CHAPTER 3

# 第3章

# 基本控件与布局管理器

**本章主要内容：**

android. wigdet 包；
Form widget 控件；
TextFields 控件；
布局管理器；
Image 和 Media 控件；
Time 和 Date 控件。

## 3.1 widget 包与控件

开发 Android 应用程序，设计制作能吸引用户的界面(User Interface，UI)是最重要的，UI 对于应用软件恰如外貌对于美女。界面能够影响用户的第一印象，决定了用户是否愿意体验我们开发的软件。Android 为开发 UI 提供了很多控件(组成 UI 的元素)，这些控件位于 android. widget 包中，继承自 View 类或 ViewGroup 类。继承 View 类的控件一般都是可见的，可以与使用者交互；继承 ViewGroup 类的控件一般都是不可见的布局控制器，用于存放其他的控件或布局。

Android 提供了一系列 View 类控件(如按钮、文本框、下拉列表)和 ViewGroup 布局(如线性布局、相对布局等)，熟练使用这些控件可以快速开发界面精美的 Android 应用程序。应用软件的界面基本都是由控件和布局有机组合实现，对于界面复杂的软件可以采用布局中嵌套布局的形式实现。界面设计的一般结构如图 3.1 所示。

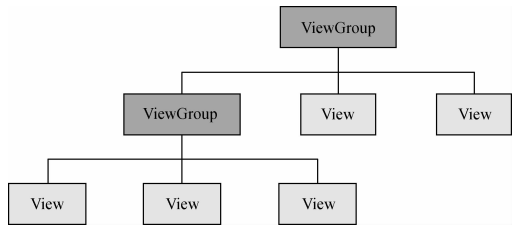


图 3.1 UI 设计的一般结构

### 3.1.1　控件的分类

打开 Android 项目中 res 文件夹下 layout 文件夹中的布局文件 main. xml，布局设计界面主要分为左右两个部分。左边是 Android 提供的系统控件，右边是当前界面的预览视图。界面的编辑可以通过下方的 Graphic Layout 按钮和 main. xml 按钮切换，图形化的编辑方式较为直观，操作比较简单，将选定的控件直接拖到右侧的预览视图即可，所见即所得，但是处理较为复杂的界面设计时力不从心。XML 文件编辑方式通过标签和属性定义控件，可以更加灵活地对布局进行设计，在处理复杂界面设计时一般都采用这种方法。

Graphic Layout 界面的上方是对设备屏幕的设定，包括对屏幕尺寸的设定、横竖屏的设定(Portrait 为竖屏，Landscape 为横屏)、显示的样式等。

单击 Pallete 右边的下拉三角，可以选择系统控件的预览模式。默认是 Show Small Previews，如图 3.2 所示。切换为 Show Only Icons 的效果如图 3.3 所示。

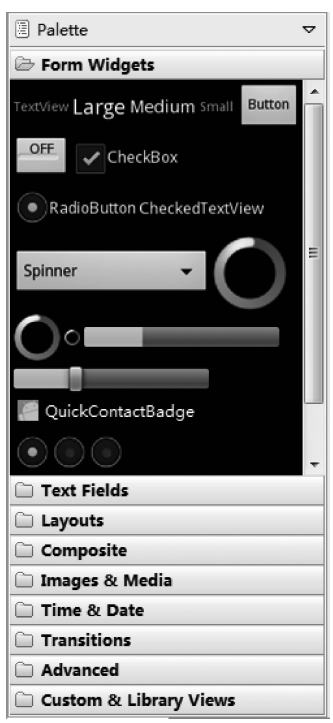


图 3.2　预览模式的控件

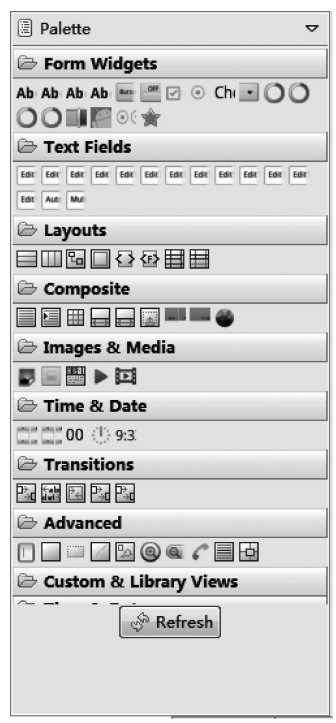


图 3.3　图标模式的控件

Android 的系统控件共分为 8 类，基本包括应用程序中所需要的大部分控件，在有特殊需要的情况下，可以自定义其他控件。

(1) Form Widgets，表单类控件包括 TextView(文本标签)、Button(按钮)、RadioButton(单选按钮)、CheckBox(复选按钮)、ProgressBar(进度条)、Spinner(下拉列表)等。

(2) Text Fields，文本框控件，根据输入内容的不同分为普通文本框、电话文本框、数字文本框等。

(3) Layouts，布局控件包括 LinearLayout(线性布局)、RelativeLayout(相对布局)、

FrameLayout(单帧布局)、TableLayout(表格布局),另外再加上 AbsoluteLayout(绝对布局),组成 Android 应用开发中的 5 大布局管理器。通过这 5 种布局管理器的相互嵌套可以实现复杂的界面布局。

(4) Composite,组合控件,这是比较复杂的控件,包括 ListView(列表视图)、GridView(表格视图)、ScrollView(滚动视图)、TabHost(标签容器)、WebView(webkit 内核浏览器)等。

(5) Images 和 Media,图片和媒体控件,包括 ImageView(图片视图)、ImageButton(图片安全)、VideoView(视频视图)等控件。

(6) Time 和 Date,包括 TimePicker(时间选择器)、DatePicker(日期选择器)、AnalogClock(时钟)、DigitalClock(电子时钟)等控件。

(7) Transitions,包括 ImageSwitcher(图片切换)、ViewSwitcher(视图切换)等控件。

(8) Advanced,包括一些高级控件,使用时稍微复杂一些。

### 3.1.2 UI 的编辑方式

Android 应用程序中 UI 的设计主要有两种方式,一种是使用 Layout 文件设计布局,然后指定 Activity 使用该布局文件;另一种是直接继承 View 视图,使用代码绘制 Activity 的视图(游戏中主要采用这种)。借助 Eclipse 完成第一种 UI 布局比较简单。使用 Layout 布局文件充当界面时,首先要创建布局文件。

展开项目,选择 Layout 文件夹,单击右键,选择 New→Other 命令,打开“新建文件”窗口(第一次新建 Android XML 文件需要从 Other 中选择,以后可以直接在 New 中选择),如图 3.4 所示。选择 Android XML File,单击 Next 按钮,打开文件配置窗口,如图 3.5 所示。

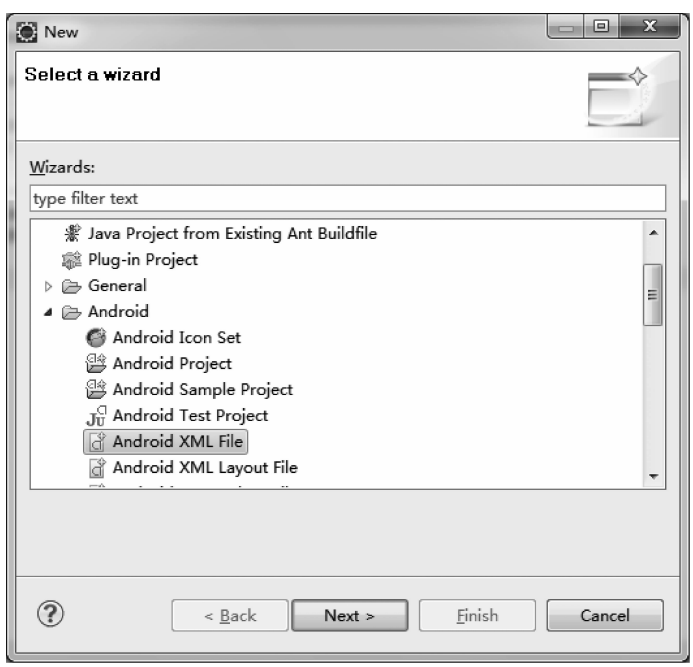


图 3.4 “新建文件”窗口


图 3.5　XML 文件配置窗口

(1) Resource Type：新建文件的类型，默认是 Layout，即布局文件，可以选择其他文件类型，如 Values，用于存放字符串数据。

(2) Project：所属项目。

(3) File：文件名，只能用小写字母、数字和_，并且以小写字母开头。

(4) Root Element：选择布局文件的布局管理器，默认选择 LinearLayout 线性布局。

确定布局之后，单击 Finish 按钮，结束文件创建过程。布局打开时默认是图形化编辑模式(Graphic Layout)。根据 UI 的设计，选择左侧恰当的控件，直接拖动到右侧的预览视图中即可。这种图形化编辑模式比较简单，但无法编辑复杂的 UI。单击底部的文件名按钮，可以切换到代码视图，所有的控件都可以使用标签直接在代码视图中创建。

### 3.1.3　控件的属性

控件添加到布局文件中后，可以根据需要调整相应属性。在 Graphic Layout 视图中可以在控件上单击右键，配置控件的各项属性。在代码视图中需要在标签内部指定属性名，然后赋值。Android 中控件的属性比较多，表 3.1 列出了常用的一些属性，以及它们的含义和取值信息。

表 3.1 Android 控件的常用属性

| 属性名 | 简　介 | 取值信息 |
|---|---|---|
| android:id | 控件的 ID,具有唯一性 | 自定义 |
| android:layout_width | 控件宽度 | 系统值:<br>fill_parent 填充(充满)父容器<br>match_parent 匹配父容器<br>wrap_content 包围内容<br>自定义值:<br>直接指定控件尺寸 |
| android:layout_height | 控件高度 | |
| android:text | 显示的文本信息 | 引用 value 中字符串,或直接指定字符串值 |
| android:background | 设定背景图片或颜色 | 引用 drawable 中的图片,或直接给出 RGB 颜色 |
| android:textColor | 文字颜色 | |
| android:textSize | 文字大小 | |
| android:textStyle | 文字风格 | normal、bold、italic |
| android:maxLines | 最大行数 | |
| android:gravity | 文字的对齐方式 | top、bottom、left、right 等 |
| android:password | 文本输入框是否是密码 | true、false |
| android:selectAllOnFocus | 文本输入框在获得焦点时全选文字 | true、false |
| android:inputType | 文本输入框的输入内容 | number、date、time 等 |
| android:textAppearance | 文字的显示大小 | ? android:attr/textAppearanceLarge 等系统值 |
| android:padding | 内容距控件边缘的填充间距 | |
| android:onClick | 控件单击时执行的方法 | 方法名 |
| android:layout_gravity | 控件在父容器中的对齐方式 | top、bottom、left、right 等 |
| android:layout_margin | 距离其他控件的边缘 | |

以上列出的控件属性并没有区分控件,有的属性需要用于特定的控件,比如 android:inputType 属性用于指定文本输入框的类型,明确用于输入什么类型的内容。代码 03-1 是对上述属性的具体使用,该布局文件位于项目 Part03,布局文件名为 mylayout.xml,完整代码请查看随书配套资料。该段代码所设计的 UI 如图 3.6 所示。

**代码 03-1**

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/textView1"
        android:layout_width="fill_parent"
        android:layout_height="100px"
        android:text="文本视图控件"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:background="#FFFFFF"
```

```
            android:textColor = "#000000"
            />
        <Button
            android:id = "@ + id/button1"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "按钮 1" />

        <Button
            android:id = "@ + id/button2"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:text = "@string/bt2" />
        <Button
            android:id = "@ + id/button3"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "按钮 3"
            android:layout_gravity = "right"
            />
    </LinearLayout>
```

布局文件由标签构成，所有标签都需要成对出现，形如＜LinearLayout＞＜/LinearLayout＞，标签内部可以嵌入其他标签；或者＜Botton/＞，标签内部不能嵌入其他标签(如果有 HTML 基础，对这种编辑方式会感到亲切)。每个标签(某种控件)的属性写在标签的开始部分，属性结尾没有(这种属性赋值方式与 CSS 类似)。

图 3.6 中使用了 4 种标签完成布局。最外层是 LinearLayout 标签，这是线性布局管理器，对于放入该标签内部的控件按照线性排列。属性 android:layout_width 和 android:layout_height 指定线性布局管理器的宽和高，值 match_parent 指明宽高需要匹配父容器(此处父容器指手机显示屏区域)。android:orientation 属性对于线性布局非常重要，指明是水平线性(取值 horizontal)或是竖直线性(取值 vertical)。


图 3.6　mylayout 布局界面

LinearLayout 标签的内部是一个 TextView 标签和三个 Button 标签。对于控件一般需要指明其 android:id 属性(在需要的情况下，布局管理器也可以指明 id 属性)。android:id 属性的赋值需要使用@＋id/id 名(id 名是自定义的)，其中＋表示当修改完某个布局文件并保存后，系统会自动在 R.java 文件中生成相应的 int 类型变量，以方便在代码中采用R.id.id 名的方式引用该控件。例如，@＋id/button1 会在 R.java 文件中生成int button1＝value 的属性，其中 value 是一个十六进制的数，是自动计算产生的。

如果需要在布局文件中引用其他资源，需要使用@id/id 名(引用其他控件)、@string/字符串名(引用 values 文件夹下 strings 中字符串)、@drawable/图片名(引用 drawable-×××文

件下中图片)或@layout/布局名(引用其他布局)的方式实现。比如 android:text="@string/bt2",其值是引用 strings.xml 文件中名为 bt2 的字符串。

Android 控件的属性比较多,这些属性大多来自 View 类。另外,不同控件又具有特殊属性,在使用过程中需要注意以下几点。

- 先了解一般常用属性及其作用,切勿贪多。
- 掌握 android:id 属性的使用,包括如何增加 id 和引用其他资源。
- 控件都需要设置宽高属性。
- 控件的赋值建议采用引用赋值方式,这样便于后期的国际化,不过为了直观在后续讲解中大多采用了直接赋值的方式。

## 3.2 Form Widgets

Form Widget 列表中包含 TextView、Button、ToggleButton、RadioButton、CheckBox、CheckedTextView、ProgressBar、SeekBar、Spinner、QuickContactBadge、RadioGroup 和 RatingBar。下面对其中比较重要的控件逐一介绍。

### 3.2.1 TextView

用于显示字符串的文本标签,不能编辑。TextView 的属性 android:textAppearance 规定文字的显示方式,可以使用如下系统值,效果如图 3.7 所示。

(1) ? android:attr/textAppearanceLarge,大字号显示文字。

(2) ? android:attr/textAppearanceMedium,中等字号显示文字。

(3) ? android:attr/textAppearanceSmall,小字号显示文字。

(4) 默认,默认字号显示文字。


图 3.7 TextView 不同字号的显示

### 3.2.2 Button

Button 控件几乎是家喻户晓了,主要用于单击操作,处理相应事件。在 Android 中按钮的事件处理方式有两种,一种是直接给按钮注册监听器(这种方式与 J2SE 中事件处理模型一致),另一种是 android:onClick 属性,直接指定处理单击事件的方法。

代码 03-2 演示两种事件处理方式,完整代码请查看随书配套资料 Part03 项目,MainActivity.java 源代码文件。在 onCreate 方法中,初始化三种控件,通过 findViewById(通过 Id 获取 View 控件)方法,获取布局文件 layoutbutton 中的三个控件,并强转为对于类型。b1 按钮直接注册监听器(采用匿名内部类实现)OnClickListener(注意此监听器是 android.view.View.OnClickListener,不要引错),监听按钮的单击。b2 按钮没有注册监听

器，在布局文件（如代码 03-3 所示）中，采用 android:onClick="buttonClick"属性，指明由方法 buttonClick 处理该按钮的单击事件。

**代码 03-2**

```
@Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layoutbutton);
        //实例化控件
        b1 = (Button) findViewById(R.id.bt_1);
        b2 = (Button) findViewById(R.id.bt_2);
        tv = (TextView) findViewById(R.id.tv_1);
        //注册监听器，监听器由匿名内部类实现
        b1.setOnClickListener(new OnClickListener(){
            @Override
            public void onClick(View v) {
                tv.setText("监听器处理按钮单击!");
            }
        });
    }
    //处理按钮单击事件
    public void buttonClick(View v){
        tv.setText("方法处理按钮单击!");
    }
```

定义 buttonClick 方法时，有两点需要注意。一是此类方法必须是 public 修饰，二是参数列表只能有一个 View 类型参数。当指定按钮被单击时，作为 View 传入此方法。

**代码 03-3**

```
<TextView
           android:id = "@ + id/tv_1"
           android:layout_width = "fill_parent"
           android:layout_height = "wrap_content"
           android:text = "TextView"
           android:textAppearance = "?android:attr/textAppearanceLarge"
           />
      <Button
           android:id = "@ + id/bt_1"
           android:layout_width = "fill_parent"
           android:layout_height = "wrap_content"
           android:text = "注册监听器响应单击" />
      <Button
           android:id = "@ + id/bt_2"
           android:layout_width = "fill_parent"
           android:layout_height = "wrap_content"
           android:text = "指明响应单击的方法"
           android:onClick = "buttonClick"
            />
```

### 3.2.3 ToggleButton

ToggleButton(开关按钮)是 Android 系统中比较简单的一个组件,是一个具有选中和未选择双状态的按钮,并且需要为不同的状态设置不同的显示文本。常用属性如下。

(1) android:textOn="开启",当按钮处于选中状态显示的文字。

(2) android:textOff="关闭",当按钮处于未选中状态显示的文字。

(3) android:disabledAlpha="0.1",当按钮未选中时,按钮的 Alpha 值,取值范围为 0～1。

ToggleButton 常用的监听器是 OnClickListener 和 OnCheckedChangeListener,都可以监听按钮的状态变化。代码 03-4 演示了两种监听器处理 ToggleButton 的状态改变,完整代码参考随书配套资料 Part03 项目,MainActivity.java 文件,布局文件是 layouttogglebutton.xml。

**代码 03-4**

```
ToggleButton tb1,tb2;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //测试 toggleButton
        setContentView(R.layout.layouttogglebutton);
        //初始化控件
        tb1 = (ToggleButton) findViewById(R.id.toggleButton1);
        tb2 = (ToggleButton) findViewById(R.id.toggleButton2);
        //注册监听器
        tb1.setOnClickListener(new OnClickListener(){
            @Override
            public void onClick(View v) {
                if(tb1.isChecked()){
                    setTitle("tb1 On!"); //设置应用程序的标题
                }else{
                    setTitle("tb1 Off!"); //设置应用程序的标题
                }
            }}
        );
        tb2.setOnCheckedChangeListener(new OnCheckedChangeListener(){
            @Override
            public void onCheckedChanged(CompoundButton buttonView,
                    boolean isChecked) {
                if(isChecked){
                    setTitle("tb2 开启!");
                }else{
                    setTitle("tb2 关闭!");
                }
            }
        });
    }
```

在 OnClickListener 监听器中,通过 ToggleButton 的 isChecked 方法,判断按钮是否选

中。在 OnCheckedChangeListener 监听器中，可以通过 onCheckedChange 方法的第二个参数判断按钮是否处于选中状态。运行效果如图 3.8 所示。

ToggleButton 是 CompoundButton 类的子类。CompoundButton 继承 Button 类，具有两种状态：选中或未选中，按钮被单击时状态会发生切换，该类按钮常常被用于功能开关，如开启背景音乐或关闭背景音乐。CompoundButton 类的另外两个常用子类是 RadioButton 和 CheckBox。它们都比 Button 类多一个监听器，即 OnCheckedChangeListener。

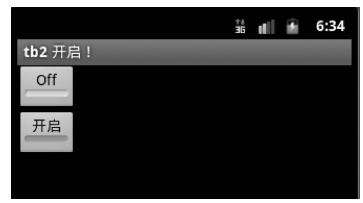


图 3.8　ToggleButton 运行效果

### 3.2.4　RadioButton 与 RadioGroup

RadioButton 是单选按钮，继承自 CompoundButton。RadioGroup 是单选按钮组，继承自 LinearLayout 类，本身不能直接使用，需要与 RadioButton 配合使用，用于将 RadioButton 聚合成一组。

代码 03-5 演示如何使用 RadioGroup，完整代码请参考随书配套资料（这句话出现了很多次，目的是初学者不要盲目照抄下面的代码，然后就理所当然地运行，那样肯定是无法运行的）Part03 项目，MainActivity.java 文件，布局文件是 layoutradiogroup.xml。

**代码 03-5**

```
RadioGroup rgroup;
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    //测试 RadioButton 与 RadioGroup
    setContentView(R.layout.layoutradiogroup);
    tv = (TextView) findViewById(R.id.tv_rg);
    rgroup = (RadioGroup) findViewById(R.id.radioGroup1);
    rgroup.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(RadioGroup group, int checkedId) {
            tv.setText("当前选中控件 Id: " + checkedId);
            RadioButton rb = (RadioButton) findViewById(checkedId);
            tv.append("\n" + rb.getText());
        }}
    );
}
```

虽然每个 RadioButton 都有 OnClickListener 监听器和 OnCheckedChangeListener 监听器，但直接处理每个单选按钮比较麻烦，需要每个都做判断，所以一般情况下可以给 RadioGroup 添加监听器，用于判断该组中哪个单选按钮被选中。

仔细阅读代码 03-5，会发现 RadioGroup 的监听器也叫 OnCheckedChangeListener，但在实现该监听器时，又添加了 RadioGroup.，这说明此处的 OnCheckedChangeListener 监听器是 RadioGroup 类中一个内部接口，为了避免产生歧义，在创建该接口的实现类时直接指

定了该接口所属的外部类，采用形如外部类。内部接口的方式创建匿名内部类。这种创建接口的方式在以后会经常见到。监听器中方法 onCheckedChanged 的第二个参数即为备选按钮的 ID。代码运行效果如图 3.9 所示。


图 3.9　RadioGroup 运行效果

### 3.2.5　CheckBox

复选框，继承自 CompoundButton 类，具有 OnClickListener 监听器和 OnCheckedChangeListener 监听器，通过 isChecked 方法，判断该按钮是否处于选中状态。

### 3.2.6　CheckedTextView

该类继承自 TextView，实现了 Checkable 接口，常用于 ListView 中。使用自定义适配器，根据指定数据与布局样式填充相应数据。在后续章节学习完 ListView 与适配器后，感兴趣的读者可以尝试 CheckedTextView 与 ListView 的配合使用。

### 3.2.7　ProgressBar

Android 中的进度条有两类：默认的环形进度条和需要设置的水平进度条。进度条的大小和状态可以通过 style 属性设置。style 的设置信息如下。

(1) style="? android:attr/progressBarStyleLarge"，大尺寸的环形进度条。

(2) style="? android:attr/progressBarStyleSmall"，小尺寸的环形进度条。

(3) 不设置，模式尺寸的环形进度条。

(4) style="? android:attr/progressBarStyleHorizontal"，水平进度条。

或者采用如下的方式设置进度条样式。

(1) style="@android:style/Widget.ProgressBar.Small"

(2) style="@android:style/Widget.ProgressBar.Lager"

(3) style="@android:style/Widget.ProgressBar.Horizontal"

进度条常用于执行时间较长的代码块中，比如下载、更新等操作，可以给用户心理上的安慰。但是在 Android 中进度条的操作稍微麻烦一点，因为 Android 不允许子线程修改 Activity 的界面，而进度条都是由子线程控制的。Android 给出的解决办法是使用 Handler 类，处理子线程中需要更新 UI 的操作。关于 Handler 的深度介绍请阅读后续章节。

对于水平进度条，经常使用的属性有：

(1) android:max="100"，设置进度条的最大值。

(2) android:progress=10，设置第一层进度条的初始值。

(3) android:secondaryProgress="80"，设置第二层进度条的初始值。

进度条的常用方法如下。

(1) int getMax()：返回这个进度条的范围的最大值。

(2) int getProgress()：返回当前进度。

(3) int getSecondaryProgress()：返回当前次要进度。

(4) void incrementProgressBy(int diff)：指定增加的进度，每次推进的步伐。

(5) boolean isIndeterminate()：指示进度条是否在不确定模式下。

(6) void setIndeterminate(boolean indeterminate)：设置不确定模式下，用于无法确定时间的任务。

(7) void setVisibility(int v)：设置该进度条是否可视。

代码 03-6 演示如何使用 Handler 类在子进程中更新 UI，完整代码请参考随书配套资料 Part03 项目，MainActivity.java 文件，布局文件是 layoutprogressbar.xml。

**代码 03-6**

```
ProgressBar pb1,pb2;
Handler handler = new Handler();
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    //测试进度条
    setContentView(R.layout.layoutprogressbar);
    pb1 = (ProgressBar) findViewById(R.id.progressBar1);
    pb2 = (ProgressBar) findViewById(R.id.progressBar4);
    changeProgress();
}
private void changeProgress() {
    Thread t = new Thread(new MyThread());
    t.start();
}
class MyThread implements Runnable{
    @Override
    public void run() {
        while(pb2.getProgress()< pb2.getMax()){
            handler.post(new Runnable(){
                @Override
                public void run() {
                    pb2.setProgress(pb2.getProgress() + 2);
                    pb2.setSecondaryProgress(pb2.getSecondaryProgress() + 10);
                    if(pb2.getSecondaryProgress()> = pb2.getMax())pb2.setSecondaryProgress(0);
                }
            });
```

```
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        }
    }
```

上述代码将修改进度条的功能放在 handler.post(匿名线程)中完成,由 handler 交给(post)Activity UI 的主线程更新进度。三个环形进度条没有处理进度,都是默认为 indeterminate 模式,在这种模式下,进度条将一直循环,等延时代码执行结束时,将进度条隐藏即可。代码运行效果如图 3.10 所示。


图 3.10 ProgressBar 运行效果

在 Android 中使用子线程需要注意以下事项。

- 一个 Android 应用程序的 UI 由一个主线程维护,不允许子线程修改。
- 子线程不能直接更新 Activity 中的 UI,需要使用 Handler 通知主线程修改 UI。
- Handler 的详细解释在后续章节,此处暂不深究。

### 3.2.8 SeekBar

拖动条是进度条的间接子类,拥有进度条的所有属性和方法,支持滑块的推动。代码 03-7 演示如何使用 SeekBar,完整代码请参考随书配套资料 Part03 项目,MainActivity.java 文件,布局是 layoutseekbar.xml。

**代码 03-7**

```
SeekBar sbar;
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //测试拖动条
    setContentView(R.layout.layoutseekbar);
   tv = (TextView) findViewById(R.id.tv_sb);
   sbar = (SeekBar) findViewById(R.id.seekBar1);
   //sbar 注册监听器
```

```
sbar.setOnSeekBarChangeListener(new OnSeekBarChangeListener(){
      @Override
      public void onProgressChanged(SeekBar seekBar, int progress,
               boolean fromUser) {
          tv.setText("当前进度: " + progress);
      }
      @Override
      public void onStartTrackingTouch(SeekBar seekBar) {

      }
      @Override
      public void onStopTrackingTouch(SeekBar seekBar) {

      }}
 );
```

SeekBar 拥有 OnSeekBarChangeListener 监听器，当滑块滑动时执行监听器中的对应方法。onStartTrackingTouch 方法和 onStartTrackingTouch 方法对应滑动的开始和结束，onProgressChanged 方法对应进度值的改变。属性 android:thumb="@drawable/sb0" 可以设定滑块为指定图片，该图片最好是 png 格式(支持透明效果)。代码运行效果如图 3.11 所示。


图 3.11　SeekBar 运行效果

### 3.2.9　Spinner

Spinner 是类似于下拉列表的一种控件，用户可以从中选择相应选项。Spinner 中的数据需要使用 Adapter(适配器)填充，适配器在后续章节中介绍。代码 03-8 演示如何使用 Spinner 控件，完整代码请参考 Part03 项目，MainActivity.java 文件，布局是 layoutspinner.xml。

**代码 03-8**

```
Spinner spinner;
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //测试 Spinner
    setContentView(R.layout.layoutspinner);
    spinner = (Spinner) findViewById(R.id.spinner1);
    tv = (TextView) findViewById(R.id.tv_spinner);
    spinner.setPrompt("请选择三国人物: "); //
    final List list  = new ArrayList();
    list.add("郭嘉");
    list.add("诸葛亮");
    list.add("周瑜");
```

```
        list.add("司马懿");
        list.add("陆逊");
         ArrayAdapter adapter  = new ArrayAdapter(this, android.R.layout.simple_spinner_item,
    list);
        spinner.setAdapter(adapter);
        spinner.setOnItemSelectedListener(new OnItemSelectedListener(){
            @Override
            public void onItemSelected(AdapterView<?> arg0, View arg1,
                    int arg2, long arg3) {
                tv.setText("已选择: " + arg2 + " : " + arg3);
                tv.append("\n" + list.get(arg2));
            }
            @Override
            public void onNothingSelected(AdapterView<?> arg0) {

            }}
        );
```

上述代码采用 ArrayAdapter 适配器，将数据填充到 Spinner 中，并添加了 OnItemSelectedListener 监听器，处理下拉列表中元素的选择。代码运行效果如图 3.12 所示。


图 3.12　Spinner 运行效果

### 3.2.10　QuickContactBadge

快速联系人标记，该控件用于快速查找与指定条件匹配的联系人信息。需要开启访问联系人的权限，在 AndroidManifest.xml 文件中增加权限：

```
<uses-permission android:name="android.permission.READ_CONTACTS"/>
```

代码 03-9 演示如何使用 QuickContactBadge 控件，完整代码参考 Part03 项目，MainActivity.java 文件，布局文件 layoutquickcontactbadge.xml，应用程序配置文件需要增加读取联系人的权限。

**代码 03-9**

```
QuickContactBadge qcb;
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    //测试 QuickContactBadge
    setContentView(R.layout.layoutquickcontactbadge);
    qcb = (QuickContactBadge) findViewById(R.id.quickContactBadge1);
    qcb.assignContactFromEmail("test@163.com", true);
    qcb.assignContactFromPhone("13902208866", true);
    qcb.setMode(QuickContact.MODE_SMALL);
}
```

该段代码如果要运行成功，需要在虚拟机中新建联系人信息，指明电话号码 1390220×××，邮件 test@163.com。代码运行效果如图 3.13 所示。

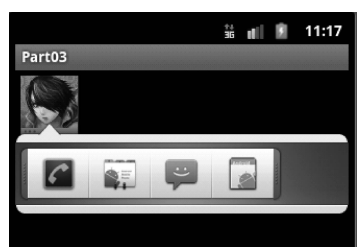


图 3.13　QuickContactBadge 运行效果

### 3.2.11　RatingBar

RatingBar 是基于 SeekBar 和 ProgressBar 的扩展，用星形来显示等级评定。使用 RatingBar 的默认尺寸时，用户可以触摸/拖动或使用键来设置评分。另外两种样式是小尺寸的 ratingBarStyleSmall 和大尺寸的 ratingBarStyleIndicator，其中大尺寸时只适合指示，不能用于用户交互。

(1) style="? android:attr/ratingBarStyleSmall"，指定为小尺寸 RatingBar。

(2) style="? android:attr/ratingBarStyleIndicator"，指定为大尺寸 RatingBar。

代码 03-10 演示 RatingBar 的使用，完整代码参考 Part03 项目，MainActivity.java 文件，布局是 layoutratingbar.xml。

**代码 03-10**

```
RatingBar rb1,rb2,rb3;
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //测试 RatingBar
    setContentView(R.layout.layoutratingbar);
```

```
        rb1 = (RatingBar) findViewById(R.id.ratingBar1);
        rb2 = (RatingBar) findViewById(R.id.ratingBar2);
        rb3 = (RatingBar) findViewById(R.id.ratingBar3);
        tv = (TextView) findViewById(R.id.tv_rb);
        b1 = (Button) findViewById(R.id.bt_rb);
        b1.setOnClickListener(new OnClickListener(){
            @Override
            public void onClick(View arg0) {
                tv.setText("评价结果：");
                tv.append("\n 商品质量：" + rb1.getProgress());
                tv.append("\n 服务态度：" + rb2.getProgress());
                tv.append("\n 物流速度：" + rb3.getProgress());
            }
        });
    }
```

RatingBar 的默认最大值是 10，可以通过 android:max="100"属性修改。默认尺寸的 RatingBar 可以与用户交互，支持单击或拖动。代码运行效果如图 3.14 所示。


图 3.14　RatingBar 运行效果

## 3.3　TextFields

在 Graphic Layout 视图中展开 Text Fields 列表，共有 15 个文本输入框，用于完成不同类型文本内容的输入，如 Plain Text(普通文本框)、Password(密码输入框)、E-mail(电子邮箱输入框)、Postal Address(通信地址输入框)、Number(数字输入框)等。这些输入框都对应 Android Widget 包中的 EditText 控件，上述诸多输入类型的划分由该控件的属性 android:inputType 指定。该属性的取值可以影响虚拟键盘的输入模式，比如作为数字输入框时，虚拟键盘会自动切换为数字输入模式，所以该属性对 EditText 控件尤为重要，可以在用户输入时减少切换输入法的操作。android: inputType 属性常用取值与说明见表 3.2。

表 3.2　inputType 属性常用取值与说明

| android：inputType 取值 | 说明 | 取值类型 |
|---|---|---|
| android:inputType="number" | 输入内容为数字，虚拟键盘为切换为数字输入模式 | 数值 |
| android:inputType="numberDecimal" | 输入内容为带小数点的浮点格式 | |
| android:inputType="phone" | 输入内容为电话号码，虚拟键盘切换为拨号键盘 | |
| android:inputType="datetime" | 输入内容为时间日期 | |
| android:inputType="date" | 输入内容为日期键盘 | |
| android:inputType="textCapWords" | 输入文本首字母大写 | 字符串 |
| android:inputType="textCapSentences" | 输入文本时仅第一个字母大写 | |
| android:inputType="textAutoCorrect" | 输入文本时自动更正 | |
| android:inputType="textAutoComplete" | 输入文本时自动完成 | |
| android:inputType="textMultiLine" | 可以多行输入 | |
| android:inputType="textUri" | 输入内容为网址，虚拟键盘会显示 www、com 等内容 | |
| android:inputType="textEmailAddress" | 输入内容电子邮件地址，虚拟键盘会显示@符号 | |
| android:inputType="textPostalAddress" | 输入内容为地址 | |
| android:inputType="textPassword" | 密码输入框 | |
| android:inputType="textVisiblePassword" | 可见密码输入框 | |

EditText 控件是 TextView 控件的子类，是具有输入编辑功能的文本框。除了 inputType 属性，常用的属性还有 android:digits、android:editable、android:lines 等，详细描述请参考表 3.3。

表 3.3　EditText 控件的常用属性

| 属性 | 说　明 |
|---|---|
| android:digits | 设置 EditText 所接受的字符，如 1234567890 |
| android:editable | 设置 EditText 是否可以编辑，取值为 true 或 false |
| android:gravity | 文本的对齐方式 |
| android:hint | 文本框没有输入内容时显示的提示信息 |
| android:maxLength | 设置文本最大长度 |
| android:lines | 设置文本框显示的行数 |
| android:singleLine | 设置是否单行 |
| android:textSize | 文字大小，单位建议采用 sp |

## 3.4　布局管理器

Android 布局管理器都继承自 ViewGroup 类，用于存放其他控件或嵌套其他布局。常用的布局管理器有 5 个，分别是 LinearLayout（线性布局）、RelativeLayout（相对布局）、TableLayout（表格布局）、FrameLayout（帧布局）和 AbsoluteLayout（绝对布局）。

### 3.4.1　LinearLayout

线性布局管理器对存放其中的控件或布局采用线性方式管理，通过属性 android:

orientation 设定线性的方向，取值 vertical 为竖直方向线性，取值 horizontal 为水平线性。线性布局的 android：gravity 属性可以设定其中元素的对齐方式。

代码 03-11 演示如何使用线性布局设计登录 UI 界面，完整代码请参考项目 Part03，布局文件 testlinearlayout. xml。运行效果如图 3.15 所示。

**代码 03-11**

```
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:orientation = "vertical" >
    <LinearLayout
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:orientation = "horizontal"
        >
        <TextView
            android:id = "@ + id/textView1"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "账号： "
            android:textAppearance = "?android:attr/textAppearanceMedium" />
        <EditText
            android:id = "@ + id/editText1"
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
            android:layout_weight = "1" >
            <requestFocus />
        </EditText>
    </LinearLayout>
    <LinearLayout
        android:id = "@ + id/linearLayout1"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content" >
        <TextView
            android:id = "@ + id/textView2"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "密码： "
            android:textAppearance = "?android:attr/textAppearanceMedium" />
        <EditText
            android:id = "@ + id/editText2"
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
            android:layout_weight = "1"
            android:inputType = "textPassword" />
    </LinearLayout>
    <LinearLayout
        android:id = "@ + id/linearLayout2"
```

```
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
            android:gravity = "right"
             >

            < Button
                android:id = "@ + id/button1"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:text = "登录" />
            < Button
                android:id = "@ + id/button2"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:text = "注册" />
        </LinearLayout >
    </LinearLayout >
```

上述代码中使用 LinearLayout 竖直线性布局管理器，嵌入三个水平线性布局管理器，并将最后一个线性布局管理器的 android:gravity 属性设置为右对齐。

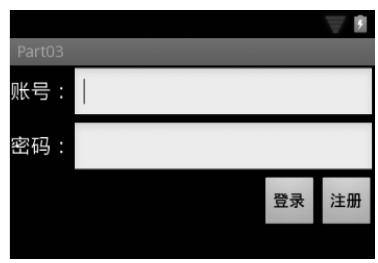


图 3.15　线性布局界面效果图

### 3.4.2　RelativeLayout

相对布局管理器使用元素的相对参考位置布局，在容器中的子元素可以使用彼此之间的相对位置或者与容器之间的相对位置进行定位，比如在哪个元素的哪个方位，与哪个元素采取何种对齐方式。RelativeLayout 中常用的属性有三类，第一类是属性值必须为 id 引用；第二类是属性值为 true 或 false；第三类是属性值为具体像素值，具体解释参考表 3.4。

**表 3.4　RelativeLayout 常用属性介绍**

| 属　　性 | 作　　用 | 备　　注 |
|---|---|---|
| android:layout_below | 在某元素的下方 | 属性值为其他控件的 id 引用，形如 @ id/id-name |
| android:layout_above | 在某元素的上方 | |
| android:layout_toLeftOf | 在某元素的左边 | |
| android:layout_toRightOf | 在某元素的右边 | |
| android:layout_alignTop | 本元素的上边缘和某元素的上边缘对齐 | |
| android:layout_alignLeft | 本元素的左边缘和某元素的左边缘对齐 | |
| android:layout_alignBottom | 本元素的下边缘和某元素的下边缘对齐 | |
| android:layout_alignRight | 本元素的右边缘和某元素的右边缘对齐 | |

续表

| 属　　性 | 作　　用 | 备　　注 |
| --- | --- | --- |
| android:layout_centerHorizontal | 水平居中 | 取值为 true 或 false |
| android:layout_centerVertical | 竖直居中 | |
| android:layout_centerInparent | 相对于父元素完全居中 | |
| android:layout_alignParentBottom | 贴紧父元素的下边缘 | |
| android:layout_alignParentLeft | 贴紧父元素的左边缘 | |
| android:layout_alignParentRight | 贴紧父元素的右边缘 | |
| android:layout_alignParentTop | 贴紧父元素的上边缘 | |
| android:layout_marginBottom | 底边缘的距离 | 取值为具体的像素值，如 5px |
| android:layout_marginLeft | 左边缘的距离 | |
| android:layout_marginRight | 右边缘的距离 | |
| android:layout_marginTop | 上边缘的距离 | |

代码 03-12 演示 RelativeLayout 的使用，完整代码请参考 Part03 项目，testrelativelayout.xml 布局文件，代码的布局效果如图 3.15 所示。

**代码 03-12**

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/r_tv1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:text="账号："
        android:textAppearance="?android:attr/textAppearanceMedium" />
    <EditText
        android:id="@+id/r_et1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/r_tv1"
        android:hint="请输入账号"
        >
    </EditText>
    <TextView
        android:id="@+id/r_tv2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
       android:layout_below="@id/r_et1"
        android:text="密码："
        android:textAppearance="?android:attr/textAppearanceMedium" />
    <EditText
```

```
            android:id = "@ + id/r_et2"
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
            android:layout_toRightOf = "@id/r_tv2"
            android:layout_below = "@id/r_et1"
            android:hint = "请输入密码"
            >
        </EditText>
        <Button
            android:id = "@ + id/r_bt1"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "注册"
            android:layout_below = "@id/r_et2"
            android:layout_alignParentRight = "true"
            />
        <Button
            android:id = "@ + id/r_bt2"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "登录"
            android:layout_below = "@id/r_et2"
            android:layout_toLeftOf = "@id/r_bt1"
            />
    </RelativeLayout>
```

在线性布局中指明元素位置时是以其他元素为参考，需要引用其他控件，可以通过@id/id name的方式实现。相对布局中的控件都应该指明id值，便于其他元素的引用。此外，由于参考控件的不同，相同的界面布局，布局代码也不尽相同。

### 3.4.3　TableLayout

表格布局管理器以行和列的形式对控件进行管理，每一行对应一个TableRow对象或一个View控件。当行为TableRow对象时，可以添加子控件，默认情况下，每个子控件占据一列。当行为View时，该View将独占一行。

TableLayout列数由子元素最多的TableRow确定，元素的个数即为列数。元素的宽高由表格决定，layout_width和layout_height失去原来的作用。常用属性与作用描述见表3.5。

**表3.5　TableLayout常用属性与作用**

| 属　　性 | 作　　用 |
| --- | --- |
| android:stretchColumns | 设置可伸展的列，伸展后使得该行元素充满整行 |
| android:shrinkColumns | 设置可收缩的列，当内容没有占满一行时没有效果 |
| android:collapseColumns | 设置可隐藏的列 |
| android:layout_column | 指定单元格显示在第几列 |
| android:layout_span | 指定单元格跨几列 |

代码 03-13 演示 TableLayout 的使用，完整代码参考 Part03 项目，testtablelayout. xml 布局文件，代码运行效果如图 3.16 所示。

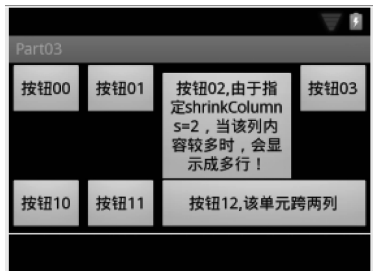


图 3.16　TableLayout 布局效果图

**代码 03-13**

```
<TableLayout
android:layout_width = "fill_parent"
android:layout_height = "wrap_content"
android:shrinkColumns = "2"
>
    <TableRow >
        <Button android:text = "按钮 00"/>
        <Button android:text = "按钮 01"/>
        <Button android:text = "按钮 02,由于指定 shrinkColumns = 2,当该列内容较多时,
                会显示成多行!"/>
        <Button android:text = "按钮 03"/>
    </TableRow >
    <TableRow >
        <Button android:text = "按钮 10"/>
        <Button android:text = "按钮 11"/>
        <Button android:text = "按钮 12,该单元跨两列"
            android:layout_span = "2"
            />
    </TableRow >
    <ImageView
        android:background = "#ffffff"
        android:layout_height = "2px"
        />
</TableLayout >
```

上述代码中的表格是三行(两个 TableRow，一个 ImageView 独占一行)四列(由第 0 行确定，注意表格的行、列都从 0 开始)。属性 android:shrinkColumns="2"指明第 2 列具有收缩特性，其内容过多时会显示为多行，如果去掉该属性，第 2 列以后的内容会被挤压出屏幕可视区。属性 android:layout_span="2"指明该元素所在列占两列宽度。第三行是 ImageView 控件，占据一行的宽度。

TableLayout 不同于 Web 开发时的表格，没有边框线。在使用时注意以下几点。

- TableLayout 中的元素宽度由单元格决定，不能由控件的宽度决定。当指明宽度后，可以配合 android:stretchColumns="*"（所有列都具有伸展特性）达到平均分配各列的宽度，如图 3.17 所示。
- 在行中内容没有充满整行的情况下，被指定为具有伸展特性的列会自动伸展，以充满整行，否则没有效果。


图 3.17　平分整行效果

### 3.4.4　FrameLayout

帧布局是 5 大布局中比较简单的一个，所有放入其中的元素都以左上角为参考，对齐父容器的左边和顶部，后放入的元素将覆盖前面的元素。帧布局常用于显示图片，充当应用程序背景，上面叠加应用程序。代码 03-14 演示 FrameLayout 的使用，完整代码参考 Part03 项目，testframelayout.xml 文件。

**代码 03-14**

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <ImageView
        android:src="@drawable/pic4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="底层"
        android:textAppearance="?android:attr/textAppearanceMedium" />
    <TextView
        android:id="@+id/textView2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="上层"
        android:textSize="56sp"
        />
</FrameLayout>
```

### 3.4.5 AbsoluteLayout

绝对布局以屏幕左上角为坐标原点，水平向右为 $x$ 轴正方向，竖直向下为 $y$ 轴正方向。AbsoluteLayout 使用坐标点定位虽然精确，但代码灵活性较低，在分辨率不同的设备上会显示出不同的效果，因此这种布局一般不建议使用。AbsoluteLayout 常用的属性如下。

(1) android：layout_x，指定控件的 $x$ 坐标。

(2) android：layout_y，指定控件的 $y$ 坐标。

## 3.5 Image 和 Media

### 3.5.1 ImageView 与 BitmapFactory

ImageView 控件主要用于呈现图片，在很多场合都会用到。Android 支持的图片格式有 PNG、JPG 和 GIF 三种，将图片放入 res 文件下的 drawable-×××文件夹中，就可以通过 R.drawable 图片名引用该图片。图片的命名要符合标识符命名规则，且不能使用大写字母。

ImageView 控件的常用属性如下。

(1) android:src，指定 ImageView 显示的图片，可以通过@drawable/图片名，引用项目中的资源。

(2) android:scaleType，图片的放缩模式。可以取以下值。

① matrix，采用用矩阵来绘图，从上到下、由左到右的顺序。

② fitXY，拉伸图片(不按比例)以填充 View 的宽高。

③ fitStart，把图片按比例扩大/缩小到视图的最小边，显示在视图的上部分位置。

④ fitCenter，按比例缩放图片到视图的最小边，居中显示。

⑤ fitEnd，按比例缩放图片到视图的最小边，显示在视图的下部分位置。

⑥ center，按原图大小显示图片，但图片宽高大于 View 的宽高时，中间部分显示。

⑦ centerCrop，按比例缩放图片，使得图片长(宽)大于等于视图的相应维度。

⑧ centerInside，当原图宽高小于或等于 View 的宽高时，按原图大小居中显示；反之将原图缩放至 View 的宽高居中显示。

给 ImageView 控件设置图片，可以直接使用 xml 属性，但大多数情况都是使用代码指定它的图片资源，这就要用到 BitmapFactory 类(工厂模式生产图片)。该类位于 android.graphics 包中，拥有众多静态方法，用于获取不同位置的图片。给 ImageView 的图片可以来源于项目、手机 SD 卡或是直接读取网络上图片。代码 03-15 演示直接给 ImageView 设定图片资源的方式，完整代码请参考 Part03 项目，MainActivity.java 源文件。

**代码 03-15**

```
ImageView iv;
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //测试 ImageView
    setContentView(R.layout.layoutimageview);
    iv = (ImageView) findViewById(R.id.iv);
    //直接设定资源
    iv.setImageResource(R.drawable.pic4);
}
```

上述代码可以在程序运行时执行，设定图片资源，也可以通过代码 03-16 的方式设置图片资源，完整代码参考 Part03 项目，layoutimageview.xml 布局文件。这种方式设置的图片资源会被代码覆盖，所以如果该控件的图片不需要更换，可以直接在布局文件中指明。需要注意的是 android:src 属性和 android:background 属性都可以指定图片，但原理不同，前者是指定图片，位于上层，后者是指定背景，位于下层，且不支持缩放操作。

**代码 03-16**

```
<ImageView
    android:id="@+id/iv"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:src="@drawable/pic17"
    android:background="@drawable/pic6"
    android:scaleType="centerInside"
    />
```

读取项目中的图片资源比较简单，但在应用上有很多的局限性。如果涉及图片资源过多，会造成打包时软件所占空间比较大。当图片比较多时，可以选择存放在 SD 卡中，需要使用时再读取即可。将其他资源创建成图片，需要使用 BitmapFactory 类，该类的方法都是静态的，返回值都是 Bitmap，常用的方法如下。

(1) decodeByteArray(byte[] data, int offset, int length)，从字节数字创建图片。

(2) decodeFile(String pathName)，从文件路径创建图片。

(3) decodeResource(Resources res, int id)，从 res 资源创建图片。

(4) decodeStream(InputStream is)，从输入流创建图片。

代码 03-17 演示使用 BitmapFactory 读取 SD 卡中的图片信息，并设置给 ImageView。完整代码请查看 Part03 项目，MainActivity.java 文件。

**代码 03-17**

```
ImageView iv;
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
```

```
        //测试 ImageView
        setContentView(R.layout.layoutimageview);
        iv = (ImageView) findViewById(R.id.iv);
        //直接设定资源
        //iv.setImageResource(R.drawable.pic4);
        //读取 SD 卡中图片
        FileInputStream fis = null;
        Bitmap img = null;
        if(Environment.getExternalStorageState()!= null){              //如果手机插卡
            File dir  = new File("/sdcard/pic");
            File pic = new File(dir,"pic6.jpg");
            try {
                if(dir.exists() &&pic.exists()){                       //资源存在
                    fis = new FileInputStream(pic);
                    Log.i("print", pic.getAbsolutePath());
                    img = BitmapFactory.decodeFile(pic.getAbsolutePath());
                    //img = BitmapFactory.decodeStream(fis);           //通过输入流构建 Bitmap
                    iv.setImageBitmap(img);
                }
            } catch (FileNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }finally{
                if(fis!= null)
                    try {
                        fis.close();
                    } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
            }
        }
    }
```

以上代码涉及读取 SD 卡的相关知识，具体内容在后续章节会有介绍。读取 SD 卡中图片，既可以通过图片路径，也可以通过输入流。代码 03-17 如果想运行并输出结果，需要在手机虚拟机的 SD 卡中先放入图片。

虚拟机默认不支持 SD 卡，在 Eclipse 菜单下的 Window 中选择 AVD Manager，选中虚拟机，单击 Edit 按钮，打开虚拟机编辑界面，在 HardWare 区域单击 New 按钮，添加新的硬件支持。选择 SD Card Support。这样虚拟机就可以模拟 SD 卡了。

选择菜单 Window→Open Perspective→DDMS 命令，切换到 DDMS 视图。打开 File Explorer 选项卡，如图 3.18 所示。展开 mnt 文件夹，sdcard 文件夹即对应虚拟机的 SD 卡根路径。通过右侧手机符号图标(push a file onto device)，就可以向虚拟机的 SD 卡中存入文件。

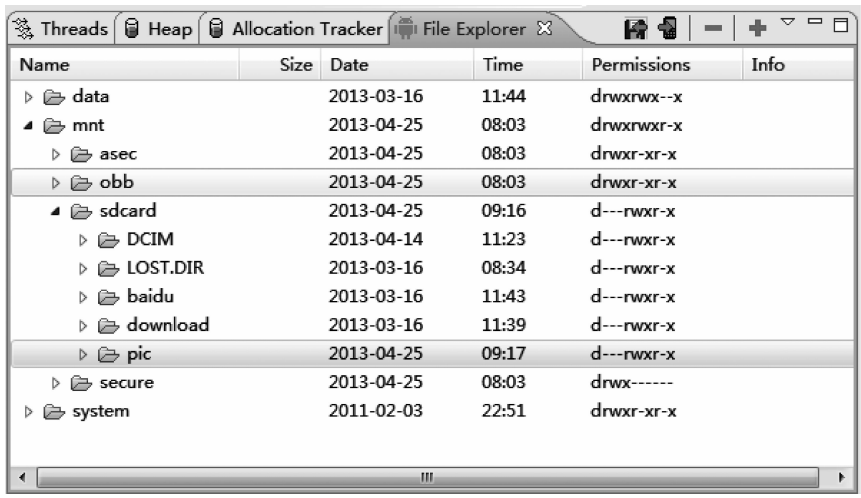


图 3.18　虚拟机文件浏览界面

### 3.5.2　ImageButton

图片按钮直接继承 ImageView，具有按钮的功能，并可以根据需要自定义外观样式。ImageButton 的自定义外观主要有两种实现方式，一种是借助 XML 文件，一种是添加触屏事件。

#### 1. 借助 XML 自定义样式

新建布局文件 layoutimagebutton.xml，如代码 03-18 所示，完整代码查看随书配套资料。ImageButton 的属性 android:src＝"@drawable/btstyle"指明控件采用的样式。此处 btstyle 是 drawable 文件夹中的一个文件，而不是图片。

**代码 03-18**

```
<ImageButton
        android:id="@+id/imageButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#00000000"
        android:src="@drawable/btstyle"
        />
```

btsyle.xml 文件被当作可绘制资源使用。在 drawable 文件夹上单击右键，选择创建 Android XML 文件，此时创建的文件是 drawable 类型，并会自动放入 res 下 drawable 文件夹中。该文件自定义的代码如代码 03-19 所示，定义了按钮被单击、释放、获得焦点和一般情况下的样式。

**代码 03-19**

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="false" android:drawable="@drawable/bt_release" />
    <item android:state_focused="true" android:drawable="@drawable/bt_release" />
    <item android:state_pressed="true" android:drawable="@drawable/bt_press" />
    <item android:drawable="@drawable/bt_release" />
</selector>
```

### 2. 使用触屏事件自定义样式

ImageButton 继承 View，获得了 OnTouchListener(触屏监听器)，实现监听器就可以完成自定义按钮样式。代码 03-20 演示如何使用 ImageButton 的触屏监听器，完整代码参看 Part03 项目，MainActivity.java 源文件。

**代码 03-20**

```
ImageButton ibt;
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //测试 ImageButton
    setContentView(R.layout.layoutimagebutton);
    ibt = (ImageButton) findViewById(R.id.imageButton2);
    ibt.setOnTouchListener(new OnTouchListener(){
        @Override
        public boolean onTouch(View v, MotionEvent event) {
            //Log.i("print", "toch");
            if(event.getAction() == MotionEvent.ACTION_DOWN){
                //触屏按下
                ibt.setImageDrawable(getResources().getDrawable(R.drawable.bt_press));
            }else if(event.getAction() == MotionEvent.ACTION_UP){
                //触屏释放
                ibt.setImageDrawable(getResources().getDrawable(R.drawable.bt_release));
            }
            return false;
        }
    });
}
```

触屏事件在后续章节中会有深入学习。该监听器中 onTouch 方法响应屏幕的操作，包括按下、抬起、滑屏等操作。所有事件都封装在 MotionEvent 类中。

## 3.6 Time 和 Date

### 3.6.1 TimePicker 和 DatePicker

TimePicker 和 DatePicker 都继承自 FrameLayout 类。时间选择控件向用户显示一天中的时间(可以为 24 小时，也可以为 AM/PM 制)，并允许用户进行选择。日期选择控件的

主要功能是向用户提供包含年、月、日的日期数据并允许用户对其修改。

TimePicker 控件通过 OnTimeChangedListener 监听器，处理时间改变事件；DatePicker 控件通过 OnDateChangedListener 监听器，处理日期改变事件。

代码 03-21 演示 TimePicker 与 DatePicker 的使用，完整代码参看 Part03 项目，MainActivity.java 文件，布局文件查看 layouttimedate.xml，运行效果如图 3.19 所示。


图 3.19　时间日期选择器

**代码 03-21**

```
TimePicker tp;
DatePicker dp;
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //测试 TimePicker 和 DatePicker
    setContentView(R.layout.layouttimedate);
    tp = (TimePicker) findViewById(R.id.timePicker1);
    dp = (DatePicker) findViewById(R.id.datePicker1);
    tv1 = (TextView) findViewById(R.id.td_t);
    tv2 = (TextView) findViewById(R.id.td_d);
   //
    tp.setIs24HourView(true);
    tp.setOnTimeChangedListener(new OnTimeChangedListener(){
        @Override
        public void onTimeChanged(TimePicker view, int hourOfDay, int minute) {
            tv1.setText("时间: " + hourOfDay + ":" + minute);
        }
    });
    //
```

```
        dp.init(2013, 8, 13, new OnDateChangedListener(){
            @Override
            public void onDateChanged(DatePicker view, int year,
                    int monthOfYear, int dayOfMonth) {
                tv2.setText("日期: " + year + " - " + (monthOfYear + 1) + " - " + dayOfMonth);
            }}
        );
    }
```

TimePicker 支持 24 小时和 AM、PM 时间格式，可以通过方法 tp. setIs24HourView (true)设置为 24 小时制式。DatePicker 改变事件的监听器需要通过 init 方法传入，并设置初始化的日期。需要注意的是 DatePicker 的月份是从 0 开始的。

### 3.6.2 Chronometer

Chronometer 是 TextView 的子类，可以用 1s 的时间间隔进行计时，并显示出计时结果。Chronometer 类有三个重要的方法：start、stop 和 setBase，其中 start 方法表示开始计时；stop 方法表示停止计时；setBase 方法表示重新计时。此外可以通过 setFormat 方法设置显示时间的格式。

代码 03-22 演示简要计数器的功能，完整代码请参看 Part03 项目，MainActivity. java 文件，布局文件为 layoutchronometer. xml，运行效果如图 3. 20 所示。

**代码 03-22**

```
Chronometer chro;
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    //测试 Chronometer
    setContentView(R.layout.layoutchronometer);

    chro = (Chronometer) findViewById(R.id.chronometer1);
    chro.setFormat("计时: %s"); // %s 会被 Chronometer 的格式替换
    b1 = (Button) findViewById(R.id.c_start);
    b2 = (Button) findViewById(R.id.c_end);
    b3 = (Button) findViewById(R.id.c_reset);
    b1.setOnClickListener(new OnClickListener(){
        @Override
        public void onClick(View v) {
            chro.start();
        }
    });
    b2.setOnClickListener(new OnClickListener(){
        @Override
        public void onClick(View v) {
            chro.stop();
        }
    });
    b3.setOnClickListener(new OnClickListener(){
```

```
        @Override
        public void onClick(View v) {
            chro.setBase(SystemClock.elapsedRealtime());
        }
    });
}
```


图 3.20　计数器运行界面

### 3.6.3　AnalogClock 与 DigitalClock

Android 中的 AnalogClock 与 DigitalClock 都是时钟控件，前者是指针式时钟，后者是数字式时钟。这两个控件的使用比较简单，只需要在布局文件中添加控件，不需要编写代码。新建布局文件，如代码 03-23 所示，完整代码参看 Part03 项目，layoutclock.xml 文件，并将其设置给 MainActivity。

**代码 03-23**

```
<AnalogClock
    android:id="@+id/analogClock1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
<DigitalClock
    android:id="@+id/digitalClock1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="DigitalClock" />
```

AnalogClock 控件继承 View，DigitalClock 继承 TextView，都是已经封装好的控件，只需要在指定位置放置控件即可。代码运行效果如图 3.21 所示。


图 3.21　指针时钟与数字时钟

# 习　题

1. Android 中控件的属性 layout_width 的取值有哪些？各有什么样的作用？

2. 编写一个 Hello World 程序，实现在屏幕上居中（垂直和水平两个方向）显示文本 Hello World，屏幕背景为黑色，字体颜色为白色。

3. 在日志中打印出当前日期和时间，在代码中产生一个异常（空指针、数组越界、除 0 等），捕获该异常并在日志中显示详细的异常信息。

4. 设计并实现数学计算机。

5. 请简要说明 android:stretchColumns 和 android:shrinkColumns 属性的作用。