第3章

Android基础界面编程

Android 程序开发主要分为三部分：界面设计、代码流程控制和资源建设。代码和资源主要是由开发者进行编写和维护的，对于大部分用户来说是不关心的，展现在用户面前最直观的就是界面设计。作为一个程序设计者，必须首先考虑用户的体验，只有用户满意了开发的产品，应用才能推广，才有价值，因此界面设计尤为重要。

控件是 Android 项目开发的基本组成单位，通过使用组件可以高效地开发 Android 程序，所以熟练掌握控件的使用是进行 Android 程序开发的重要前提。本章对“购物商城”基本组件进行分析，认识常用基础控件和布局管理器的使用方法。通过对基础控件和布局管理器的综合运用实现“欢乐购商城”登录、注册、首页表格等页面。

学习目标

本章要点

(1) 掌握文本显示框的功能和用法。
(2) 熟悉文本编辑框的常用属性。
(3) 掌握按钮的简单用法。
(4) 掌握图片的使用方法。
(5) 掌握线性布局的功能和用法。
(6) 熟悉表格布局的功能和用法。
(7) 掌握相对布局的功能和用法。
(8) 掌握网格布局的功能和用法。
(9) 掌握布局的嵌套使用。

视频详解

3.1 基础 View 组件简介

任务陈述

“欢乐购商城”项目中每个页面中都运用了 Android 基础界面控件，本节运用基础界面控件实现“欢乐购商城”登录页面、注册页面。

分析：登录页面整体采用垂直方向线性布局，从上往下摆放图片(ImageView)、文本编辑框(EditText)、按钮(Button)、文本显示框(TextView)基础控件，如图 3-1 所示。

注册页面整体采用垂直方向线性布局，从上往下摆放图片(ImageView)、文本编辑框(EditText)、按钮(Button)，如图 3-2 所示。

图 3-1　登录页面

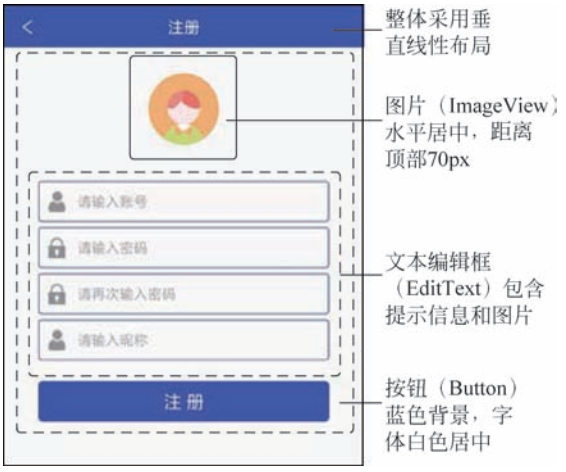

图 3-2　注册页面

相关知识

Android 中所有的组件都继承于 View 类，View 类代表的就是屏幕上的一块空白的矩形区域，该空白区域可用于绘画和事件处理。不同的界面组件，相当于对这个矩形区域做了一些处理，如文本显示框、按钮等。

View 类有一个重要的子类：ViewGroup。ViewGroup 类是所有布局类和容器组件的基类，它是一个不可见的容器，它里面还可以添加 View 组件或 ViewGroup 组件，主要用于定义它所包含的组件的排列方式，例如，网格排列或线性排列等。通过 View 和 ViewGroup

的组合使用，从而使得整个界面呈现一种层次结构。ViewGroup 内包含的组件如图 3-3 所示。

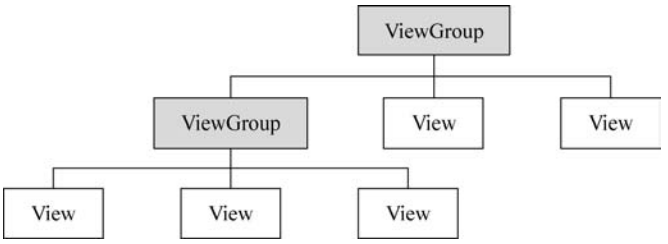

图 3-3　ViewGroup 组件的层次结构

Android 中控制组件的显示有两种方式：一种是通过 XML 布局文件来设置组件的属性进行控制；另一种是通过 Java 代码调用相应的方法进行控制。这两种方式控制 Android 界面显示的效果是完全一样的。实际上，XML 文件的属性与 Java 代码中方法之间存在着一一对应的关系。从 Android API 文档中 View 类的介绍中，可查看所有的属性与方法之间的对应关系，在此只列出一些常用的属性供参考，如表 3-1 所示。

表 3-1　View 类的常见 XML 属性、对应方法及说明

XML 属 性	对 应 方 法	说　明
android：alpha	setAlpha(float)	设置组件的透明度
android：background	setBackgroundResource(int)	设置组件的背景
android：clickable	setClickable(boolean)	设置组件是否可以触发单击事件
android：focusable	setFocusable(boolean)	设置组件是否可以得到焦点
android：id	setId(int)	设置组件的唯一 ID
android：minHeight	setMinimumHeight(int)	设置组件的最小高度
android：minWidth	setMinimumWidth(int)	设置组件的最小宽度
android：padding	setPadding(int,int,int,int)	在组件四边设置边距
android：scaleX	setScaleX(float)	设置组件在 X 轴方向的缩放
android：visibility	setVisibility(int)	设置组件是否可见

几乎每个界面组件都需要设置 android：layout_height、android：layout_width 这两个属性，用于指定该组件的高度和宽度，主要有以下三种取值。

（1）fill_parent：表示组件的高或宽与其父容器的高或宽相同。

（2）wrap_content：表示组件的高或宽恰好能包裹内容，随着内容的变化而变化。

（3）match_parent：该属性值与 fill_parent 完全相同，Android 2.2 之后推荐使用 match_parent 代替 fill_parent。

虽然两种方式都可以控制界面的显示，但是它们又各有优缺点。

（1）完全使用 Java 代码来控制用户界面不仅烦琐，而且界面和代码相混合，不利于解耦、分工。

（2）完全使用 XML 布局文件虽然方便、便捷，但灵活性不好，不能动态改变属性值。

因此，我们经常会混合使用这两种方式来控制界面，一般来说，习惯将一些变化小的、比较固定的、初始化的属性放在 XML 文件中管理，而对于那些需要动态变化的属性则交给 Java 代码控制。例如，可以在 XML 布局文件中设置文本显示框的高度和宽度以及初始时

的显示文字，在代码中根据实际需要动态地改变显示的文字。

3.1.1　文本显示框 TextView


视频详解

TextView 类直接继承于 View 类，主要用于在界面上显示文本信息，类似于一个文本显示器，从这个方面来理解，有点儿类似于 Java 编程中的 JLable 的用法，但是比 JLable 的功能更加强大，使用更加方便。TextView 可以设置显示文本的字体大小、颜色、风格等属性，TextView 的常见属性如表 3-2 所示。

表 3-2　TextView 类的常见 XML 属性、对应方法及说明

XML 属性	对应方法	说　明
android：gravity	setGravity(int)	设置文本的对齐方式
android：height	setHeight(int)	设置文本框的高度(以 pixel 为单位)
android：text	setText(CharSequence)	设置文本的内容
android：textColor	setTextColor(int)	设置文本的颜色
android：textSize	setTextSize(int,float)	设置文本的大小
android：textStyle	setTypeface(Typeface)	设置文本的风格
android：typeface	setTypeface(Typeface)	设置文本的字体
android：width	setWidth(int)	设置文本框的宽度(以 pixel 为单位)
Android：drawableLeft	setCompoundDrawablesWith IntrinsicBounds(int,int,int,int)	要绘制在文本左侧的可绘制对象

这些是文本显示控件都具有的功能。除此之外，Android 中的 TextView 还具有自动识别文本中的各种链接、能够显示字符串中的 HTML 标签的格式。识别自动链接的属性为 android：autoLink，该属性的值有以下几种。

(1) none：不匹配任何格式，这是默认值。

(2) web：只匹配网页，如果文本中有网页，网页会以超链接的形式显示。

(3) email：只匹配电子邮箱，电子邮箱会以超链接的形式显示。

(4) phone：只匹配电话号码，电话号码会以超链接的形式显示。

(5) map：只匹配地图地址。

(6) all：匹配以上所有。

当匹配时，相应部分会以超链接显示，单击超链接，会自动运行相关程序。例如，单击电话号码超链接会调用拨号程序，单击网页超链接会打开网页等。

而显示 HTML 标签格式，则需要通过 Java 代码来控制。首先为该文本框添加一个 id 属性，然后在 onCreate()方法中，通过 findViewById(R.id.***)获取该文本框，最后通过 setText()方法来设置显示的内容。

【例 3-1】 使用 TextView 显示“欢迎使用《Android 零基础入门到实战》”文本，其中，“《Android 项目开发实战教程》”为蓝色字体。

注意：本书代码中，右箭头所指内容为编者对读者的提示，实际编译时无须添加。

程序清单 3-1：chart0301\app\src\main\res\layout\activity_main.xml

```
1    <TextView
2        android:id="@+id/myText"                    →为 TextView 添加 id 属性
```

```
3        android:textSize = "18sp"                    →字体大小设置为 18sp
4        android:layout_width = "wrap_content"        →组件宽度为内容包裹
5        android:layout_height = "wrap_content"       →组件高度为内容包裹
6        />
```

程序清单 3-2：chart0301\app\src\main\java\com\jxcia\chart0301\MainActivity. java

```
1    TextView tv = (TextView) findViewById(R. id. myText); //myText 为 id
2    tv. setText(Html. fromHtml("欢迎使用< font color = blue >《Android 项目开发实战教程》
     </font >"));
```

该代码的显示效果是：《Android 零基础入门到实战》这几个字为蓝色，其他字的颜色为布局文件中设置的颜色，如图 3-4 所示。

欢迎使用《Android零基础入门到实战》

图 3-4　TextView 实现效果

视频详解

3.1.2　文本编辑框 EditText

TextView 的功能仅是用于显示信息而不能编辑，好的应用程序往往需要与用户进行交互，让用户进行输入信息。为此，Android 中提供了 EditText 组件，EditText 是 TextView 类的子类，与 TextView 具有很多相似之处。它们最大的区别在于，EditText 允许用户编辑文本内容。使用 EditText 时，经常使用到的属性有以下几个。

（1）android：hint：设置当文本框内容为空时，文本框内显示的提示信息，一旦输入内容，该提示信息立即消失，当删除所有输入的内容时，提示信息又会出现。

（2）android：password：设置文本框是否为密码框，值为 true 或者 false，设置为 true 时，输入的内容将会以点替代，但已不推荐使用了。

（3）android：inputType：设置文本框接收值的类型，例如，只能是数字、电话号码等。

【例 3-2】 以“欢乐购商城”登录页面输入手机号编辑框为例，介绍文本编辑框使用方法，界面程序代码如下。

程序清单 3-3：char0302\app\src\main\res\layout\activity_main. xml

```
1    < EditText
2        android:layout_width = "match_parent"              →组件宽度为填充父容器
3        android:layout_height = "40dp"                     →组件高度为 40dp
4    android:drawableLeft = "@drawable/user_name_icon"      →编辑框左侧设置图片
5        android:drawablePadding = "10dp"                   →图片内边距为 10dp
6        android:gravity = "center_vertical"                →编辑框内容垂直居中
7        android:hint = "请输入手机号"                       →提示内容
8        android:paddingLeft = "8dp"                        →编辑左侧内编辑为 8dp
9        android:singleLine = "true"                        →单行输入
```

```
10      android:textColor = "#000000"                →字体颜色为白色
11      android:textColorHint = "#a3a3a3"            →提示字体颜色为灰色
12      android:textSize = "14sp" />                 →字体大小设置为 14sp
```

程序运行效果如图 3-5 所示。

图 3-5 EditText 运行效果

3.1.3 按钮 Button


视频详解

Button 也是继承于 TextView，功能非常单一，就是在界面中生成一个按钮，供用户单击。单击按钮后，会触发一个单击事件，开发人员针对该单击事件可以设计相应的事件处理；从而实现与用户交互的功能。用户可以设置按钮的大小、显示文字以及背景等。当我们想把一张图片作为按钮时，有两种方法：一种是将该图片作为 Button 的背景图片；另一种是使用 ImageButton 按钮，将该图片作为 ImageButton 的 android：src 属性值即可。需注意的是，ImageButton 按钮不能指定 android：text 属性，即使指定了，也不会显示任何文字。

【例 3-3】 以“欢乐购商城”登录页面“登录”按钮为例，介绍按钮使用方法，界面布局代码如下。

程序清单 3-4：char0303\app\src\main\res\layout\activity_main.xml

```
1   <Button
2       android:layout_width = "fill_parent"              →组件宽度为填充父容器
3       android:layout_height = "40dp"                    →组件高度为 40dp
4       android:layout_gravity = "center_horizontal"      →组件对外水平居中
5       android:layout_marginLeft = "35dp"                →组件左边外边距 35dp
6       android:layout_marginRight = "35dp"               →组件右边外边距 35dp
7       android:layout_marginTop = "15dp"                 →组件顶部外边距 15dp
8       android:background = "#6200EE"                    →背景为蓝紫色
9       android:text = "登 录"                            →组件文字
10      android:textColor = "#FFFFFF"                     →字体颜色为白色
11      android:textSize = "18sp" />                      →字体大小 18sp
```

程序运行效果如图 3-6 所示。

图 3-6 按钮界面运行效果

3.1.4 图片视图 ImageView


视频详解

ImageView（图片视图）的作用与 TextView 类似，TextView 用于显示文字，ImageView 则用于显示图片，既然是显示图片，那就要设置图片的来源，ImageView 中有一个 src 属性用于指定图片的来源。显示图片还存在另外一个问题，就是当图片比 ImageView 的区域大的时候如何显示呢？在 ImageView 中有一个常用并且重要的属性 scaleType，用于设置图片的缩放类型。该属性值主要包含以下几个。

fitCenter：保持纵横比缩放图片，直到该图片能完全显示在 ImageView 中，缩放完成后将该图片放在 ImageView 的中央。

fitXY：对图片横向、纵向独立缩放，使得该图片完全适应于该 ImageView，图片的纵横比可能会改变。

centerCrop：保持纵横比缩放图片，以使得图片能完全覆盖 ImageView。

【例 3-4】 以“欢乐购商城”登录页面登录按钮为例，介绍按钮使用方法，界面布局代码如下。

程序清单 3-5：char0304\res\layout\activity_main.xml

```
1   <ImageView
2       android:layout_width="70dp"                    →组件宽度为 70dp
3       android:layout_height="70dp"                   →组件高度为 70dp
4       android:layout_gravity="center_horizontal"     →组件对外水平居中
5       android:layout_marginTop="70dp"                →组件顶部外边距 70dp
6       android:scaleType="fitCenter"                  →图片缩放类型保持纵横比
7       android:src="@drawable/head" />                →引用图片地址
```

程序运行效果如图 3-7 所示。

图 3-7 图片控件运行效果

视频详解

3.1.5 实战演练——登录页面

通过前面对基础界面控件 TextView、EditText、Button 和 ImageView 的介绍，可以初步实现“欢乐购商城”的登录页面。如图 3-1 所示，在“欢乐购商城”登录页面中整体采用垂直方向线性布局，从上往下摆放图片（ImageView）、文本编辑框（EditText）、按钮（Button）、文本显示框（TextView）基础控件，以下介绍登录界面具体实现过程。

在布局文件中多次用到@string/ *** 作为 android：text 的属性值，表示引用 R.java 中 string 内部类的成员变量所代表的资源。这些常量值是在 strings.xml 文件中定义的。查看 strings.xml 文件的内容如下。

程序清单 3-6：chart0305\app\src\main\res\values\strings.xml

```
1   <resources>
2       <string name="app_name">chart0205</string>
3       <string name="inputname">请输入手机号</string>
4       <string name="inputpassword">请输入密码</string>
5       <string name="btnLogin">登 录</string>
6       <string name="registerNow">立即注册</string>
7       <string name="findPassword">找回密码?</string>
8   </resources>
```

在布局文件中多次用到@color/ *** 作为 android：textColor 的属性值，表示引用 R.java 中 Color 内部类的成员变量所代表的资源。这些常量值是在 colors.xml 文件中定义的。查看 colors.xml 文件的内容如下。

程序清单 3-7：chart0305\app\src\main\res\values\colors.xml

```
<resources>
    <color name="colorPrimary">#6200EE</color>
    <color name="colorPrimaryDark">#3700B3</color>
    <color name="colorAccent">#03DAC5</color>
    <color name="white">#FFFFFF</color>
</resources>
```

其实在设置 android：text 和 android：textColor 属性时，可以直接将这些字符串常量赋值给该属性，但是建议不要这么做。因为一些字符串常量可能会在多处被使用，如果都在属性里写，不仅占用更多的内存，而且修改起来也比较麻烦，需要一个个进行修改；另一方面，统一放在 strings.xml 和 colors.xml 文件中，日后如果要修改，只需要修改相应的资源文件就可以了，而不用去更改别的文件，可扩展性比较好。

登录页面主布局实现，代码如下。

程序清单 3-8：char0305\app\src\main\res\layout\activity_main.xml

```
<LinearLayout 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:background="@color/white"                    →背景引用颜色文件中白色
    android:orientation="vertical"                       →线性布局方向为垂直
    tools:context=".MainActivity">
    <ImageView
        android:layout_width="70dp"
        android:layout_height="70dp"
        android:layout_gravity="center_horizontal"       →图片控件对外水平居中
        android:layout_marginTop="70dp"                  →图片控件顶部外边距 70dp
 android:background="@drawable/defaultuser_icon" />
     <EditText
        android:layout_width="fill_parent"
        android:layout_height="40dp"
        android:layout_gravity="center_horizontal"       →控件对外水平居中
        android:layout_marginLeft="35dp"
        android:layout_marginRight="35dp"
        android:layout_marginTop="35dp"
 android:drawableLeft="@drawable/user_name_icon"         →引用图片地址
        android:drawablePadding="10dp"                   →图片内边距 10dp
        android:gravity="center_vertical"                →控件内容垂直居中
        android:hint="@string/inputname"                 →文字引用 string.xml 中文字地址
        android:paddingLeft="8dp"
        android:singleLine="true"                        →单行输入
        android:textColor="@color/white"
        android:textColorHint="#a3a3a3"                  →提示文字颜色为灰色
        android:textSize="14sp" />                       →字体大小为 14sp
```

```
31          < EditText
32              android:layout_width = "fill_parent"
33              android:layout_height = "40dp"
34              android:layout_gravity = "center_horizontal"
35              android:layout_marginLeft = "35dp"
36              android:layout_marginRight = "35dp"
37              android:layout_marginTop = "5dp"
38              android:drawableLeft = "@drawable/psw_icon"
39              android:drawablePadding = "10dp"
40              android:hint = "@string/inputpassword"   →文字引用 string.xml 中文字地址
41              android:inputType = "textPassword"
42              android:paddingLeft = "8dp"
43              android:singleLine = "true"
44              android:textColor = "@color/white"
45              android:textColorHint = "#a3a3a3"
46              android:textSize = "14sp" />
47          < Button
48              android:layout_width = "fill_parent"
49              android:layout_height = "40dp"
50              android:layout_gravity = "center_horizontal"      →按钮对外水平居中
51              android:layout_marginLeft = "35dp"
52              android:layout_marginRight = "35dp"
53              android:layout_marginTop = "15dp"
54              android:background = "@color/colorPrimary"       →背景引用颜色文件中颜色
55              android:text = "@string/btnLogin"      →文字引用 string.xml 中文字地址
56              android:textColor = "@color/white"     →字体颜色引用颜色文件中白色
57              android:textSize = "18sp" />
58          < LinearLayout
59              android:layout_width = "fill_parent"
60              android:layout_height = "fill_parent"
61              android:layout_marginLeft = "35dp"
62              android:layout_marginRight = "35dp"
63              android:layout_marginTop = "8dp"
64              android:gravity = "center_horizontal"
65              android:orientation = "horizontal">     →线性布局方向为水平
66              < TextView
67                  android:layout_width = "0dp"
68                  android:layout_height = "wrap_content"
69                  android:layout_weight = "1"         →线性布局中权重设置为 1
70                  android:gravity = "center_horizontal"
71                  android:padding = "8dp"
72                  android:text = "@string/registerNow"→文字引用 string.xml 中文字地址
73                  android:textColor = "@color/colorPrimary"   →背景引用颜色文件中颜色
74                  android:textSize = "14sp" />
75              < TextView
76                  android:layout_width = "0dp"
77                  android:layout_height = "wrap_content"
78                  android:layout_weight = "1"         →线性布局中权重设置为 1
```

```
79                  android:gravity = "center_horizontal"
80                  android:padding = "8dp"
81                  android:text = "@string/findPassword"→文字引用 string.xml 中文字地址
82                  android:textColor = "@color/colorPrimary"   →背景引用颜色文件中颜色
83                  android:textSize = "14sp" />
84          </LinearLayout >
85      </LinearLayout >
```

登录页面实现后，编辑框（如图 3-8 所示）与图 3-1 对比会发现用户名和密码编辑框未实现边框，其中边框需要自定义图片作为背景，具体实现将在 4.1.2 节中讲解。

图 3-8　登录页面实现效果

3.2　布局管理器


视频详解

任务陈述

“欢乐购商城”项目中每个页面都使用了布局管理器，例如，在登录和注册页面中整体采用了线性布局。本节重点讲解布局管理器中的线性布局、表格布局、相对布局、层布局、网格布局。本节以实现“欢乐购商城”首页作为案例（不包括列表），先以表格布局实现，然后采用线性布局和网格布局综合使用实现，如图 3-9 所示。同时对相对布局、层布局的布局管理器进行讲解，完成常用布局管理器的学习。

相关知识

3.1 节中学习了几种简单的界面组件，其中在登录页面实现中已经使用到了布局管理器线性布局，如果缺少布局管理器，组件排列杂乱，影响美观。本节将学习 Android 中提供的几种管理界面组件的布局管理器。

图 3-9　首页面表格布局

Android 中的布局管理器本身也是一个界面组件，所有的布局管理器都是 ViewGroup 类的子类，都可以当作容器类来使用。因此，可以在一个布局管理器中嵌套其他布局管理器。Android 中布局管理器可以根据运行平台来调整组件的大小，具有良好的平台无关性。Android 中用得最多的布局主要有：线性布局、表格布局、相对布局、层布局。


视频详解

3.2.1　线性布局

线性布局是最常用也是最基础的布局方式。在前面的示例中，就使用到了线性布局，它用 LinearLayout 类表示。线性布局和 Java 编程中 AWT 编程里的 FlowLayout 有些相似，它们都会将容器里的所有组件一个挨着一个排列。

它提供了水平和垂直两种排列方向，通过 android:orientation 属性进行设置，默认为垂直排列。

（1）当为水平方向时，不管组件的宽度是多少，整个布局只占一行，当组件宽度超过容器宽度时，超出的部分将不会显示。

（2）当为垂直方向时，整个布局文件只有一列，每个组件占一行，不管该组件宽度有多小。

在线性布局中，除了设置高度和宽度外，主要设置如下属性。

（1）android:gravity：设置布局管理器内组件的对齐方式，可以同时指定多种对齐方式的组合，多个属性之间用竖线隔开，但竖线前后不能出现空格。例如，bottom|center_horizontal 代表出现在屏幕底部，而且水平居中。

（2）android:orientation：设置布局管理器内组件的排列方向，可以设置为 vertical（垂直排列）或 horizontal（水平排列）。

（3）android:id：用于给当前组件指定一个 ID 属性，在 Java 代码中可以应用该属性单独引用该组件。为组件指定 ID 属性后，在 R.java 文件中，会自动派生一个对应的属性。在 Java 代码中，可以通过 findViewById()方法获取该属性。

（4）android:background：用于为该组件设置背景，可以是背景图片，也可以是背景颜色。为组件指定背景图片时，可以将准备好的图片复制到目录下，然后使用下面的代码进行设置。

```
android:background = "@drawable/background"
```

如果想指定背景颜色时，可以使用颜色值，例如，想要指定背景颜色为白色，可以使用下面的代码。

```
android:background = "#FFFFFF"
```

线性布局与 AWT 编程中 FlowLayout 的最明显的区别：在 FlowLayout 中组件一个个地排列到边界就会自动从下一行重新开始；在线性布局中如果一行的宽度或一列的高度超过了容器的宽度或高度，那么超出的部分将无法显示，如果希望超出的部分能够滚动显示，则需在外边包裹一个滚动组件，ScrollView（垂直滚动）或 HorizontalScrollView（水平滚动）。

在使用 LinearLayout 时，子控件可以设置 layout_weight。layout_weight 的作用是设置子控件在 LinearLayout 的重要度（控件的大小比重）。如果在一个 LinearLayout 里面放置两个 Button：Button1 和 Button2，Button1 的 layout_weight 设置为 1，Button2 的 layout_weight 设置为 2，且两个 Button 的 layout_width 都设置为 fill_parent，则 Button1 占据屏幕宽度的三分之二，而 Button2 占据三分之一。如果两个 Button 的 layout_width 都设置成 wrap_content，则情况刚好相反，Button1 占三分之一，Button2 占三分之二。

【例 3-5】 下面通过一个实例讲解线性布局管理使用方法，界面布局代码如下，程序运行效果如图 3-10 所示。

图 3-10　线性布局管理器实现效果

程序清单 3-9：char0306\app\src\main\res\layout\activity_main.xml

```
1    <ScrollView  →垂直滚动条
    xmlns:android="http://schemas.android.com/apk/res/android"
2       xmlns:app="http://schemas.android.com/apk/res-auto"
3       xmlns:tools="http://schemas.android.com/tools"
4       android:layout_width="match_parent"
5       android:layout_height="wrap_content"
6       tools:context=".MainActivity">
7       <LinearLayout
8           android:layout_width="match_parent"
9           android:layout_height="wrap_content"
10          android:orientation="vertical">          →线性布局方向为垂直
11              <LinearLayout
12                  android:layout_width="match_parent"
13                  android:layout_height="130dp"
14                  android:orientation="horizontal"> →线性布局方向为水平
15                  <TextView
16                      android:layout_width="wrap_content"
17                    android:layout_height="match_parent"
18                      android:gravity="center"       →文本编辑框内部居中
19                      android:layout_weight="1"      →权重比为1
20                      android:text="红色"
21                      android:textSize="25sp"
22                      android:background="#F00"/>  →背景颜色为白色
23                  <TextView
24                      android:layout_width="wrap_content"
25                    android:layout_height="match_parent"
26                      android:layout_weight="2"        →权重比为2
27                      android:gravity="center"         →文本编辑框内部居中
28                      android:text="绿色"
29                      android:textSize="25sp"
30                      android:background="#0F0"/>      →背景颜色为绿色
31                  <TextView
32                      android:layout_width="wrap_content"
33                      android:layout_height="match_parent"
34                      android:layout_weight="3"        →权重比为3
35                      android:text="蓝色"
36                      android:gravity="center"         →文本编辑框内部居中
37                      android:textSize="25sp"
38                      android:background="#00F"/>      →背景颜色为蓝色
39              </LinearLayout>
40           <LinearLayout
41               android:layout_width="match_parent"
42               android:layout_height="wrap_content"
43               android:gravity="center_horizontal"     →线性布局内部水平居中
44               android:orientation="vertical">         →线性布局方向垂直
45               <ImageView
46                   android:layout_width="150dp"
47                   android:layout_height="150dp"
48                   android:scaleType="fitCenter" →图片缩放格式保持纵横比,完全显示
49                   android:padding="3dp"          →图片内边距3dp
50                   android:src="@drawable/a000"/→引用图片地址
```

```
51                <ImageView
52                    android:layout_width = "150dp"
53                    android:layout_height = "150dp"
54                    android:scaleType = "fitCenter"→图片缩放格式保持纵横比,完全显示
55                    android:padding = "3dp"          →图片内边距 3dp
56                    android:src = "@drawable/a001"/>→引用图片地址
57                <ImageView
58                    android:layout_width = "150dp"
59                    android:layout_height = "150dp"
60                    android:scaleType = "fitCenter"→图片缩放格式保持纵横比,完全显示
61                    android:padding = "3dp"          →图片内边距 3dp
62                    android:src = "@drawable/a002"/>→引用图片地址
63                <ImageView
64                    android:layout_width = "150dp"
65                    android:layout_height = "150dp"
66                    android:scaleType = "fitCenter"→图片缩放格式保持纵横比,完全显示
67                    android:padding = "3dp"                    →图片内边距 3dp
68                    android:src = "@drawable/a003"/>           →引用图片地址
69                <ImageView
70                    android:layout_width = "150dp"
71                    android:layout_height = "150dp"
72                    android:scaleType = "fitCenter"→图片缩放格式保持纵横比,完全显示
73                    android:padding = "3dp"                    →图片内边距 3dp
74                    android:src = "@drawable/a004"/>           →引用图片地址
75            </LinearLayout>
76        </LinearLayout>
77    </ScrollView>
```

3.2.2　表格布局


视频详解

表格布局是指以行和列的形式来管理界面组件，由 TableLayout 类表示，不必明确声明包含几行几列，而通过添加 TableRow 来添加行，在 TableRow 中添加组件来添加列。

TableRow 就是一个表格行，本身也是容器，可以不断地添加其他组件，每添加一个组件就是在该行中增加一列，如果直接向 TableLayout 中添加组件，而没有添加 TableRow，那么该组件将会占用一行。

在表格布局中，每列的宽度都是一样的，列的宽度由该列中最宽的那个单元决定，整个表格布局的宽度则取决于父容器的宽度，默认总是占满父容器本身。

TableLayout 继承了 LinearLayout，因此它完全支持 LinearLayout 所支持的全部 XML 属性，另外，TableLayout 还增加了自己所特有的属性。

(1) android:collapseColumns：隐藏指定的列，其值为列所在的序号，从 0 开始，如果需要隐藏多列，可用逗号隔开这些序号。

(2) android:shrinkColumns：收缩指定的列以适合屏幕，使整行能够完全显示，不会超出屏幕，用于当某一行的内容超过屏幕的宽度时，会使该列自动换行，其值为列所在的序号。如果没有该属性，则超出屏幕的部分会自动截取，不会显示。

(3) android:stretchColumns：尽量把指定的列填充空白部分。该属性用于某一行的内容不足以填充整个屏幕，这样指定某一列的内容扩张以填满整个屏幕，其他列的宽度不变。

如果某一列有多行，而每行的列数可能不相同，那么可扩展列的宽度是一致的，不会因为某一行有多余的空白而填充整行。也就是说，不管在哪一行，它的宽度都是相同的。

（4）android:layout_column：控件在 TableRow 中所处的列。如果没有设置该属性，默认情况下，控件在一行中是一列挨着一列排列的。通过设置该属性，可以指定控件所在的列，这样就可以达到中间某一个列为空的效果。

（5）android:layout_span：该控件所跨越的列数，即将多列合并为一列。

【例 3-6】 下面以“欢乐购商城”首页面商品分类图片按钮栏为例，介绍表格使用方法，界面布局代码如下，程序运行效果如图 3-11 所示。

图 3-11 表格布局管理器实现效果

程序清单 3-10：char0307\app\src\main\res\layout\activity_main.xml

```
<LinearLayout 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:orientation="vertical"                    →线性布局垂直方向
    tools:context=".MainActivity">
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:scaleType="centerCrop"                →图片缩放格式保持纵横比，覆盖控件
        android:src="@drawable/jhs" />                →引用图片地址
    <TableLayout
        android:layout_gravity="center_horizontal"    →表格布局对外水平居中摆放
```

```
15              android:layout_width = "wrap_content"
16              android:layout_height = "wrap_content"
17              android:gravity = "center">      →组件内部居中显示
18              < TableRow >                     →添加一行
19                  < ImageView
20                      android:layout_width = "73dp"
21                      android:layout_height = "73dp"
22                      android:clickable = "true"          →可单击
23                      android:src = "@drawable/a" />      →引用图片地址
24                  < ImageView
25                      android:layout_width = "73dp"
26                      android:layout_height = "73dp"
27                      android:src = "@drawable/b" />      →引用图片地址
28                  < ImageView
29                      android:layout_width = "73dp"
30                      android:layout_height = "73dp"
31                      android:src = "@drawable/c" />      →引用图片地址
32                  < ImageView
33                        android:layout_width = "73dp"
34                      android:layout_height = "73dp"
35                      android:src = "@drawable/d" />      →引用图片地址
36                  < ImageView
37                      android:layout_width = "73dp"
38                      android:layout_height = "73dp"
39                      android:src = "@drawable/e" />      →引用图片地址
40              </TableRow >
41              < TableRow >                                →添加一行
42                  < ImageView
43                      android:layout_width = "73dp"
44                      android:layout_height = "73dp"
45                      android:src = "@drawable/f" />      →引用图片地址
46                  < ImageView
47                      android:layout_width = "73dp"
48                      android:layout_height = "73dp"
49                      android:src = "@drawable/g" />      →引用图片地址
50                  < ImageView
51                      android:layout_width = "73dp"
52                      android:layout_height = "73dp"
53                      android:src = "@drawable/h" />      →引用图片地址
54                  < ImageView
55                      android:layout_width = "73dp"
56                      android:layout_height = "73dp"
57                      android:src = "@drawable/i" />      →引用图片地址
58                  < ImageView
59                      android:layout_width = "73dp"
60                      android:layout_height = "73dp"
61                      android:src = "@drawable/k" />      →引用图片地址
62              </TableRow >
63          </TableLayout >
64      </LinearLayout >
```

3.2.3　相对布局


视频详解

相对布局，顾名思义就是相对于某个组件的位置，由 RelativeLayout 类表示，这种布局

的关键是找到一个合适的参照物，如果甲组件的位置需要根据乙组件的位置来确定，那么要求先定义乙组件，再定义甲组件。

在相对布局中，每个组件的位置可通过它相对于某个组件的方位以及对齐方式来确定，因此相对布局中常见的属性如表 3-3 所示。由于父容器是确定的，所以与父容器方位与对齐的关系取值为 true 或 false。

表 3-3　相对布局中常用属性设置

属　　性	说　　明
android：layout_centerHorizontal	设置该组件是否位于父容器的水平居中位置
android：layout_centerVertical	设置该组件是否位于父容器的垂直居中位置
android：layout_centerInParent	设置该组件是否位于父容器的正中央位置
android：layout_alignParentTop	设置该组件是否与父容器顶端对齐
android：layout_alignParentBottom	设置该组件是否与父容器底端对齐
android：layout_ alignParentLeft	设置该组件是否与父容器左边对齐
android：layout_ alignParentRight	设置该组件是否与父容器右边对齐
android：layout_toRightOf	指定该组件位于给定的 ID 组件的右侧
android：layout_toLeftOf	指定该组件位于给定的 ID 组件的左侧
android：layout_above	指定该组件位于给定的 ID 组件的上方
android：layout_below	指定该组件位于给定的 ID 组件的下方
android：layout_alignTop	指定该组件与给定的 ID 组件的上边界对齐
android：layout_ alignBottom	指定该组件与给定的 ID 组件的下边界对齐
android：layout_ alignLeft	指定该组件与给定的 ID 组件的左边界对齐
android：layout_ alignRight	指定该组件与给定的 ID 组件的右边界对齐

【例 3-7】 在智能手机中，当系统中有软件更新时，经常会显示提示软件更新页面。本实例使用相对布局实现一个显示软件更新提示的界面，提示文字相对于父容器居中，“以后再说”按钮相对于提示文字右对齐、底部方向，“现在更新”按钮相对于“以后再说”按钮底部对齐、左部方向。界面布局代码如下，程序运行效果如图 3-12 所示。

图 3-12　相对布局管理器实现效果

程序清单 3-11：char0308\app\src\main\res\layout\activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">                          →线性布局垂直方向
    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:textSize="20sp"      →图片缩放格式保持纵横比,覆盖控件
         android:text="发现购物商城新的版本,您现在要更新吗?"      →引用图片地址
         android:layout_centerInParent="true"/>
    <Button                                                 →表格布局对外水平居中摆放
         android:text="现在更新"
         android:id="@+id/button1"
         android:layout_width="wrap_content"                →组件内部居中显示
         android:layout_height="wrap_content"               →添加一行
         android:layout_below="@id/text"
         android:layout_toLeftOf="@id/button2"/>
    <Button
         android:text="以后再说"                            →可单击
         android:id="@+id/button2"                          →引用图片地址
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignRight="@id/text"
         android:layout_below="@id/text"/>                  →引用图片地址
</RelativeLayout>
```

3.2.4　层布局


视频详解

层布局也叫帧布局，由 FrameLayout 类表示。其每个组件占据一层，后面添加的层会覆盖前面的层，后面的组件会叠放在先前的组件之上。如果后面组件大于前面的组件，那么前面的组件将会完全被覆盖，不可见；如果后面组件无法完全覆盖前面的组件，则未覆盖部分显示先前的组件。该布局在开发中设计地图时经常用到，因为是按层次方式布局，需要实现层面显示的样式时就可以采用这种布局方式，比如要实现一个类似百度地图的布局，我们移动的标志是在一个图层的上面。

【例 3-8】 通过实例讲解 Android 程序中使用层布局管理器，界面布局代码如下。

程序清单 3-12：char0309\app\src\main\res\layout\activity_main.xml

```
<FrameLayout 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"
```

```
6        tools:context = ".MainActivity">
7        <TextView
8            android:layout_width = "300dp"        →控件宽度为 300dp
9            android:layout_height = "300dp"       →控件高度为 300dp
10           android:background = "#f00"           →背景颜色为红色
11           android:layout_gravity = "center"     →控件对外水平居中摆放
12           />
13       <TextView
14           android:layout_width = "200dp"        →控件宽度为 200dp
15           android:layout_height = "200dp"       →控件高度为 200dp
16           android:background = "#0f0"           →背景颜色为绿色
17           android:layout_gravity = "center"     →控件对外水平居中摆放
18           />
19       <TextView
20           android:layout_width = "100dp"        →控件宽度为 100dp
21           android:layout_height = "100dp"       →控件高度为 100dp
22           android:background = "#00f"           →背景颜色为蓝色
23           android:layout_gravity = "center"     →控件对外水平居中摆放
24           />
25       <TextView
26           android:layout_width = "50dp"         →控件宽度为 50dp
27           android:layout_height = "50dp"        →控件高度为 50dp
28           android:background = "#fff"           →背景颜色为白色
29           android:layout_gravity = "center"     →控件对外水平居中摆放
30           />
31       <TextView
32           android:layout_width = "20dp"         →控件宽度为 20dp
33           android:layout_height = "20dp"        →控件高度为 20dp
34           android:background = "#000"           →背景颜色为黑色
35           android:layout_gravity = "center"     →控件对外水平居中摆放
36           />
37   </FrameLayout >
```

程序运行效果如图 3-13 所示。


视频详解

3.2.5 网格布局

网格布局由 GridLayout 代表，是 Android 4.0 新增的布局管理器，因此需要在 Android 4.0 之后的版本中才能使用该布局管理器。如果希望在更早的 Android 平台上使用该布局管理器，则需要导入相应的支撑库。GridLayout 的作用类似于 HTML 中的 table 标签，它把整个容器划分成若干行和若干列个网格，每个网格可以放置一个组件。除此之外，也可以设置一个组件横跨多个列、一个组件纵跨多个行。网格布局和 TableLayout（表格布局）有点儿类似，不过它功能更多，使用更加方便，具有以下优势：

（1）可以自己设置布局中组件的排列方式。

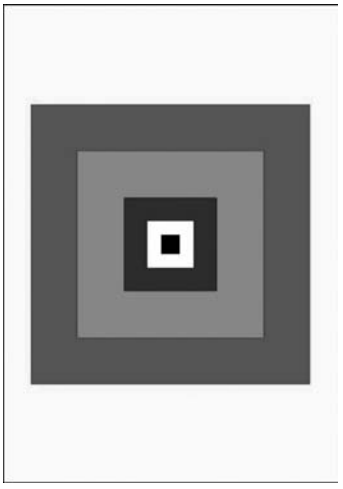
图 3-13 层布局管理器实现效果

(2) 可以自定义网格布局有多少行,多少列。

(3) 可以直接设置组件位于某行某列。

(4) 可以设置组件横跨几行或者几列。

3.2.6　项目实战——首页布局


视频详解

在讲解表格布局时,以“欢乐购商城”首页面商品分类图片按钮栏为例进行实现,现对表格布局实现案例使用网格布局进行优化,把表格布局替换成网格布局,界面布局代码如下,程序运行效果如图 3-11 所示。

程序清单 3-13：char0310\app\src\main\res\layout\activity_main.xml

```
<LinearLayout 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:orientation="vertical"                    →线性布局垂直方向
    tools:context=".MainActivity">
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:scaleType="centerCrop"                →图片缩放格式保持纵横比,覆盖控件
        android:src="@drawable/jhs" />                →引用图片地址
     <GridLayout
        android:layout_gravity="center_horizontal"→网格布局对外水平居中摆放
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"                      →组件内部居中显示
        android:columnCount="5"                       →网格布局设置 5 列
        android:orientation="horizontal"              →控件水平摆放
        android:rowCount="2">                         →网格布局设置 2 行
            <ImageView
                android:layout_width="73dp"   →可单击
                android:layout_height="73dp" →引用图片地址
                android:clickable="true"
                android:src="@drawable/a" />
            <ImageView
                android:layout_width="73dp"   →引用图片地址
                android:layout_height="73dp"
                android:src="@drawable/b" />
            <ImageView
```

```
31                    android:layout_width = "73dp"          →引用图片地址
32                    android:layout_height = "73dp"
33                    android:src = "@drawable/c" />
34                <ImageView
35                    android:layout_width = "73dp"          →引用图片地址
36                    android:layout_height = "73dp"
37                    android:src = "@drawable/d" />
38                <ImageView
39                    android:layout_width = "73dp"          →引用图片地址
40                    android:layout_height = "73dp"
41                    android:src = "@drawable/e" />         →添加一行
42                <ImageView
43                    android:layout_width = "73dp"
44                    android:layout_height = "73dp"
45                    android:src = "@drawable/f" />         →引用图片地址
46                <ImageView
47                    android:layout_width = "73dp"
48                    android:layout_height = "73dp"
49                    android:src = "@drawable/g" />         →引用图片地址
50                <ImageView
51                    android:layout_width = "73dp"
52                    android:layout_height = "73dp"
53                    android:src = "@drawable/h" />         →引用图片地址
54                <ImageView
55                    android:layout_width = "73dp"
56                    android:layout_height = "73dp"
57                    android:src = "@drawable/i" />         →引用图片地址
58                <ImageView
59                    android:layout_width = "73dp"
60                    android:layout_height = "73dp"
61                    android:src = "@drawable/k" />         →引用图片地址
62        </GridLayout>
63    </LinearLayout>
```

本章小结

本章围绕“欢乐购商城”项目登录页面、注册页面和首页面商品分类栏引入 Android 基础界面控件和布局管理器讲解。详细介绍了几种最基本的界面组件的功能和常用属性，包括文本显示框、文本编辑框和按钮等，并通过实现登录页面案例演示了基本界面控件具体使用用法。为了使这些组件排列美观，继续学习了 Android 中几种常见的布局管理器，包括线性布局、表格布局、相对布局、层布局和网格布局，它们各有优缺点。线性布局方便，需使用的属性较少，但不够灵活；表格布局中通过 TableRow 添加行，每列的宽度一致；相对布局

则通过提供一个参照物来准确定义各个控件的具体位置，通常在一个实例中会用到多种布局，把各种布局结合起来达到所要的界面效果。网格布局相对表格布局使用更加方便，只需要设置行、列、摆放方向就可以控制控件摆放。

自测习题

1. 下列(　　)可作 EditText 编辑框的提示信息。

A. android:inputType　　B. android:text

C. android:digits　　D. android:hint

2. 为下面控件添加 android:text="Hello"属性，运行时无法显示文字的控件是(　　)。

A. Button　　B. EditText　　C. ImageButton　　D. TextView

3. 下列选项中，前后两个类不存在继承关系的是(　　)。

A. TextView、EditText　　B. TextView、Button

C. Button、ImageButton　　D. ImageView、ImageButton

4. 假设手机屏幕宽度为 400px，现采取水平线性布局放置 5 个按钮，设定每个按钮的宽度为 100px，那么该程序运行时，界面显示效果为(　　)。

A. 自动添加水平滚动条，拖动滚动条可查看 5 个按钮

B. 只可以看到 4 个按钮，超出屏幕宽度部分无法显示

C. 按钮宽度自动缩小，可看到 5 个按钮

D. 程序运行出错，无法显示

5. 表格布局中，设置某一列是可扩展的正确的做法是(　　)。

A. 设置 TableLayout 的属性：android:stretchColumns="x"，x 表示列的序号

B. 设置 TableLayout 的属性：android:shrinkColumns="x"，x 表示列的序号

C. 设置具体列的属性：android:stretchable="true"；

D. 设置具体列的属性：android:shrinkable="true"；

6. 相对布局中，设置以下属性时，属性值只能为 true 或 false 的是(　　)。

A. android:layout_below　　B. android:layout_alignParentLeft

C. android:layout_alignBottom　　D. android:layout_toRightOf

7. 布局文件中有一个按钮(Button)，如果要让该按钮在其父容器中居中显示，正确的设置是(　　)。

A. 设置按钮的属性：android: layout_gravity="center"

B. 设置按钮的属性：android: gravity="center"

C. 设置按钮父容器的属性：android: layout_gravity="center"

D. 设置按钮父容器的属性：android: gravity="center"

8. 根据所学的相对布局的知识，设计出如图 3-14 所示界面，要求在文本编辑框内只能输入数字，并且输入的内容会以“密码隐藏”的形式显示。

9. 运用所学知识，实现“欢乐购商城”注册页面。

图 3-14　相对布局实现输入框