码如图 3-32 所示。

第三步，根据上一节在布局中定义控件的步骤，添加显示头像的 ImageView 控件到布局中，代码及其预览效果如图 3-33 所示。

第四步，在布局中添加显示用户手机号码的 TextView 控件，其代码及效果如图 3-34 所示。

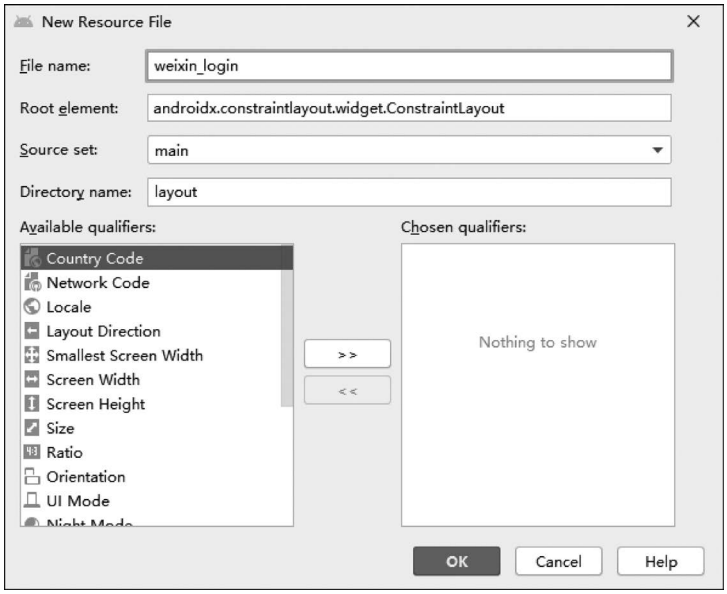


图 3-30　填写布局文件信息对话框

图 3-31　新建布局文件的代码

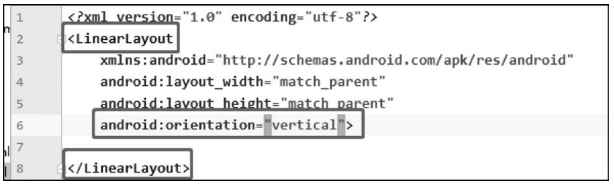


图 3-32　修改根标签后的布局文件代码

图 3-33　添加显示头像控件的代码

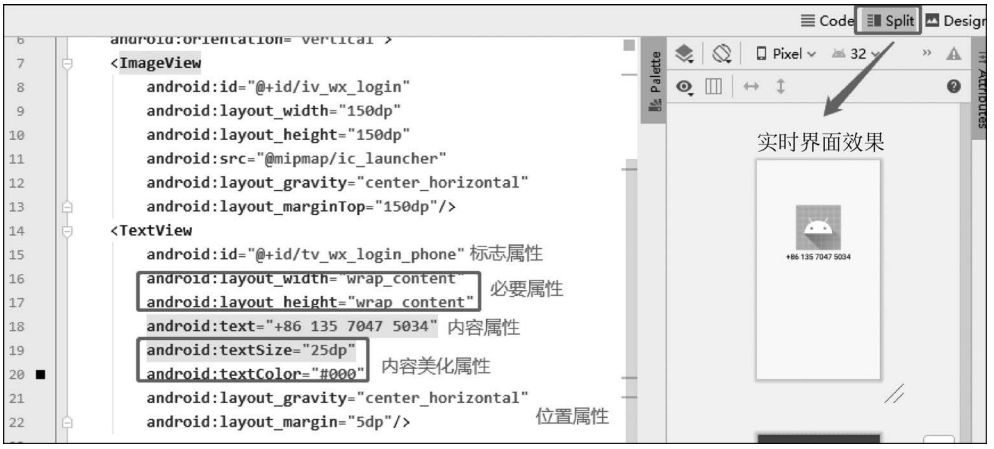


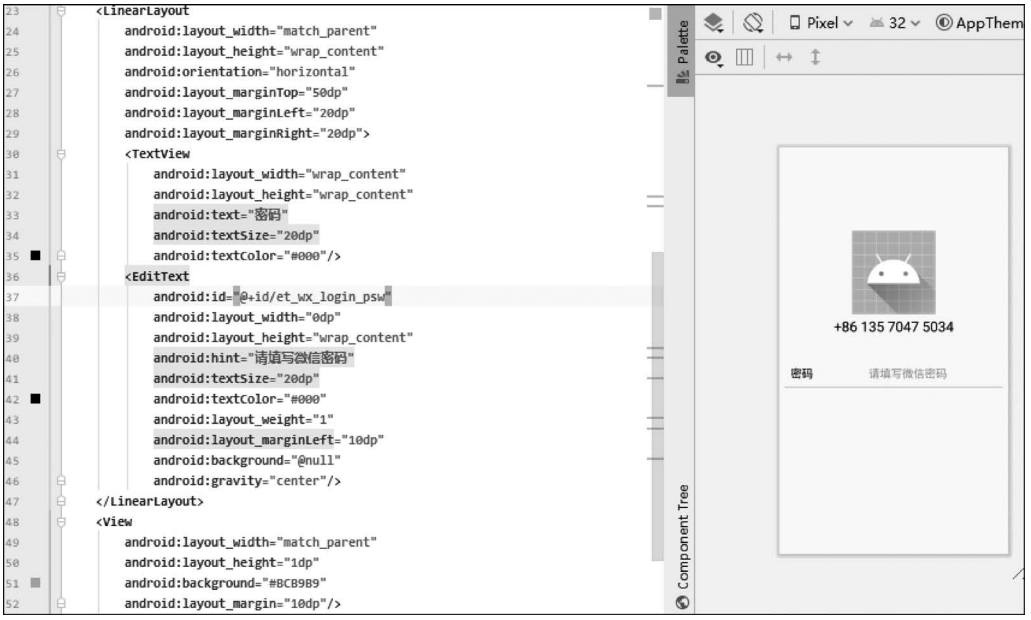
图 3-34　显示手机号码控件的代码

第五步，添加用户输入密码的控件及其提示控件，这两个控件在同一行，因此可以使用一个线性布局将它们包裹起来，然后使用 weight 设置控件的宽度。由于线性布局是水平方向，而分割线在最下面，因此不能使用 divide 属性设置它们的分割线，这时候可以使用一个 View 控件实现分割线功能，代码及其界面效果如图 3-35 所示。只有 EditText 控件里设置了 id 属性，因为只有这个控件的内容是需要被源代码控制的，其他控件不需要，因此不需要添加该属性。

图 3-35　输入密码行代码及其预览效果

第六步，添加“切换验证方式”和“登录”按钮控件，“切换验证方式”按钮选用 TextView 控件实现，在下一章会使用下拉列表控件代替，代码及其预览效果如图 3-36 所示。

第七步，添加底部菜单栏，使用三个 TextView 控件实现页面底部的菜单，同样使用一个 LinearLayout 布局包裹这三个控件，控件间的分割线可以使用线性布局中的 divider 属性设置。使用 divider 属性前需要准备一张分割线的图片，这里可以使用 shape 资源制作分割线，shape 资源制作过程是选中 drawable 目录，右击，选择 New，再选择 Drawable Resource File，在弹框中填写文件名和根标签，其他选项默认，如图 3-37 所示，填写文件信




```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginTop="50dp"
    android:layout_marginLeft="20dp"
    android:layout_marginRight="20dp">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="密码"
        android:textSize="20dp"
        android:textColor="#000"/>
    <EditText
        android:id="@+id/et_wx_login_psw"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="请填写微信密码"
        android:textSize="20dp"
        android:textColor="#000"
        android:layout_weight="1"
        android:layout_marginLeft="10dp"
        android:background="@null"
        android:gravity="center"/>
</LinearLayout>
<View
    android:layout_width="match_parent"
    android:layout_height="1dp"
    android:background="#BCB9B9"
    android:layout_margin="10dp"/>
```

图 3-36　“切换验证方式”和“登录”按钮控件参考代码及其预览效果

息后单击 OK 按钮，在打开的 shape 文件中编写分割线代码，如图 3-38 所示，底部菜单栏的布局代码如图 3-39 所示。

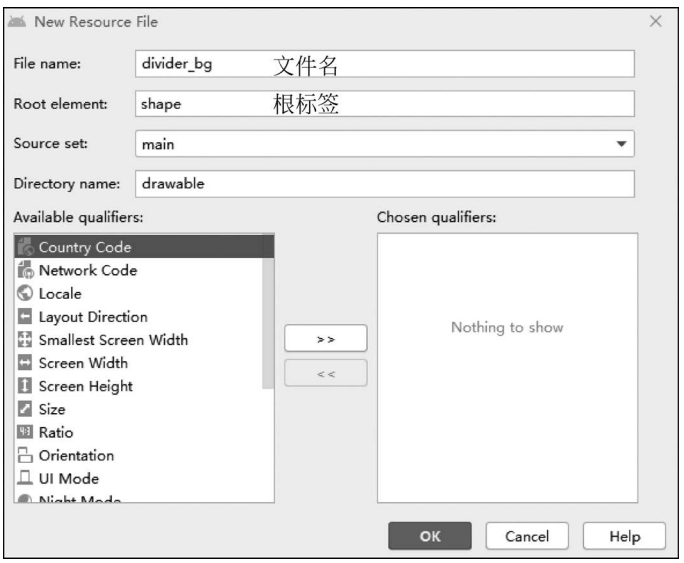


图 3-37　填写 shape 资源文件信息

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <solid android:color="#D3D1D1"/>
    <size android:width="1dp"/>
</shape>
```

图 3-38　shape 资源文件中的代码


```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="40dp"
    android:orientation="horizontal"
    android:divider="@drawable/divider_bg"
    android:showDividers="middle"
    android:dividerPadding="5dp"
    android:gravity="center"
    android:layout_marginTop="80dp">
    <TextView
        android:id="@+id/tv_wx_login_find"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="找回密码"
        android:textSize="20dp"
        android:padding="5dp"/>
    <TextView
        android:id="@+id/tv_wx_login_emerge"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="紧急冻结"
        android:textSize="20dp"
        android:padding="5dp"/>
    <TextView
        android:id="@+id/tv_wx_login_more"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="更多选项"
        android:textSize="20dp"
        android:padding="5dp"/>
</LinearLayout>
```

图 3-39　底部菜单栏的代码

第八步，为该布局创建一个 Activity，步骤是选中项目中的 java/主包名（一般是 java 目录下的第一个包），接着右键，选择 New，再选择 New Java Class，在弹框中填写类名，接着双击 Class，在打开的类中让该类继承 Activity，然后重写 onCreate()方法，在该方法中通过 setContentView()方法绑定布局，其代码如图 3-40 所示。

```
package com.example.lab1;

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

import androidx.annotation.Nullable;

public class WeiXinActivity extends Activity {
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.weixin_login);
    }
}
```

这部分导包代码是自动生成的，不需要写。

图 3-40　绑定布局的 Activity 代码

第九步，将第八步创建的 Activity 设置成项目的启动页面，清单文件的代码如图 3-41 所示。

第十步，运行项目到手机模拟器中，运行效果如图 3-42 所示。

完成上述例子后，相信读者对使用线性布局已经有了比较深刻的认识。在使用线性布局实现界面效果时，首先要确定界面里控件的排列方向是水平方向还是垂直方向，接着在布局文件里添加内容控件，在添加控件时先编写控件的内容属性，再编写美化属性，最后根据设计图的效果在控件中使用相应的位置属性设定控件在界面中的位置。对于那些简单布局

的界面，线性布局就能够实现，但是要实现一些不规则的界面效果时，线性布局就显得比较臃肿和复杂，接下来介绍一种灵活的布局方式——相对布局。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.lab1">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/main_index_my_pressed"
        android:roundIcon="@mipmap/main_index_my_pressed"
        android:label="Lab1"
        android:configChanges="locale"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".SearchActivity">
        </activity>
        <activity android:name=".WeiXinActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

图 3-41　在清单文件中设置 Activity 作为项目启动页面的代码

图 3-42　登录页面的运行效果

3.2.2　使用相对布局实现登录页面

相信读者在课前学习任务中已经学习了相对布局的常用属性及其使用的技巧，在使用相对布局实现界面效果时，需要注意的是在添加控件时建议给控件设置 id 属性，方便设置控件在页面中的位置。接下来请读者参照以下步骤使用相对布局在布局文件中实现微信登录界面的效果。

步骤 1，新建一个布局文件，其文件命名为 weixin_login_r. xml，并将其根标签修改为 RelativeLayout，其代码如图 3-43 所示。

```
weixin_login.xml × | weixin_login_r.xml × | divider_bg.xml × | styles.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">
</RelativeLayout>
```

图 3-43　布局文件的代码

步骤 2，在上一步的布局文件中第 5 行代码右尖括号后回车，然后添加头像控件到界面适当的位置，添加完头像控件后完整的参考代码如图 3-44 所示。

步骤 3，在第 2 步的基础上，在第 12 行代码的下边，添加显示手机号码的控件，可以使用 android:layout_below 属性将显示手机号码的控件设置到显示头像控件的下方，添加后

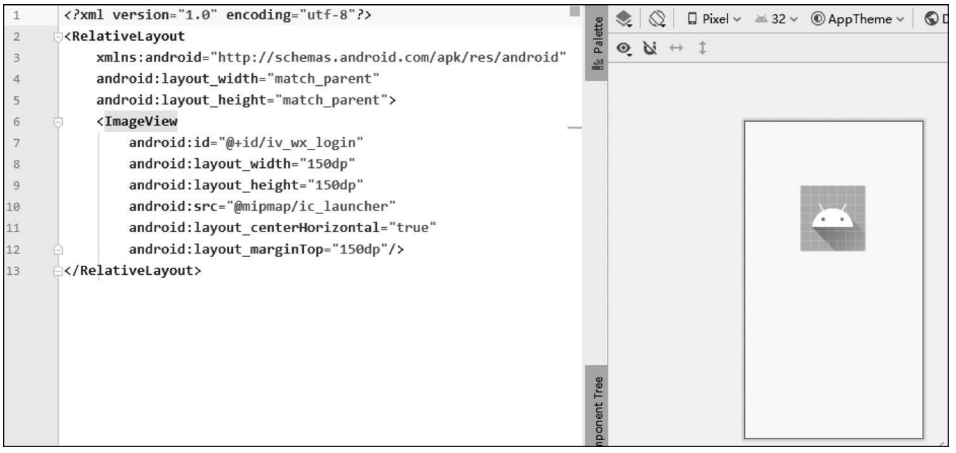


图 3-44　添加头像控件的参考代码及其预览效果

的参考代码如图 3-45 所示。

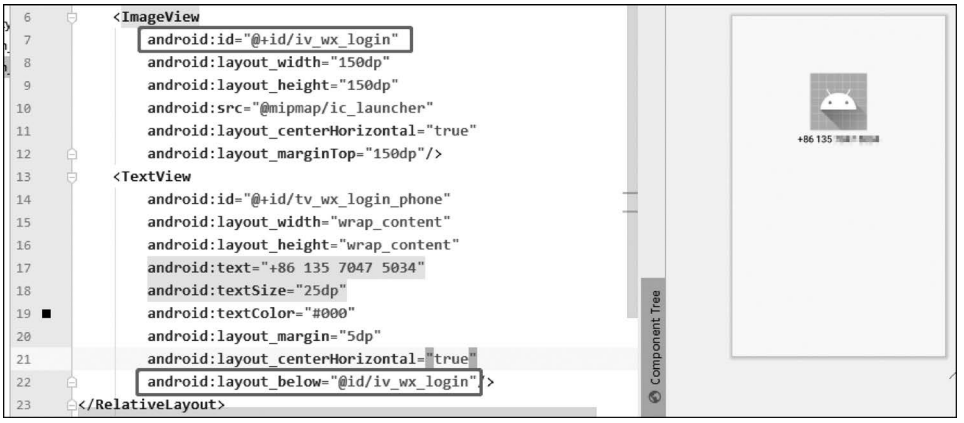


图 3-45　添加显示手机号码控件的参考代码及其预览效果

步骤 4，在上一步的基础上添加用户输入密码控件，使用 android:layout_below 属性将新添加的控件设置在显示手机号码控件的下方，其参考代码和预览效果如图 3-46 所示。

步骤 5，添加“切换验证方式”和“登录”按钮的控件，同样可以使用 android:layout_below 属性将这两个属性添加到相应控件的下面，其参考代码和预览效果如图 3-47 所示。

步骤 6，添加底部菜单栏控件，由于底部菜单栏包含三个单独的控件，而且是横向排列的，控件之间有分割线，所以选择使用一个线性布局包裹底部菜单栏的三个控件，其代码和预览效果如图 3-48 所示。

步骤 7，打开上一节新建的 WeiXinActivity，将 onCreate()方法中的 setContentView()方法中的参数修改为本节课新建的布局资源，其代码如图 3-49 所示。

步骤 8，将项目运行在手机模拟器中，运行效果如图 3-50 所示。

通过使用线性布局和相对布局实现了相同的界面效果，进行对比可以发现，线性布局实现元素呈线性方式排列的界面效果，比较方便，相对布局实现元素排列不规则的界面效果，比较灵活，两种布局也可以相互嵌套使用。

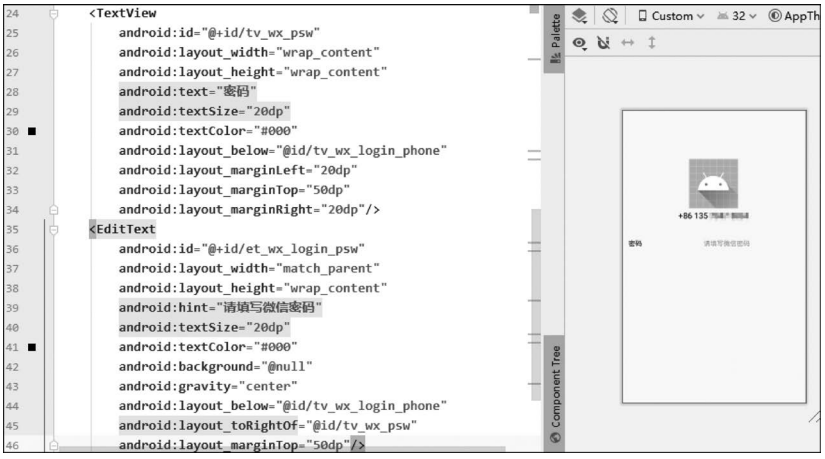


图 3-46　添加用户输入密码控件的代码及其预览效果

图 3-47　添加“切换验证方式”和“登录”按钮的控件的参考代码和预览效果

图 3-48　底部导航栏的参考代码和预览效果

```
public class WeiXinActivity extends Activity {
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.weixin_login_r);
    }
}
```

图 3-49　在 Activity 中绑定布局资源的参考代码

3.2.3　使用表格布局实现计算器页面

在课前学习任务中，读者已经学习完了表格布局常用属性和使用技巧，接下来读者可以参考以下的步骤使用 TableLayout 来实现一个计算器界面，其效果如图 3-51 所示。从图 3-51 可知，该界面第一行和第二行都是一个用于显示文本的控件，第三行到第六行都是 4 个按钮，第七行是 3 个按钮，下面是实现该界面的参考步骤。

图 3-50　运行效果

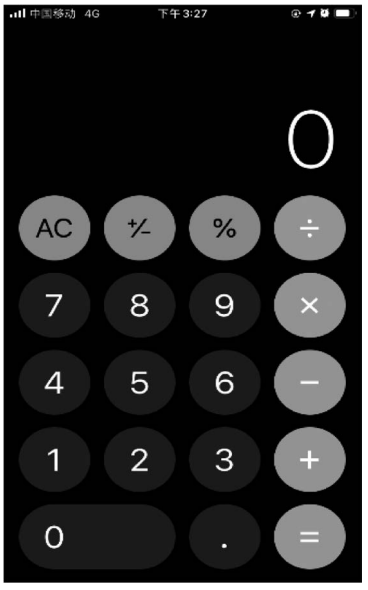


图 3-51　计算器界面效果

步骤 1，新建一个布局文件，并将布局文件里的根标签修改为 TableLayout，其参考代码如图 3-52 所示。

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

</TableLayout>
```

图 3-52　计算器界面效果布局根标签代码

步骤 2，修改页面的背景颜色为黑色，添加第一行和第二行显示数字和表达式的控件，并设置文字颜色为白色，其参考代码和预览效果如图 3-53 所示。

图 3-53　添加第一行和第二行显示数字和表达式的控件的参考代码和预览效果

如图 3-53 所示，刚才添加的两个控件分别占一行，它们的宽度自动适应了屏幕的宽度。

步骤 3，使用 TableRow 添加第三行按钮，其参考代码和预览效果如图 3-54 所示。在 TableRow 标签中添加控件，这些控件会被放置在同一行，如果所有控件的宽度和大于屏幕的宽度，那么超出屏幕宽度的控件用户将看不到。

图 3-54　添加第三行按钮参考代码及其预览效果

步骤 4，参考步骤 3 添加第三行按钮的代码，继续添加计算器第四行的按钮控件，其参考代码和预览效果如图 3-55 所示。

步骤 5，使用同样的方法，添加第五行的按钮控件，其参考代码和预览效果如图 3-56 所示。

步骤 6，使用同样的方法，添加第六行的按钮控件，其参考代码和预览效果如图 3-57 所示。

```
<TableRow
    android:layout_marginTop="10dp">
    <Button
        android:id="@+id/bt_7"
        android:text="7"
        android:textSize="50dp"
        android:layout_marginLeft="15dp"/>
    <Button
        android:id="@+id/bt_8"
        android:text="8"
        android:textSize="50dp"
        android:layout_marginLeft="10dp"/>
    <Button
        android:id="@+id/bt_9"
        android:text="9"
        android:textSize="50dp"
        android:layout_marginLeft="10dp"/>
    <Button
        android:id="@+id/bt_multiple"
        android:text="×"
        android:textSize="50dp"
        android:layout_marginLeft="10dp"/>
</TableRow>
```

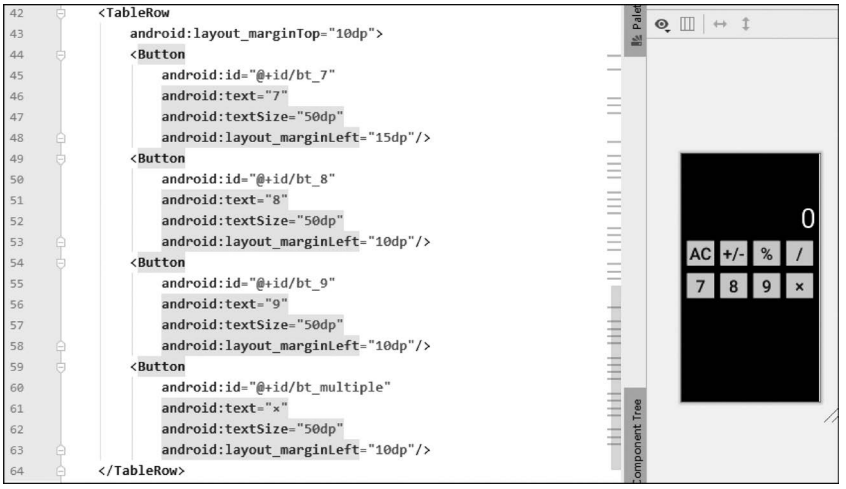


图 3-55　添加第四行按钮参考代码及其预览效果

```
<TableRow
    android:layout_marginTop="10dp">
    <Button
        android:id="@+id/bt_4"
        android:text="4"
        android:textSize="50dp"
        android:layout_marginLeft="15dp"/>
    <Button
        android:id="@+id/bt_5"
        android:text="5"
        android:textSize="50dp"
        android:layout_marginLeft="10dp"/>
    <Button
        android:id="@+id/bt_6"
        android:text="6"
        android:textSize="50dp"
        android:layout_marginLeft="10dp"/>
    <Button
        android:id="@+id/bt_sub"
        android:text="-"
        android:textSize="50dp"
        android:layout_marginLeft="10dp"/>
```

图 3-56　添加第五行按钮参考代码及其预览效果

```
<TableRow
    android:layout_marginTop="10dp">
    <Button
        android:id="@+id/bt_3"
        android:text="3"
        android:textSize="50dp"
        android:layout_marginLeft="15dp"/>
    <Button
        android:id="@+id/bt_2"
        android:text="2"
        android:textSize="50dp"
        android:layout_marginLeft="10dp"/>
    <Button
        android:id="@+id/bt_1"
        android:text="1"
        android:textSize="50dp"
        android:layout_marginLeft="10dp"/>
    <Button
        android:id="@+id/bt_add"
        android:text="+"
        android:textSize="50dp"
        android:layout_marginLeft="10dp"/>
</TableRow>
```

图 3-57　添加第六行按钮参考代码及其预览效果

步骤 7，参照添加第三行按钮的代码，添加第七行按钮控件，其参考代码和预览效果如图 3-58 所示。

```
<TableRow
    android:layout_marginTop="10dp">
    <Button
        android:id="@+id/bt_0"
        android:text="0"
        android:textSize="50dp"
        android:layout_marginLeft="15dp"
        android:layout_span="2"/>
    <Button
        android:id="@+id/bt_point"
        android:text="."
        android:textSize="50dp"
        android:layout_marginLeft="10dp"/>
    <Button
        android:id="@+id/bt_equal"
        android:text="="
        android:textSize="50dp"
        android:layout_marginLeft="10dp"/>

</TableRow>
```

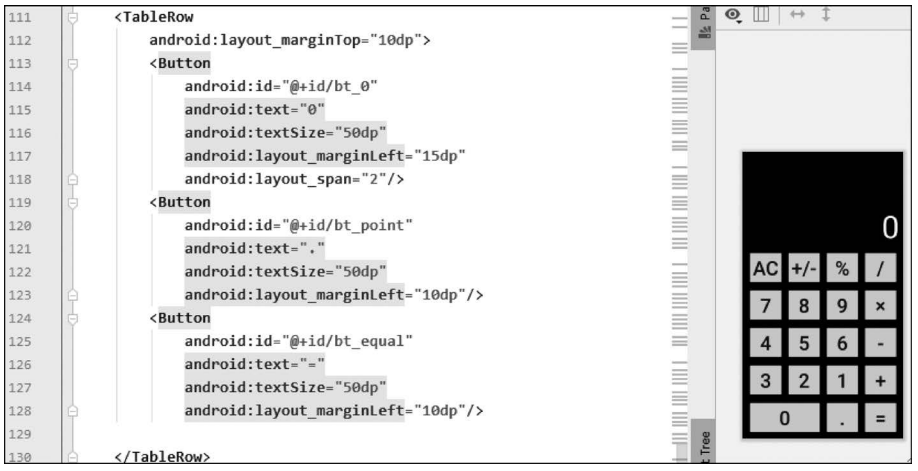


图 3-58　添加第七行按钮参考代码及其预览效果

步骤 8，可以暂时将计算器的布局资源与 WeiXinActivity 进行绑定，其参考代码如图 3-59 所示。

```
public class WeiXinActivity extends Activity {
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.caculator_activity);
    }
}
```

图 3-59　将计算器的布局资源与 WeiXinActivity 进行绑定的参考代码

步骤 9，将应用程序运行到手机模拟器中，其运行效果如图 3-60 所示。

图 3-60　运行到手机模拟器的效果

计算器界面结构与题目要求的计算器界面结构效果基本一致，但是按钮的形状不一致，在后面的章节中会讲到使用 selector 资源来美化按钮控件。

3.2.4　使用网格布局实现计算器页面

在课前任务中读者已经学习过 GridLayout 的常用属性和使用技巧了，接着通过完成计算器界面效果来深入学习 GridLayout 的相关属性和使用技巧，并对比 TableLayout 实现同样的效果时有何异同之处，请读者参考以下的步骤完成计算器页面的效果。

步骤 1，新建一个布局文件，其文件名为 calculator_grid，然后将根标签修改为 GridLayout，设置子控件的排列方向为垂直方向，设置页面的最大行数为 7，最大列数为 4，修改背景颜色为黑色，其参考代码如图 3-61 所示。

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:rowCount="7"
    android:columnCount="4"
    android:background="#000">

</GridLayout>
```

图 3-61　布局文件根标签为 GridLayout 的参考代码

步骤 2，添加第一行和第二行显示数字和表达式的控件，其参考代码及其预览效果如图 3-62 所示，由图可知，每一个控件都可以通过 android:layout_row 属性设置控件所在的行，需要注意的是行号从 0 开始，如果一个控件独占一行，可以使用 android:layout_columnSpan 设置控件横跨多少列。

```
<TextView
    android:id="@+id/tv_caculator_exp"
    android:layout_row="0"
    android:layout_columnSpan="4"
    android:layout_width="match_parent"
    android:textSize="80dp"
    android:textColor="#fff"
    android:gravity="right"
    android:padding="10dp"/>
<TextView
    android:id="@+id/tv_caculator_num"
    android:layout_row="1"
    android:layout_columnSpan="4"
    android:layout_width="match_parent"
    android:text="0"
    android:textSize="80dp"
    android:gravity="right"
    android:padding="10dp"
    android:textColor="#fff"/>
```

图 3-62　添加第一行和第二行显示数字和表达式控件的参考代码及预览效果

步骤 3，在 GridLayout 中添加控件时，先确定子控件所在的行号和列号，然后再确定宽高，添加第三行按钮的参考代码和预览效果如图 3-63 所示。第三行控件的 layout_row 的属性值都为 2，然后通过 layout_column 属性来设置控件所在的列，同样列号也是从 0 开始。

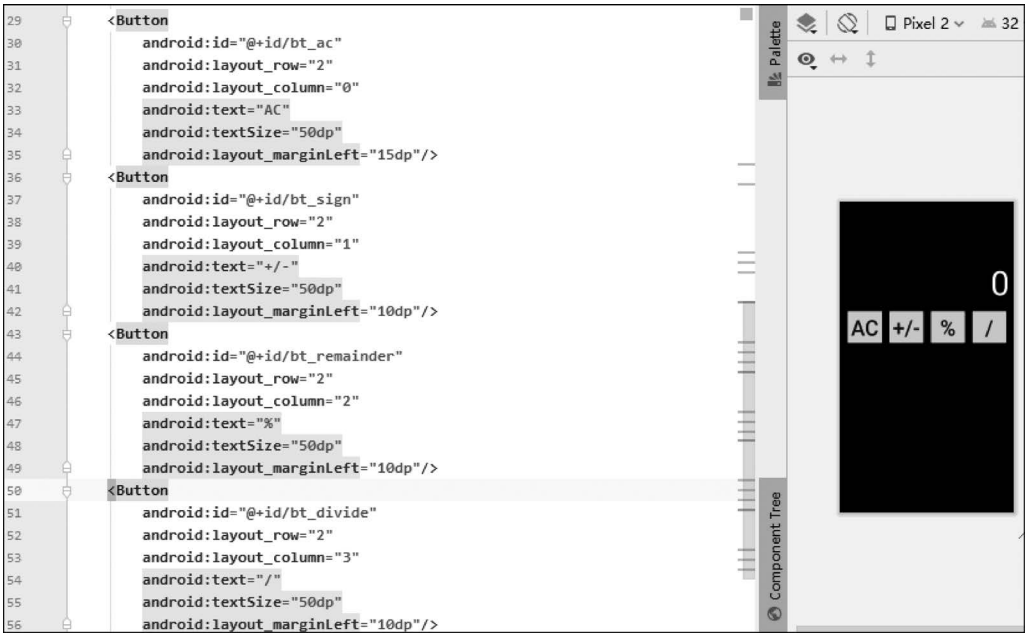
步骤 4，参照第三行按钮的代码，添加第四行按钮控件，其参考代码和预览效果如图 3-64 所示。

步骤 5，根据添加第三行和第四行按钮的步骤，添加第五行按钮控件，其参考代码和预览效果如图 3-65 所示。

步骤 6，以同样的方法添加第六行按钮控件，其参考代码和预览效果如图 3-66 所示。

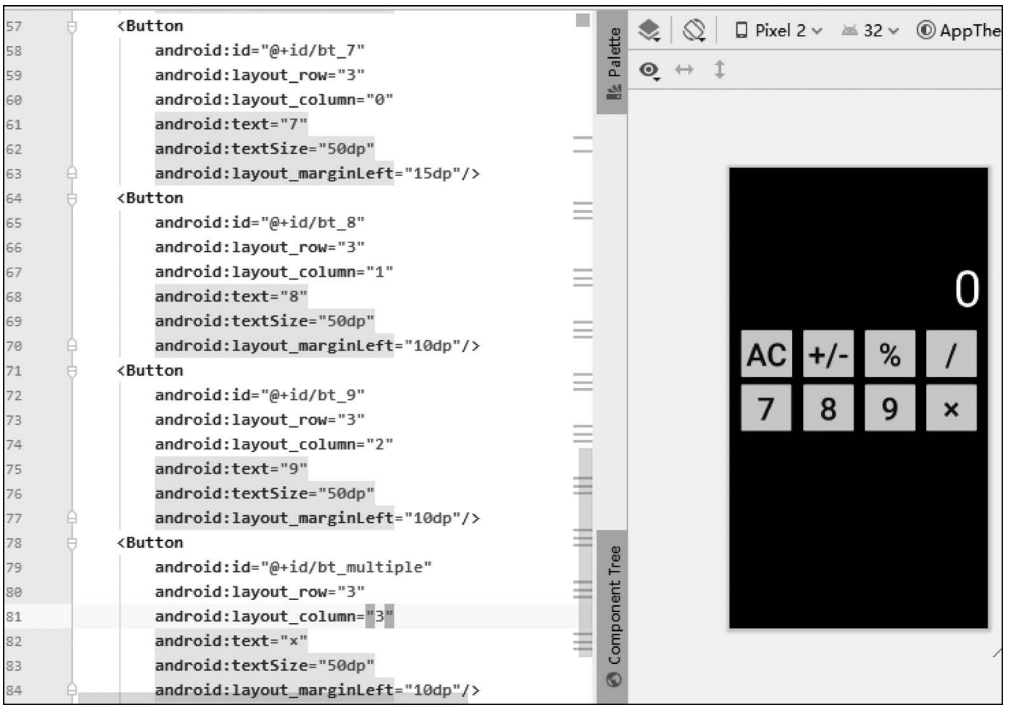
步骤 7，添加第七行按钮控件，其参考代码和预览效果如图 3-67 所示，第七行按钮控件中数字 0 按钮需要设置横跨两列，所以使用了 columnSpan 属性，因为要让按钮的宽度占满两列，所以使用了 layout_gravity="fill_horizontal"填满水平方向。

步骤 8，将本次的布局资源与 WeiXinActivity 进行绑定，其参考代码如图 3-68 所示。




```
<Button
    android:id="@+id/bt_ac"
    android:layout_row="2"
    android:layout_column="0"
    android:text="AC"
    android:textSize="50dp"
    android:layout_marginLeft="15dp"/>
<Button
    android:id="@+id/bt_sign"
    android:layout_row="2"
    android:layout_column="1"
    android:text="+/-"
    android:textSize="50dp"
    android:layout_marginLeft="10dp"/>
<Button
    android:id="@+id/bt_remainder"
    android:layout_row="2"
    android:layout_column="2"
    android:text="%"
    android:textSize="50dp"
    android:layout_marginLeft="10dp"/>
<Button
    android:id="@+id/bt_divide"
    android:layout_row="2"
    android:layout_column="3"
    android:text="/"
    android:textSize="50dp"
    android:layout_marginLeft="10dp"/>
```

图 3-63　添加第三行按钮的参考代码和预览效果

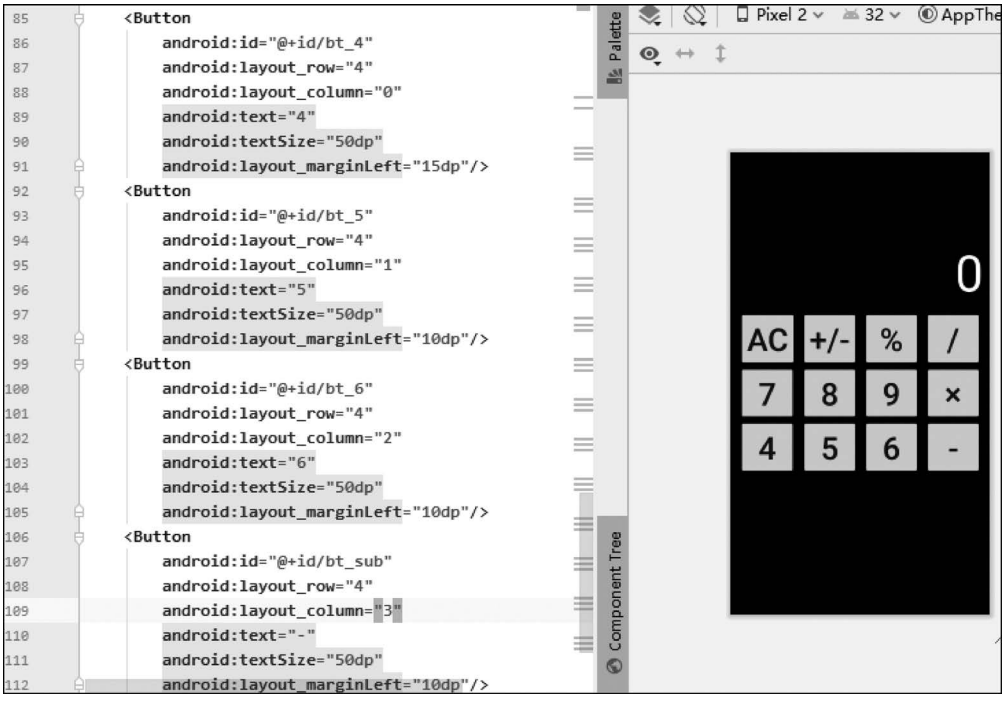



```
<Button
    android:id="@+id/bt_7"
    android:layout_row="3"
    android:layout_column="0"
    android:text="7"
    android:textSize="50dp"
    android:layout_marginLeft="15dp"/>
<Button
    android:id="@+id/bt_8"
    android:layout_row="3"
    android:layout_column="1"
    android:text="8"
    android:textSize="50dp"
    android:layout_marginLeft="10dp"/>
<Button
    android:id="@+id/bt_9"
    android:layout_row="3"
    android:layout_column="2"
    android:text="9"
    android:textSize="50dp"
    android:layout_marginLeft="10dp"/>
<Button
    android:id="@+id/bt_multiple"
    android:layout_row="3"
    android:layout_column="3"
    android:text="x"
    android:textSize="50dp"
    android:layout_marginLeft="10dp"/>
```

图 3-64　添加第四行按钮的参考代码和预览效果

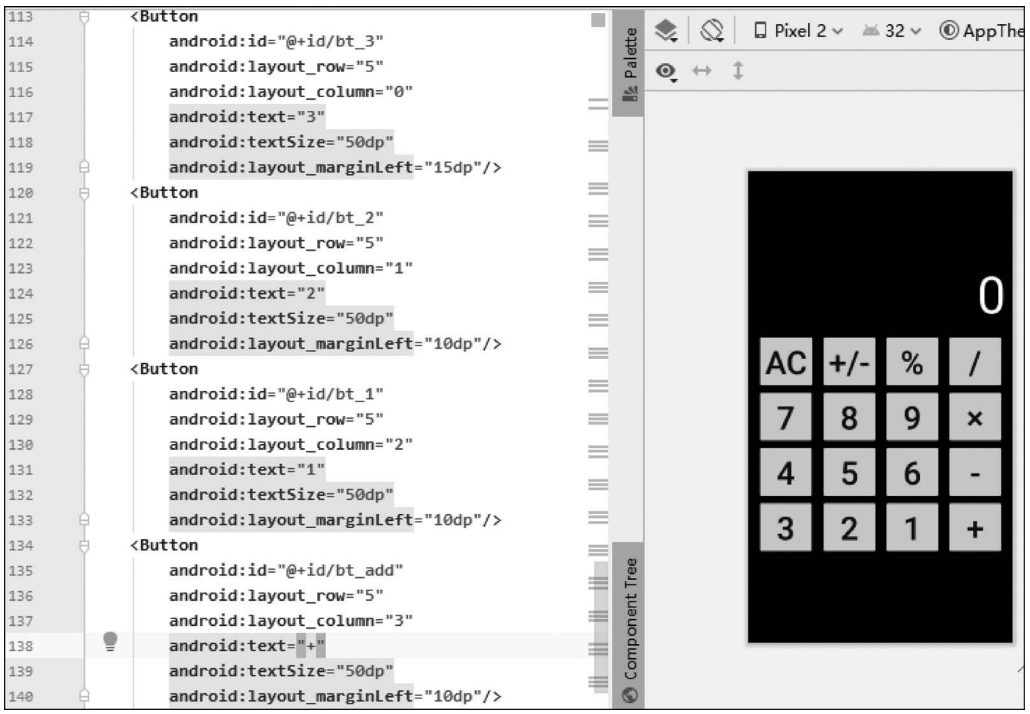
步骤 9，将应用程序运行到手机模拟器中，其效果如图 3-69 所示。

通过对比发现使用表格布局和网格布局可以实现同样的界面效果，但是表格布局的行需要依赖 TableRow 标签才能实现一行放置多个控件，而在网格布局中，每个子控件都可以




```
<Button
    android:id="@+id/bt_4"
    android:layout_row="4"
    android:layout_column="0"
    android:text="4"
    android:textSize="50dp"
    android:layout_marginLeft="15dp"/>
<Button
    android:id="@+id/bt_5"
    android:layout_row="4"
    android:layout_column="1"
    android:text="5"
    android:textSize="50dp"
    android:layout_marginLeft="10dp"/>
<Button
    android:id="@+id/bt_6"
    android:layout_row="4"
    android:layout_column="2"
    android:text="6"
    android:textSize="50dp"
    android:layout_marginLeft="10dp"/>
<Button
    android:id="@+id/bt_sub"
    android:layout_row="4"
    android:layout_column="3"
    android:text="-"
    android:textSize="50dp"
    android:layout_marginLeft="10dp"/>
```

图 3-65　添加第五行按钮的参考代码和预览效果




```
<Button
    android:id="@+id/bt_3"
    android:layout_row="5"
    android:layout_column="0"
    android:text="3"
    android:textSize="50dp"
    android:layout_marginLeft="15dp"/>
<Button
    android:id="@+id/bt_2"
    android:layout_row="5"
    android:layout_column="1"
    android:text="2"
    android:textSize="50dp"
    android:layout_marginLeft="10dp"/>
<Button
    android:id="@+id/bt_1"
    android:layout_row="5"
    android:layout_column="2"
    android:text="1"
    android:textSize="50dp"
    android:layout_marginLeft="10dp"/>
<Button
    android:id="@+id/bt_add"
    android:layout_row="5"
    android:layout_column="3"
    android:text="+"
    android:textSize="50dp"
    android:layout_marginLeft="10dp"/>
```

图 3-66　添加第六行按钮控件的参考代码和预览效果

单独设置它的行和列的位置，用起来比较灵活方便，在两种布局中子控件的宽、高都可以根据需求使用相应的属性进行设置。

图 3-67　添加第七行按钮控件的参考代码及其预览效果

```
public class WeiXinActivity extends Activity {
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.cacularor_grid);
    }
}
```

图 3-68　布局资源与 Activity 进行绑定的参考代码

3.2.5　布局使用总结

经过课堂任务的学习，读者对 Android 应用程序页面的各个组织者都有了一定的了解，它们各有各的特点，不是相互替代，而是相互补充，一般情况下使用线性布局和相对布局就可以实现大部分的页面效果；若要实现类似于键盘和计算器这样的网格型页面效果，可以选择表格布局或者网格布局；若要实现一些不规则的页面效果可以使用约束布局，约束布局其实是相对布局的升级版；帧布局可以与 ViewPager 一起实现主页面效果；绝对布局在一些游戏场景上可能会用到。各位读者接下来可以根据自己项目的需求选择恰当的布局实现相应的界面效果。

图 3-69　运行到模拟器的效果

3.3　课后学习任务：制作一个注册页面

学习完 Android 应用程序页面的组织者，现在通过设计和实现一个注册页面来实践所学的知识，并完成配套资源中的相应实验报告。目前注册页面一般需要用户使用手机号码进行注册，符合国家推行的网络实名制政策，在一定程度上遏制了网络犯罪，图 3-70 是微信、支付宝和京东 App 的注册页面的效果图。

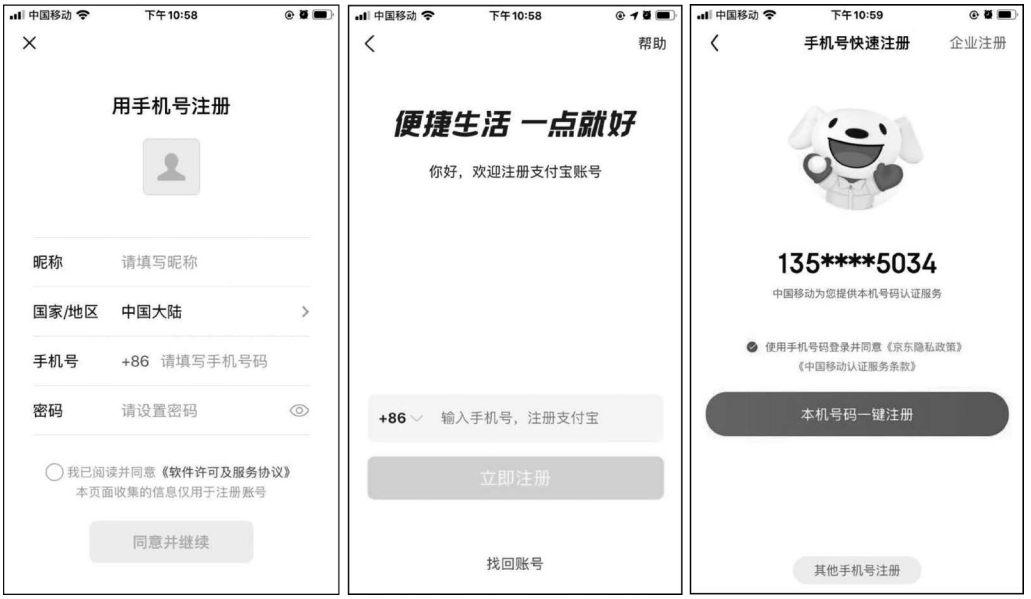


图 3-70 常见 App 的注册页面效果

3.3.1 页面分析

读者可以从注册页面所包含的控件和功能等方面分析页面的功能需求，总结出注册页面所需达到的功能效果，并写出设计应用程序注册页面所需功能的业务流程。

3.3.2 页面设计

各位读者可以根据页面分析的结果，设计出符合自己应用程序风格和功能的注册页面原型图。

3.3.3 页面实现

根据注册页面的设计图，使用相关资源，在应用程序中实现注册页面，并完成配套资料的页面实验报告。