第 5 章

UI 组件与布局

第 4 章介绍了 Activity，以及 Android 应用程序中与用户交互的基本单元。一个 Activity 中通常包含各种 UI 控件，以完成不同的功能。这些控件按照一定的位置分布在 Activity 的不同区域，这称为布局（layout）。第 2 章和第 4 章已经使用了 Button、TextView 等控件，以及最简单的布局——LinearLayout（线性布局）。本章将介绍 Android 应用程序开发中常用的 UI 组件和布局。

5.1 常用控件

Android 中有多种编写程序界面的方式可供选择。Android Studio 和 Eclipse 中都提供了可视化的界面编辑器。可视化编辑器允许使用拖曳控件的方式对布局进行编辑，并能在视图上直接修改控件的属性，但是这种方式并不利于真正了解界面背后的原理和技术。本节在讲解 UI 组件和布局时，大多数都采用编写 XML 代码的方式。

下面从 Android 系统中几种最常见的控件开始熟悉 UI 组件。

Android 提供了大量的 UI 控件，合理使用这些控件能轻松地编写出美观的界面。本节将选择几种最常用的控件，详细介绍其使用方法。

Android 控件继承结构如图 5.1 所示。

从图 5.1 可以看出，View 是 Android 所有控件的基类，同时 ViewGroup 也继承自 View。知道 View 的层级关系有助于理解 View。从图 5.1 可以发现常用的控件都继承自 View，如果掌握了 View 的知识体系，那么在界面编程时会更加得心应手。

5.1.1 View 类

可以看到，所有的 UI 控件（主要在包 android.view 和包 android.widget 中）都是 View 的子类。使用较早的 Android 版本进行应用程序开发时，每当用 findViewById(R.id.xx)方法时，总要将其返回类型进行强制类型转换，因为该方法返回的是一个 View 实例。其中不得不提 View 的子类——ViewGroup。Android 系统中的所有 UI 类都建立在 View 和 ViewGroup 类的基础上。所有 View 的子类都称为 Widget（小部件），所有 ViewGroup 的子类都称为 Layout(布局)。View 和 ViewGroup 之间采用组合设计模式，可以使得“部分-整体”同等对待。ViewGroup 作为布局容器类的最上层，布局容器里又可以有 View 和 ViewGroup。通过这种方式，获得了 UI 的组合方式。

ViewGroup 的子类用不同的方式管理容器中 View 控件的摆放位置以及显示方式；

但是,对于UI控件具体摆放到什么位置,以及大小等属性,则需要每个布局类的内部类LayoutParams进行处理,该类是ViewGroup的内部类。LayoutParams类有多个子类实现,用于指定不同的布局参数。

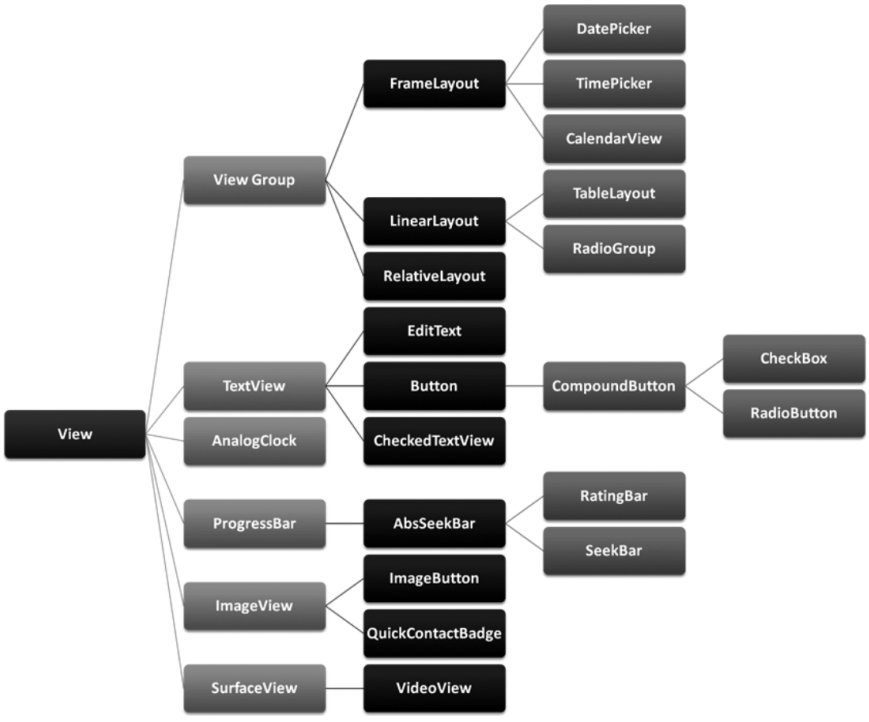

图 5.1 Android 控件继承结构

可以看到,Android中所有的UI控件都是View的子类,所以可以通过继承View类实现自定义控件。注意,此时需要重载View的构造函数。View的构造方法有下面4个,比较常用的是第一个和第二个。

- public View(Context context);
- public View(Context context, AttributeSet attrs);
- public View(Context context, AttributeSet attrs, int defStyleAttr);
- public View(Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes);

第一个构造方法的参数context代表该View对象所属的Context对象。第二个构造方法的参数attrs代表在布局文件中该View对象对应的元素相关的属性值的集合。

使用UI控件时,一般采用下面的步骤。

(1) 在布局文件中添加该UI控件的元素(元素的名称与该UI控件的类名完全相同),并定义该元素的一些属性的值(其中id属性非常重要,下一步将会使用id属性的值引用该元素)。

(2) 在其他资源文件中引用该UI控件;在源代码中定义一个该UI控件类型的变量(成员变量、局部变量均可),使用findViewById()方法使该变量引用步骤(1)中添加的UI控件。

(3) 定义处理 UI 控件某种事件的监听器,并调用步骤(2)中变量的相关方法,把监听器与 UI 控件绑定,之后如果用户触发了该种 UI 事件,那么 Android 系统将会回调已经绑定的监听器中的事件处理方法(例如,第 4 章中 Button 控件事件监听器的 onClick()方法,该方法由 View.OnClickListener 接口定义)。

上述步骤将会在后续各种控件的介绍中多次看到。

5.1.2 TextView

为展示 Android 系统中常见控件的使用方法,可在 Android Studio 中新建一个项目 UIWidgetDemo,使用自动创建的 MainActivity 以及布局文件 activity_main.xml。

第 2 章已经使用 TextView 控件显示了 Hello World! 的欢迎信息。

修改项目 UIWidgetDemo 中 activity_main.xml 的代码,具体如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/tvHello"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="厉害了,我的国!" />
</LinearLayout>
```

观察上述布局文件中的<TextView>标签,使用 android:id 属性给当前控件定义了一个唯一的标识符,然后使用 android:layout_width 和 android:layout_height 属性指定了控件的宽度和高度。Android 中所有的控件都具有这两个属性,可选值有 3 种:match_parent、fill_parent 和 wrap_content,其中 match_parent 和 fill_parent 的意义相同,现在更推荐使用 match_parent。match_parent 表示让当前控件的大小和父布局的大小一样,也就是由父布局决定当前控件的大小。wrap_content 表示让当前控件的大小能够刚好包含控件的内容,也就是由控件的内容决定当前控件的大小。根据上述规则,上述布局文件的代码表示让 TextView 的宽度和父布局的宽度(也就是手机屏幕的宽度)一样,而 TextView 的高度足够包含其中的内容就可以了。当然,除了上述选项外,也可以对控件的宽度和高度指定固定的值,但是这样做有时会在多种手机屏幕(大小和分辨率不同)的适配方面出现问题。

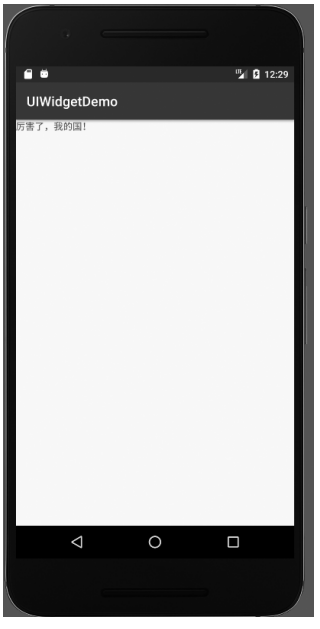

图 5.2 TextView 控件运行结果

下面通过 android:text 属性指定了 TextView 中显示的文本内容,运行该程序,运行结果如图 5.2 所示。

虽然指定的文本内容正常显示了，但是好像并没有看出 TextView 控件的宽度和手机屏幕宽度相同，其实这是由于 TextView 中的文本默认居左显示，虽然 TextView 控件的宽度充满了整个屏幕，但是由于文本内容不够长，因此从效果上看不出 TextView 控件的宽度。修改布局文件中 TextView 控件的对齐方式，代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/tvHello"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="厉害了，我的国！" />
</LinearLayout>
```

使用 android:gravity 属性指定文字的对齐方式，可选值有 top、bottom、left、right、center 等，也可以使用|同时指定多个值。这里指定的 center，效果上等价于 center_vertical | center_horizontal，表示文字在垂直和水平方向都居中对齐。重新运行该程序，运行结果如图 5.3 所示。

这也说明了 TextView 的宽度的确和屏幕宽度一样。

另外，还可以对 TextView 中文字的大小和颜色进行修改，代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/tvHello"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textSize="36sp"
        android:textColor="#ff0000"
        android:text="厉害了，我的国！" />
</LinearLayout>
```

通过 android:textSize 属性可以指定文字的大小，通过 android:textColor 属性可以指定文字的颜色。在 Android 中，字体的大小使用 sp(scaled pixels)作为单位。重新运行程序，运行结果如图 5.4 所示。

除上述属性之外，TextView 还有很多其他的属性，此处不再一一列举。

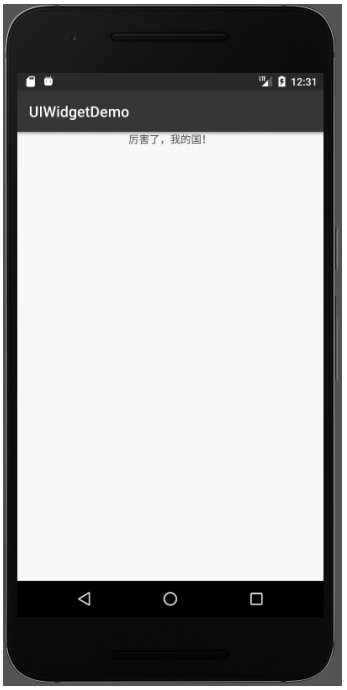

图 5.3 TextView 控件运行结果(居中对齐)

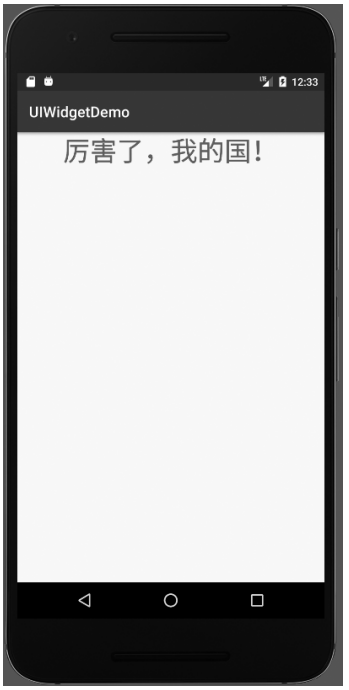

图 5.4 TextView 控件运行结果(修改字体大小)

5.1.3 Button

Button 是应用程序中和用户进行交互的一种常用的控件。第 4 章中多次使用到 Button。从图 5.1 可以看出,Button 类的直接父类是 TextView 类,因此 5.1.2 节中 TextView 的属性 Button 都会自动继承得到。在布局文件 activity_main.xml 中添加一个 Button,代码如下:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:id="@+id/buttonTest"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="TestButton" />
</LinearLayout>
```

加入 Button 之后的界面如图 5.5 所示。

仔细观察会发现,在布局文件中设置的文本值是 TestButton,但最终的显示结果却是 TESTBUTTON,这是因为 Android 系统会将 Button 中的所有英文字母自动显示为

图 5.5 加入 Button 之后的界面

大写状态。如果仍然想显示包含大小写的原有形式，可以设置 android:textAllCaps 属性值为 false，代码如下：

```
<Button
    android:id="@+id/buttonTest"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="TestButton"
    android:textAllCaps="false" />
```

接下来为该 Button 控件的单击事件绑定一个监听器，代码如下：

```
public class MainActivity extends AppCompatActivity {
    Button button;
    TextView textView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        this.button=findViewById(R.id.buttonTest);
        this.textView=findViewById(R.id.tvHello);
        //创建监听器类的一个对象
        ButtonListener btnListener=new ButtonListener();
```

```
        //把按钮控件与监听器对象绑定,也称为注册
        this.button.setOnClickListener(btnListener);
    }
    private class ButtonListener implements View.OnClickListener {
        @Override
        public void onClick(View v) {
            MainActivity.this.textView.setText("中国梦!");
        }
    }
}
```

这样,每当用户单击该按钮时,Android 系统会执行监听器类中定义的 onClick()方法,只在该方法中编写相关的业务处理逻辑即可。对于初学者来说,上述编程方式比较清晰明了,但是代码编写比较繁杂,其实,创建的监听器类的对象 btnListener 只在绑定监听器时使用,因此使用监听器类的无名对象即可。上述创建对象和绑定监听器的两行代码可以用下面的一行代码替换。

```
this.button.setOnClickListener(new ButtonListener());
```

这种方法调用中的参数,一般称为创建了 ButtonListener 类的一个匿名对象(也称无名对象)。更进一步,ButtonListener 类的类名也只在上述绑定事件监听器中使用一次,因此也可以使用实现了 View.OnClickListener 接口的一个匿名类的一个匿名对象作为 setOnClickListener()方法的参数,这样就省略了定义类 ButtonListener,而只是实现 View. OnClickListener 接口中的 onClick()方法,在 Mainactivity 类的 onCreate()方法的最后添加如下代码:

```
this.button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        MainActivity.this.textView.setText("中国梦!");
    }
});
```

添加了 Button 按钮的程序,单击该按钮后 TextView 控件中的内容会改为“中国梦!”,运行结果如图 5.6 所示。

可以看出,在调用方法时使用实现了 View. OnClickListener 接口的匿名类的匿名对象作为参数,代码非常简洁,只是写法看上去不是很直观,理解起来稍有难度。这样,如果在 MainActivity 中有多个控件,就不需要为每个控件定义一个事件处理的监听器类了,因为那种方式会导致 MainActivity 的代码结构过于繁杂,不利于代码的编写和维护。这种方法调用时使用实现了某接口的匿名类的匿名对象的方式,在第 6 章的多线程程序设计中还会多次出现,不同的是,接口变成了 Runnable,需要实现的方法变成了 public void run()。

另外,还有一种实现上述功能的方式,那就是在定义类 MainActivity 的时候,除了从

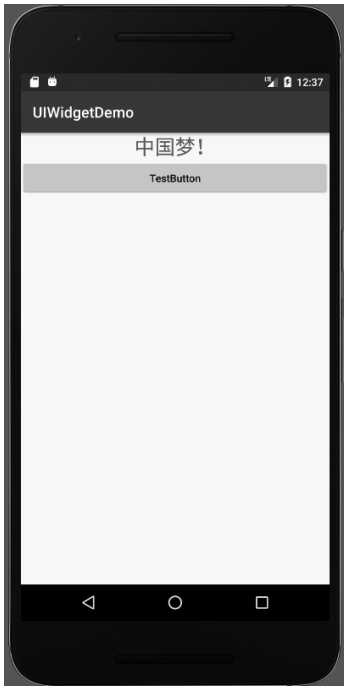

图 5.6 单击 Button 后的运行结果

AppCompatActivity 继承外,还实现 View.OnClickListener 接口,并在 MainActivity 类中实现 onClick()方法,代码如下:

```
public class MainActivity extends AppCompatActivity
                    implements View.OnClickListener {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        this.button=findViewById(R.id.buttonTest);
        this.textView=findViewById(R.id.tvHello);
        this.button.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        switch(v.getId()) {
            case R.id.buttonTest:
                this.textView.setText("中国梦!");
                break;
            default:
                break;
    }
}
```

由于是 MainActivity 类本身实现了 View.OnClickListener 接口，实现了 onClick() 方法，因此在绑定事件监听器的时候，方法调用的参数使用 this 即可。如果在 MainActivity 类有多个控件，这些控件都使用 this 对象作为事件的监听器，那么在实现 onClick()方法时，需要判断用户单击事件发生在哪个控件上，代码中使用 v.getId()获得发生单击事件的控件对象的 id 号，之后使用 switch-case 语句进行处理。由于这种方法会改变 MainActivity 类的签名，因此不推荐使用。

另外，无论是 TextView，还是 Button 控件，都可以使用 setText()方法设置其中的文字内容，该方法的定义如下：

- public final void setText(CharSequence text);
- public final void setText(char[] text, int start, int len);
- publicfinal void setText(int resid);
- public final void setText(int resid, BufferType type);

第一个方法的参数类型是 CharSequence 接口，String、StringBuffer、StringBuilder 等类都实现了该接口，因此实际参数使用上述类的对象均可。第三个方法的参数是指定的字符串资源的 id，如 R.string.app_name，而不是需要显示的整数的内容。如果需要在控件中显示数值类型的内容，须转换为 String 类型（最简单的办法是调用 toString()方法）后调用第一个方法，而不是直接调用第三个方法。

5.1.4 EditText

EditText 是应用程序用来和用户进行交互中使用非常广泛的一种控件，它和 TextView 的区别在于，EditText 允许用户在控件里输入和编辑内容，同时可以在程序中对这些内容进行处理。EditText 控件的使用场景非常广泛，在进行用户注册、用户登录、搜索、发送短信、发送微信消息、发微博、聊 QQ 等操作中，都会使用 EditText 控件。从图 5.1 中可以看出，EditText 和 Button 类一样，都继承自 TextView 类，因此也会继承得到 TextView 类的相关成员变量和方法，例如重载的多个 setText()方法。

在 activity_main.xml 布局文件中添加一个 EditText 控件，代码如下：

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

通过上述代码可以看出使用 UI 控件的一般步骤：首先通过 android:id 属性给控件定义一个唯一的 id，然后指定该控件的宽度和高度，最后再添加一些该控件特有的属性。

运行程序，运行结果如图 5.7 所示。

(a) 普通EditText控件　　(b) 带有提示信息的EditText控件　　(c) 输入内容

图 5.7　EditText 运行结果

如果具有比较丰富的使用 Android 手机的经验，会发现一些人性化的软件会在输入框中显示一些提示性的文字，然后当用户输入任何内容之后，原有的提示性的文字就会消失，这种提示功能在 Android 中很容易实现，通过 android:hint 属性(hint 的意思是提示、注意事项、暗示等)设置即可。

设置 android:hint 属性的代码如下：

```
<EditText
  android:id="@+id/etStatement"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:hint="请输入您的心情" />
```

重新运行程序，界面如图 5.7(b) 所示，输入信息后如图 5.7(c) 所示。可以看到，EditText 控件中显示了一段提示信息，当输入内容时，这段文本就会自动消失。

但是，随着输入内容的不断增加，EditText 会被不断拉长，这是由于 EditText 的高度指定的值是 wrap_content，因此该控件总能包含用户输入的文本内容，但是当输入的文本内容过多时，可能会破坏整个 Activity 的布局，导致这个界面很难看。为了解决这个问题，可以使用 android:maxLines 属性修改布局文件，如下所示。

```
<EditText
    android:id="@+id/etStatement"
    android:layout_width="match_parent"
```