Broadview

Android 核心技术与实例详解

吴亚峰 索依娜 等编著
百纳科技 审校

Android

核心技术与实例详解

全面覆盖Widget、应用软件、游戏、移动互联网平台、传统互联网服务、位置服务、平台迁移等产品方向；

- 这是一本很功利的开发用书，时刻以创赢为第一要务；
- 国内第一本基于Android 2.0版本的技术书；
- 国内最大Android社区倾力打造；
- Android程序员从业、创业全程指南。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

第 3 章 Android 布局管理器

本章要介绍的内容为 Android 平台下的布局管理器。Android 中的布局包括线性布局、表格布局、相对布局、帧布局和绝对布局。下面将分别对每个布局管理器进行详细的介绍。

3.1 控件类概述

3.1.1 View 类简介

在介绍 Android 的布局管理器之前，有必要让读者了解 Android 平台下的控件类。首先要了解的是 View 类，该类为所有可视化控件的基类，主要提供了控件绘制和事件处理的方法。创建用户界面所使用的控件都继承自 View，如 TextView、Button、CheckBox 等。

关于 View 及其子类的相关属性，既可以在布局 XML 文件中进行设置，也可以通过成员方法在代码中动态设置。View 类常用的属性及其对应方法如表 3-1 所示。

表 3-1 View 类常用属性及对应方法说明

属性名称	对应方法	描　　述
android:background	setBackgroundResource(int)	设置背景
android:clickable	setClickable(boolean)	设置 View 是否响应点击事件
android:visibility	setVisibility(int)	控制 View 的可见性
android:focusable	setFocusable(boolean)	控制 View 是否可以获取焦点
android:id	setId(int)	为 View 设置标识符，可通过 findViewById 方法获取
android:longClickable	setLongClickable(boolean)	设置 View 是否响应长点击事件
android:soundEffectsEnabled	setSoundEffectsEnabled(boolean)	设置当 View 触发点击等事件时是否播放音效
android:saveEnabled	setSaveEnabled(boolean)	如果未作设置，当 View 被冻结时将不会保存其状态
android:nextFocusDown	setNextFocusDownId(int)	定义当向下搜索时应该获取焦点的 View，如果该 View 不存在或不可见，则会抛出 RuntimeException 异常
android:nextFocusLeft	setNextFocusLeftId(int)	定义当向左搜索时应该获取焦点的 View
android:nextFocusRight	setNextFocusRightId(int)	定义当向右搜索时应该获取焦点的 View

续表

属性名称	对应方法	描　　述
android:nextFocusUp	setNextFocusUpId(int)	定义当向上搜索时应该获取焦点的 View，如果该 View

		不存在或不可见，则会抛出 RuntimeException 异常

说明：任何继承自 View 的子类都将拥有 View 类的以上属性及对应方法。

3.1.2　ViewGroup 类简介

另外一个需要了解的是 ViewGroup 类，它也是 View 类的子类，但是可以充当其他控件的容器。ViewGroup 的子控件既可以是普通的 View，也可以是 ViewGroup，实际上，这是使用了 Composite 的设计模式。Android 中的一些高级控件如 Galley、GridView 等都继承自 ViewGroup。

与 Java SE 不同，Android 中并没有设计布局管理器，而是为每种不同的布局提供了一个 ViewGroup 的子类，常用的布局及其类结构如图 3-1 所示。

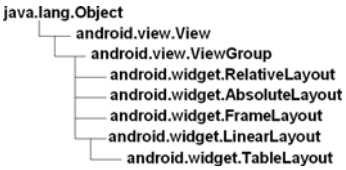

图 3-1　布局管理器的类结构

3.2　线性布局

本节将会对线性布局进行简单的介绍。首先向读者介绍 LinearLayout 类的相关知识，然后通过一个实例说明 LinearLayout 的用法。

3.2.1　LinearLayout 类简介

线性布局是最简单的布局之一，它提供了控件水平或者垂直排列的模型。同时，使用此布局时可以通过设置控件的 weight 参数控制各个控件在容器中的相对大小。LinearLayout 布局的属性既可以在布局文件（XML）中设置，也可以通过成员方法进行设置。表 3-2 给出了 LinearLayout 常用的属性及这些属性的对应设置方法。

表 3-2　LinearLayout 常用属性及对应方法

属性名称	对应方法	描　　述
android:orientation	setOrientation(int)	设置线性布局的朝向，可取 horizontal 和 vertical 两种排列方式
android:gravity	setGravity(int)	设置线性布局的内部元素的布局方式

在线性布局中可使用 gravity 属性来设置控件的对齐方式，gravity 可取的值及说明如表 3-3 所示。

提示：当需要为 gravity 设置多个值时，用“|”分隔即可。

表 3-3　gravity 可取的属性及说明

属 性 值	说　　明
top	不改变控件大小，对齐到容器顶部

续表

属 性 值	说　　明
bottom	不改变控件大小，对齐到容器底部

left	不改变控件大小，对齐到容器左侧
right	不改变控件大小，对齐到容器右侧
center_vertical	不改变控件大小，对齐到容器纵向中央位置
center-horizontal	不改变控件大小，对齐到容器横向中央位置
center	不改变控件大小，对齐到容器中央位置
fill_vertical	若有可能，纵向拉伸以填满容器
fill_horizontal	若有可能，横向拉伸以填满容器
fill	若有可能，纵向横向同时拉伸以填满容器

3.2.2 线性布局案例

在前面的章节中介绍了 LinearLayout 类的相关知识，本节将通过一个案例来说明 LinearLayout 的用法。本案例的开发步骤如下。

① 在 Eclipse 中新建一个项目 Sample_3_1，首先打开项目文件夹下 res/values 目录下的 strings.xml，在其中输入如下代码。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <resources>
3       <string name="app_name">LinearExample</string>
4       <string name="button">按钮</string>
5       <string name="add">添加</string>
6   </resources>
```

代码位置：见随书光盘中源代码/第 3 章/Sample_3_1/res/values 目录下的 strings.xml。

> 说明：在 strings.xml 中主要声明了程序中要用到的字符串资源，这样将所有字符串资源统一管理有助于提高程序的可读性及可维护性。

② 打开项目文件夹下的 res/layout 目录下的 main.xml，将其中已有的代码替换为如下代码。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:orientation="vertical"
4       android:layout_width="fill_parent"
5       android:layout_height="fill_parent"
6       android:id="@+id/lla"
7       android:gravity="right"
8       >                    <!-- 声明一个 LinearLayout 布局，并设置其属性 -->
9       <Button
10          android:text="@string/add"
11          android:id="@+id/Button01"
12          android:layout_width="wrap_content"
13          android:layout_height="wrap_content">
14      </Button>                <!-- 声明一个 Button 布局，并设置其 id 为 Button01 -->
15  </LinearLayout>
```

代码位置：见随书光盘中源代码/第 3 章/Sample_3_1/res/layout 目录下的 main.xml。

- 第 2～8 行声明了一个线性布局，第 3 行设置线性布局的朝向为垂直排列。
- 第 4～5 行设置该线性布局在其所属的父容器中的布局方式为横向和纵向填充父容器。
- 第 6 行为该线性布局声明了 ID。第 7 行设置该线性布局内部元素的布置方式为向右对齐。

- 第 9～14 行声明了一个 Button 控件，其 ID 为 Button01，第 10 行设置 Button 控件显示的文本内容为资源文件 strings.xml 中的属性值。
- 第 12～13 行设置 Button 控件在父容器中的布局方式为只占据自身大小的空间。

③ 打开项目的 Activity 文件 LinearActivity.java，将其中已有的代码替换为如下的代码。

```
1   package wyf.jc;                                          //声明包语句
2   import android.app.Activity;                             //引入相关类
3   import android.os.Bundle;                                //引入相关类
4   import android.view.View;                                //引入相关类
5   import android.widget.Button;                            //引入相关类
6   import android.widget.LinearLayout;                      //引入相关类
7   public class LinearActivity extends Activity {
8        int count=0;                                        //计数器，记录按钮个数
9       @Override
10      public void onCreate(Bundle savedInstanceState) {//重写 onCreate 方法
11         super.onCreate(savedInstanceState);
12         setContentView(R.layout.main);
13         Button button = (Button) findViewById(R.id.Button01);
                                          //获取屏幕中的按钮控件对象
14         button.setOnClickListener(       //为按钮添加 OnClickListener 接口实现
15           new View.OnClickListener(){
16               public void onClick(View v){
17                   LinearLayout ll=(LinearLayout)findViewById(R.id.lla);
                                          //获取线性布局对象
18                   String msg=LinearActivity.this.getResources().getString(R.
                 string.button);
19                   Button tempbutton=new Button(LinearActivity.this);
                                          //创建一个 Button 对象
20                   tempbutton.setText(msg+(++count)); //设置 Button 控件显示的内容
21                   tempbutton.setWidth(80);           //设置 Button 的宽度
22                   ll.addView(tempbutton);            //向线性布局中添加 View
23               }
24           });
25       }
26  }
```

代码位置：见随书光盘中源代码/第 3 章/Sample_3_1/src/wyf/jc 目录下的 main.xml。

- 代码第 8 行声明了用于记录生成的按钮编号的计数器。
- 代码第 13 行通过 findViewById 方法获取屏幕中的 Button 控件对象。
- 代码第 15～24 行为 Button 对象添加了 OnClickListener 监听器的实现。
- 代码第 17～23 行为对 OnClickListener 接口中 onClick 方法的实现，在该方法中首先获得线性布局 LinearLayout 对象的引用，然后创建一个 Button 对象并调用 LinearLayout 对象的 addView 方法将其添加到线性布局容器中。

④ 完成上述三个步骤的工作后，运行项目，在程序中单击“添加”按钮可向屏幕中添加新的按钮，效果图如图 3-2 所示。

图 3-2 为当 LinearLayout 的 orientation 属性为 vertical 时的运行效果，下面来看 orientation 值为 horizontal 时的运行效果，将步骤②中的第 3 行代码改为如下代码。

```
1   android:orientation="horizontal"
```

代码位置：见随书光盘中源代码/第 3 章/Sample_3_1/res/layout 目录下的 main.xml。

运行项目 Sample_3_1，在程序中可以单击“添加”按钮向屏幕中添加新按钮，如图 3-3 所示。

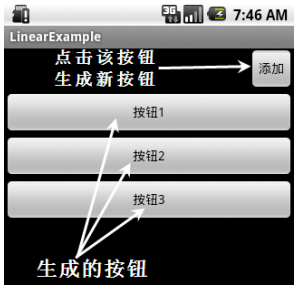

图 3-2　Sample_3_1 运行效果图 1

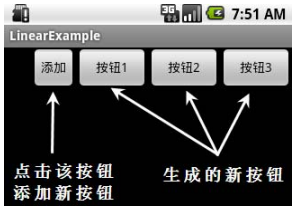

图 3-3　Sample_3_1 运行效果图 2

> 提示：在线性布局中垂直分布时占一列，水平分布时占一行。特别要注意的是，水平或垂直分布时如果超过一行，则不会像 Java SE 中的 FlowLayout 那样自动换行或换列，超出屏幕的子控件将不会被显示，除非将其放到 ScrollView 中。

3.3　表格布局

本节将要介绍的布局管理器是表格布局，首先将对 TableLayout 类进行简单的介绍，然后通过一个案例来说明表格布局的用法。

3.3.1　TableLayout 类简介

TableLayout 类以行和列的形式管理控件，每行为一个 TableRow 对象，也可以为一个 View 对象，当为 View 对象时，该 View 对象将跨越该行的所有列。在 TableRow 中可以添加子控件，每添加一个子控件为一列。

TableLayout 布局中并不会为每一行、每一列或每个单元格绘制边框，每一行可以有 0 或多个单元格，每个单元格为一个 View 对象。TableLayout 中可以有空的单元格，单元格也可以像 HTML 中那样跨越多个列。

在表格布局中，一个列的宽度由该列中最宽的那个单元格指定，而表格的宽度是由父容器指定的。在 TableLayout 中，可以为列设置三种属性。

- Shrinkable，如果一个列被标识为 shrinkable，则该列的宽度可以进行收缩，以使表格能够适应其父容器的大小。
- Stretchable，如果一个列被标识为 stretchable，则该列的宽度可以进行拉伸，以使填满表格中空闲的空间。
- Collapsed，如果一个列被标识为 collapsed，则该列将会被隐藏。

> 注意：一个列可以同时具有 Shrinkable 和 Stretchable 属性，在这种情况下，该列的宽度将任意拉伸或收缩以适应父容器。

TableLayout 继承自 LinearLayout 类，除了继承来自父类的属性和方法， TableLayout 类中还包含表格布局所特有的属性和方法。这些属性和方法说明如表 3-4 所示。

表 3-4　TableLayout 类常用属性及对应方法说明

属性名称	对应方法	描　　述
android:collapseColumns	setColumnCollapsed(int,boolean)	设置指定列号的列为 Collapsed，列号从 0 开始计算
android:shrinkColumns	setShrinkAllColumns(boolean)	设置指定列号的列为 Shrinkable，列号从 0 开始计算
android:stretchColumns	setStretchAllColumns(boolean)	设置指定列号的列为 Stretchable，列号从 0 开始计算

> 说明：setShrinkAllColumns 和 setStretchAllColumns 实现的功能是将表格中的所有列设置为 Shrinkable 或 Stretchable。

3.3.2　表格布局案例

在前面的章节中介绍了TableLayout的相关知识。本节将通过一个案例来说明TableLayout布局管理器的用法，该案例的开发步骤如下。

① 在 Eclipse 中创建一个项目 Sample_3_2。打开项目 res/values 目录下的 strings.xml，在其中输入如下代码。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <resources>
3       <string name="app_name">TableExample</string>
4       <string name="tv1">我自己是一行.....我自己是一行</string>
                                                    <!-- 该值用于独占一行的列 -->
5       <string name="tvShort">我的内容少</string>     <!-- 该值用于内容较少的列 -->
6       <string name="tvStrech">我是被拉伸的一列</string><!-- 该值用于被拉伸的列 -->
7       <string name="tvShrink">我是被收缩的一列被收缩的一列</string>
                                                    <!-- 该值用于被收缩的列 -->
8       <string name="tvLong">我的内容比较长比较长比较长</string>
                                                    <!-- 该值用于内容比较长的列 -->
9   </resources>
```

代码位置：见随书光盘中源代码/第 3 章/Sample_3_2/res/values 目录下的 strings.xml。

- 代码第 4 行声明的字符串对象将会作为独占表格中一行的 TextView 的字符串内容。
- 代码第 5 行声明的字符串对象将会作为表格某行中内容较少的 TextView 的字符串内容。
- 代码第 6 行声明的字符串对象将会作为表格某行中内容较少被拉伸的 TextView 的字符串内容。
- 代码第 7 行声明的字符串对象将会作为表格某行中内容较多被收缩的 Textview 的字符串内容。
- 代码第 8 行声明的字符串对象将会作为表格某行中内容较多的 TextView 的字符串内容。

② 开发程序的布局文件。在本案例中，在布局文件 main.xml 中定义了三个表格，每个表格中只包含一行，各表格的布局示意图如图 3-4 所示。

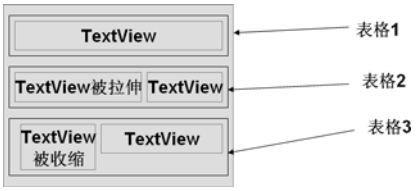

图 3-4　Sample_3_2 中表格布局示意图

下面分别介绍每个表格的内部布局方式。首先打开项目 res/layout 目录下的 main.xml 文件，将其中已有的代码替换为如下代码。

```
1   <?xml version="1.0" encoding="utf-8"?>
```

```
2   <LinearLayout
3       android:id="@+id/LinearLayout01"
4       android:layout_width="fill_parent"
5       android:layout_height="fill_parent"
6       xmlns:android="http://schemas.android.com/apk/res/android"
7       android:orientation="vertical"
8       android:background="@drawable/bbtc"
9       android:gravity="bottom"
10  >                          //声明一个垂直排列的线性布局
11      <TableLayout
12      android:id="@+id/TableLayout01"
13      android:layout_width="fill_parent"
14      android:layout_height="wrap_content"
15      android:background="@color/white"
16      xmlns:android="http://schemas.android.com/apk/res/android">
                               //声明一个表格布局
17      ……//此处省略 TableLayout 的部分代码，将在随后补全
18      </TableLayout>
19      <TableLayout
20      android:id="@+id/TableLayout02"
21      android:layout_width="fill_parent"
22      android:layout_height="wrap_content"
23      android:background="@color/white"
24      android:stretchColumns="0"
25      xmlns:android="http://schemas.android.com/apk/res/android">
                               //声明一个表格布局
26      ……//此处省略 TableLayout 的部分代码，将在随后补全
27      </TableLayout>
28      <TableLayout
29      android:id="@+id/TableLayout03"
30      android:layout_width="fill_parent"
31      android:layout_height="wrap_content"
32      android:background="@color/white"
33      android:collapseColumns="1"
34      android:shrinkColumns="0"
35      xmlns:android="http://schemas.android.com/apk/res/android">
                               //声明一个表格布局
36      ……//此处省略 TableLayout 的部分代码，将在随后补全
37      </TableLayout>
38  </LinearLayout>
```

代码位置：见随书光盘中源代码/第 3 章/Sample_3_2/res/layout 目录下的 main.xml。

- 代码第 2~10 行声明了一个线性布局，在该线性布局中将会垂直摆放三个表格布局，代码第 8 行为该布局设置了背景图片。
- 代码第 11~18 行声明了一个表格布局，该表格布局的 ID 为 TableLayout01，其背景色为白色。
- 代码第 19~28 行声明了一个表格布局，该表格布局的 ID 为 TableLayout02，其背景色为白色，并且对编号为 0 的列设置了 Stretchable 属性。
- 代码第 29~37 行声明了一个表格布局，该表格布局的 ID 为 TableLayout03，其背景色为白色，并且对编号为 1 的列设置了 Collapsed 属性，对编号为 0 的列设置了 Shrinkable 属性。

> 提示：如果需要对多个列设置 Stretchable、Shrinkable 或 Couapsed 属性，需要用逗号隔开每个要设置的列的编号。

下面具体实现每个 TableLayout，首先是 ID 为 TableLayout01 的表格布局的实现代码，如

下所示。

```
1      <TextView
2        android:text="@string/tv1"
3        android:id="@+id/TextView01"
4        android:layout_width="wrap_content"
5        android:layout_height="wrap_content"
6        android:layout_centerInParent="true"
7        android:background="@color/red"
8        android:textColor="@color/black"
9        android:layout_margin="4px"
10     >                      <!--   声明一个 TextView 控件，其 ID 为 TextView01  -->
11     </TextView>
```

代码位置：见随书光盘中源代码/第 3 章/Sample_3_2/res/layout 目录下的 main.xml。

- 代码第 2 行为 TextView 控件设置了显示的内容。
- 代码第 7 行设置 TextView 的背景色为红色。
- 代码第 8 行设置 TextView 的字体颜色为黑色。

在上述表格布局中，表格中只有一行，而在该行声明了一个 TextView 对象占满所有的列。下面介绍 ID 为 TextView02 的表格布局的实现代码，如下所示。

```
1   <TableRow
2            android:id="@+id/TableRow01"
3            android:layout_width="wrap_content"
4            android:layout_height="wrap_content">   <!--  声明一个 TableRow  -->
5               <TextView
6                android:text="@string/tvStrech"
7                android:id="@+id/TextView02"
8                android:layout_width="wrap_content"
9                android:layout_height="wrap_content"
10               android:layout_centerInParent="true"
11               android:background="@color/green"
12               android:textColor="@color/black"
13               android:layout_margin="4px"
14              >                      <!--  声明一个 TextView，ID 为 TextView02  -->
15              </TextView>
16              <TextView
17               android:text="@string/tvShort"
18               android:id="@+id/TextView03"
19               android:layout_width="wrap_content"
20               android:layout_height="wrap_content"
21               android:layout_centerInParent="true"
22               android:background="@color/blue"
23               android:textColor="@color/black"
24               android:layout_margin="4px"
25              >                      <!--  声明一个 TextView，ID 为 TextView03  -->
26              </TextView>
27  </TableRow>
```

代码位置：见随书光盘中源代码/第 3 章/Sample_3_2/res/layout 目录下的 main.xml。

- 代码第 1~4 行声明了一个 TableRow，并设置其在父容器中的布局方式。
- 代码第 5~15 行声明了 TextView，并为其指定了 ID 和在父容器中的布局方式。该 TextView 所占的列被设置了 Stretchable 属性。
- 代码第 16~26 行声明了一个 TextView 控件，并为其指定了 ID。该 TextView 中所显

示的内容较少，所以代码第 5～15 行声明的 TextView 控件会填充该 TextView 剩下的空间。

上述表格布局中总共有一个用 TableRow 声明的行，在该行中包括两列，其中一列被设置了 Stretchable 属性。下面来看 ID 为 TableLayout03 的表格布局的实现代码，如下所示。

```
1   <TableRow  android:id="@+id/TableRow02"
2           android:layout_width="wrap_content"
3           android:layout_height="wrap_content"
4          >                          <!-- 声明了一个 TableRow，ID 为 TableRow02-->
5              <TextView
6               android:text="@string/tvShrink"
7               android:id="@+id/TextView04"
8               android:layout_width="wrap_content"
9               android:layout_height="wrap_content"
10              android:layout_centerInParent="true"
11              android:background="@color/green"
12              android:textColor="@color/black"
13              android:layout_margin="4px"
14             >                       <!-- 声明了一个 TextView，ID 为 TextView04  -->
15             </TextView>
16             <TextView
17              android:text="@string/tvShort"
18              android:id="@+id/TextView05"
19              android:layout_width="wrap_content"
20              android:layout_height="wrap_content"
21              android:layout_centerInParent="true"
22              android:background="@color/blue"
23              android:textColor="@color/black"
24              android:layout_margin="4px"
25             >                       <!--  声明了一个 TableRow，TextView05  -->
26             </TextView>
27             <TextView
28              android:text="@string/tvLong"
29              android:id="@+id/TextView06"
30              android:layout_width="wrap_content"
31              android:layout_height="wrap_content"
32              android:layout_centerInParent="true"
33              android:background="@color/red"
34              android:textColor="@color/black"
35              android:layout_margin="4px"
36             >                       <!--  声明了一个 TableRow，TextView06  -->
37             </TextView>
38  </TableRow>
```

代码位置：见随书光盘中源代码/第 3 章/Sample_3_2/res/layout 目录下的 main.xml。

- 代码第 1~4 行声明了一个 TableRow，并设置了其在父容器中的布局。
- 代码第 5~15 行声明了一个 TextView 控件，该 TextView 控件所占的列被设置了 Shrinkable 属性，可收缩的列将会纵向扩展。
- 代码第 16~26 行声明了一个 TextView 控件，该 TextView 控件所占的列被设置了 Collapsed 属性，所以此 View 将不会被显示。
- 代码第 27~37 行声明了一个 TextView 控件，该 TextView 控件所显示的内容比较长，所以会迫使代码第 5~15 行声明的 TextView 控件所占的列进行收缩。

③ 完成了布局文件 main.xml 的开发之后，最后开发 Activity 部分的代码。打开项目的 Activity 类 TableActivity.java，在其中输入如下代码。

```
1   package wyf.jc;                                          //声明包语句
2   import android.app.Activity;                             //引入相关类
3   import android.os.Bundle;                                //引入相关类
4   public class TableActivity extends Activity {
5      @Override
6      public void onCreate(Bundle savedInstanceState) {     //重写 onCreate 方法
7         super.onCreate(savedInstanceState);
8         setContentView(R.layout.main);   //设置布局文件 main.xml 为当前屏幕
9      }
10  }
```

代码位置：见随书光盘中源代码/第 3 章/Sample_3_2/src/wyf/jc 目录下的 TableActivity.java。

> 说明：TableActivity 的代码比较简单，只是在 onCreate 方法中将当前的屏幕设置为步骤②中开发好的 main.xml 文件。

完成上述步骤的开发之后，下面运行本应用程序，如图 3-5 所示。

在图 3-5 中，第 2 个表格的第 1 列和第 3 个表格的第 1 列分别设置了拉伸和收缩的属性，因此在该行的其他列所显示的内容比较多或比较少时，这些设置了拉伸和收缩属性的列会自动伸长或缩短，以保证表格的宽度不变。

图 3-5　Sample_3_2 运行效果图

3.4　相对布局

本节将要介绍的是相对布局。相对布局比较容易理解，下面首先介绍 RelativeLayout 类的相关知识，然后通过一个案例来说明相对布局的使用。

3.4.1　RelativeLayout 类简介

在相对布局中，子控件的位置是相对兄弟控件或父容器而决定的。出于性能考虑，在设计相对布局时要按照控件之间的依赖关系排列，如 View A 的位置相对于 View B 来决定，则需要保证在布局文件中 View B 在 View A 的前面。

在进行相对布局时用到的属性很多，首先来看属性值只为 true 或 false 的属性，如表 3-5 所示。

表 3-5　相对布局中只取 true 或 false 的属性

属性名称	属性说明
android:layout_centerHorizontal	当前控件位于父控件的横向中间位置
android:layout_centerVertical	当前控件位于父控件的纵向中间位置

续表

属性名称	属性说明
android:layout_centerInParent	当前控件位于父控件的中央位置
android:layout_alignParentBottom	当前控件底端与父控件底端对齐

android:layout_alignParentLeft	当前控件左侧与父控件左侧对齐
android:layout_alignParentRight	当前控件右侧与父控件右侧对齐
android:layout_alignParentTop	当前控件顶端与父控件顶端对齐
android:layout_alignWithParentIfMissing	参照控件不存在或不可见时参照父控件

接下来再来看属性值为其他控件 id 的属性，如表 3-6 所示。

表 3-6　相对布局中取值为其他控件 id 的属性及说明

属性名称	属性说明
android:layout_toRightOf	使当前控件位于给出 id 控件的右侧
android:layout_toLeftOf	使当前控件位于给出 id 控件的左侧
android:layout_above	使当前控件位于给出 id 控件的上方
android:layout_below	使当前控件位于给出 id 控件的下方
android:layout_alignTop	使当前控件的上边界与给出 id 控件的上边界对齐
android:layout_alignBottom	使当前控件的下边界与给出 id 控件的下边界对齐
android:layout_alignLeft	使当前控件的左边界与给出 id 控件的左边界对齐
android:layout_alignRight	使当前控件的右边界与给出 id 控件的右边界对齐

最后要介绍的是属性值以像素为单位的属性及说明，如表 3-7 所示。

表 3-7　相对布局中取值为像素的属性及说明

属性名称	属性说明
android:layout_marginLeft	当前控件左侧的留白
android:layout_marginRight	当前控件右侧的留白
android:layout_marginTop	当前控件上方的留白
android:layout_marginBottom	当前控件下方的留白

需要注意的是在进行相对布局时要避免出现循环依赖，例如设置相对布局在父容器中的排列方式为 WRAP_CONTENT，就不能再将相对布局的子控件设置为 ALIGN_PARENT_BOTTOM。因为这样会造成子控件和父控件相互依赖和参照的错误。

3.4.2　相对布局案例

前面的章节介绍了 RelativeLayout 类的相关知识，本节将会通过一个案例来说明 RelativeLayout 的用法，开发步骤如下。

① 在 Eclipse 中新建一个项目 Sample_3_3，首先进行布局文件 main.xml 的开发。打开 res/layout 目录下的 main.xml，将其中已有的代码替换成如下代码。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <RelativeLayout
3       android:id="@+id/RelativeLayout01"
4       android:layout_width="fill_parent"
5       android:layout_height="fill_parent"
6       xmlns:android="http://schemas.android.com/apk/res/android">
                                                    <!-- 声明一个相对布局 -->
7       <ImageView
```

```
8            android:id="@+id/ImageView01"
9            android:background="@drawable/center"
10           android:layout_width="wrap_content"
11           android:layout_height="wrap_content"
12           android:layout_centerInParent="true"
13       >                                          <!-- 声明一个 ImageView 控件 -->
14       </ImageView>
15       <ImageView
16           android:id="@+id/ImageView02"
17           android:background="@drawable/down"
18           android:layout_width="wrap_content"
19           android:layout_height="wrap_content"
20           android:layout_toRightOf="@id/ImageView01"
21           android:layout_alignTop="@id/ImageView01"
22       >                                          <!-- 声明一个 ImageView 控件 -->
23       </ImageView>
24       <ImageView
25           android:id="@+id/ImageView03"
26           android:background="@drawable/up"
27           android:layout_width="wrap_content"
28           android:layout_height="wrap_content"
29           android:layout_above="@id/ImageView01"
30           android:layout_alignLeft="@id/ImageView01"
31       >                                          <!-- 声明一个 ImageView 控件 -->
32       </ImageView>
33   </RelativeLayout>
```

代码位置：见随书光盘中源代码/第 3 章/Sample_3_3/res/layout 目录下的 main.xml。

- 代码第 2~6 行声明了一个相对布局，声明了其 id 及在父控件中的布局规则。
- 代码第 7~13 行声明了一个 ImageView 控件，并在代码第 12 行设置其位置属性 android:layout_centerInParent 为 true，即该控件位于父控件的中央位置。
- 代码第 15~23 行声明了一个 ImageView 控件，并在代码第 20 和 21 行设置其位置属性 android:layout_toRightOf 和 android:layout_alignTop 均为 ImageView01，即位于 ImageView01 的上方。
- 代码第 24~32 行声明了一个 ImageView 控件并在代码第 29 和 30 行设置其位置属性 android:layout_above 和 android:layout_alignLeft 均为 ImageView01，即位于 ImageView01 的上方。

② 接下来进行 Activity 部分的开发。打开项目的 Activity 文件 RelativeActivity.java，在其中输入如下代码。

```
1   package wyf.jc;                                    //声明包语句
2   import android.app.Activity;                       //引入相关类
3   import android.os.Bundle;                          //引入相关类
4   public class RelativeActivity extends Activity {
5      @Override
6      public void onCreate(Bundle savedInstanceState) {//重写 onCreate 方法
7         super.onCreate(savedInstanceState);
8         setContentView(R.layout.main);               //设置当前屏幕为 main.xml
9      }
10  }
```

代码位置：见随书光盘中源代码/第 3 章/Sample_3_3/src/wyf/jc 目录下的 RelativeActivity.java。

> 说明：Activity 部分的代码比较简单，主要工作是在 onCreate 方法中将 Activity 的当前

> 屏幕设置为 main.xml 布局文件。

完成了上述步骤的开发，下面运行本项目，如图 3-6 所示。

在图 3-6 中，参照控件为屏幕中心的南瓜图片，兔子图片相对于南瓜在其上方，而小猪图片相对于南瓜图片在其右方。

图 3-6　Sample_3_3 运行效果图

3.5　帧布局

本节将要介绍的帧布局是最容易理解的一种布局，在本节的内容中，首先介绍 FrameLayout 类的相关知识，然后开发一个小案例来说明帧布局的用法。

3.5.1　FrameLayout 类简介

FrameLayout 帧布局在屏幕上开辟出了一块区域，在这块区域中可以添加多个子控件，但是所有的子控件都被对齐到屏幕的左上角。帧布局的大小由子控件中尺寸最大的那个子控件来决定。如果子控件一样大，同一时刻只能看到最上面的子控件。

FrameLayout 继承自 ViewGroup，除了继承自父类的属性和方法，FrameLayout 类中包含了自己特有的属性和方法，如表 3-8 所示。

表 3-8　FrameLayout 属性及对应方法

属性名称	对应方法	描　　述
android:foreground	setForeground(Drawable)	设置绘制在所有子控件之上的内容
android:foregroundGravity	setForegroundGravity(int)	设置绘制在所有子控件之上内容的 gravity 属性

> 提示：在 FrameLayout 中，子控件是通过栈来绘制的，所以后添加的子控件会被绘制在上层。

3.5.2　帧布局案例

前面的章节对帧布局 FrameLayout 类进行了简单的介绍。本节将通过一个案例对帧布局的用法进行说明，开发步骤如下。

① 在 Eclipse 中新建一个项目 Sample_3_4。打开其 res/values 目录下的 strings.xml，在其中输入如下代码。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <resources>
3       <string name="app_name">FrameExample</string>
                                          <!-- 声明名为 app_name 的字符串资源  -->
4       <string name="big">大的</string>     <!-- 声明名为 big 的字符串资源  -->
5       <string name="middle">中的</string>  <!-- 声明名为 middle 的字符串资源  -->
6       <string name="small">小的</string>   <!-- 声明名为 small 的字符串资源  -->
7   </resources>
```

代码位置：见随书光盘中源代码/第 3 章/Sample_3_4/res/values 目录下的 strings.xml。

说明：strings.xml 中声明了应用程序总会用到的字符串资源。

② 在项目 rers/values 目录下新建一个 colors.xml，在其中输入如下代码。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <resources>
3       <color name="red">#FF0000</color>        <!--  声明名为 red 的颜色资源 -->
4        <color name="green">#00FF00</color>     <!--  声明名为 green 的颜色资源 -->
5        <color name="blue">#0000FF</color>      <!--  声明名为 blue 的颜色资源 -->
6        <color name="white">#FFFFFF</color>     <!--  声明名为 white 的颜色资源 -->
7   </resources>
```

代码位置：见随书光盘中源代码/第 3 章/Sample_3_4/res/values 目录下的 colors.xml。

说明：colors.xml 中声明了应用程序中将会用到的颜色资源。这样将所有颜色资源统一管理有助于提高程序的可读性及可维护性。

③ 打开项目 res/layout 目录下的 main.xml 文件，将其中已有的代码替换为如下代码。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <FrameLayout
3        android:id="@+id/FrameLayout01"
4        android:layout_width="fill_parent"
5        android:layout_height="fill_parent"
6        android:background="@color/white"
7        xmlns:android="http://schemas.android.com/apk/res/android">
                                              <!--  声明帧布局  -->
8        <TextView
9           android:text="@string/big"
10          android:id="@+id/TextView01"
11          android:layout_width="wrap_content"
12          android:layout_height="wrap_content"
13          android:textSize="60px"
14          android:textColor="@color/green"
15       >                                    <!--  声明一个 TextView 控件  -->
16       </TextView>
17        <TextView
18           android:text="@string/middle"
19           android:id="@+id/TextView02"
20           android:layout_width="wrap_content"
21           android:layout_height="wrap_content"
22           android:textSize="40px"
23           android:textColor="@color/red"
24        >                                    <!--  声明一个 TextView 控件  -->
25        </TextView>
26       <TextView
27          android:text="@string/small"
28          android:id="@+id/TextView03"
29          android:layout_width="wrap_content"
30          android:layout_height="wrap_content"
31          android:textSize="20px"
32          android:textColor="@color/blue"
33       >                                    <!--  声明一个 TextView 控件  -->
34       </TextView>
35  </FrameLayout>
```

代码位置：见随书光盘中源代码/第 3 章/Sample_3_4/res/layout 目录下的 main.xml。

➢ 代码第 2~7 行声明了一个帧布局，并设置其在父控件中的显示方式及自身的背景颜色。

- 代码第 8~16 行声明了一个 TextView 控件，该控件 id 为 TextView01，第 13 行定义了其显示内容的字号为 60px，第 14 行定义了所显示内容的字体颜色为绿色。
- 代码第 17~25 行声明了一个 TextView 控件，该控件 id 为 TextView02，第 22 行定义了其显示内容的字号为 40px，第 23 行定义了所显示内容的字体颜色为红色。
- 代码第 26~34 行声明了一个 TextView 控件，该控件 id 为 TextView03，第 22 行定义了其显示内容的字号为 20px，第 23 行定义了所显示内容的字体颜色为蓝色。

④ 进行 Activity 部分的开发。打开程序的 Activity 文件 FrameActivity.java，在其中输入如下代码。

```
1   package wyf.jc;                                          //声明包语句
2   import android.app.Activity;                             //引入相关类
3   import android.os.Bundle;                                //引入相关类
4   public class FrameActivity extends Activity {
5       @Override
6       public void onCreate(Bundle savedInstanceState) {    //重写 onCreate 方法
7           super.onCreate(savedInstanceState);
8           setContentView(R.layout.main);                   //设置当前屏幕
9       }
10  }
```

代码位置：见随书光盘中源代码/第 3 章/Sample_3_4/src/wyf/jc 目录下的 FrameActivity.java。

> 说明：Activity 部分的代码比较简单，其主要的工作是在 onCreate 方法中将 Activity 的当前屏幕设置为 main.xml 布局文件。

完成了上述步骤的开发后，运行 Sample_3_4，其效果如图 3-7 所示。

在图 3-7 中可以看到，程序运行时所有的子控件都自动地对齐到容器的左上角，由于子控件的 TextView 是按照字号从大到小排列的，所以字号小的在最上层。

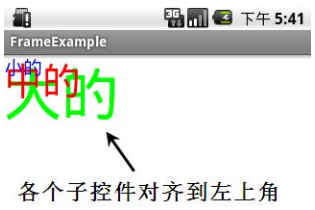

图 3-7　Sample_3_4 运行效果图

3.6　绝对布局

本节要介绍的绝对布局是一种用起来比较费时的布局管理器。首先介绍 AbsoluteLayout 类的相关知识，然后通过一个案例来说明绝对布局的用法。

3.6.1　AbsoluteLayout 类简介

所谓绝对布局，是指屏幕中所有控件的摆放由开发人员通过设置控件的坐标来指定，控件容器不再负责管理其子控件的位置。由于子控件的位置和布局都通过坐标来指定，因此 AbsoluteLayout 类中并没有开发特有的属性和方法。

3.6.2　绝对布局案例

在前面的章节中对 AbsoluteLayout 进行了简单的介绍，本节将通过一个案例来说明绝对布局的使用方法。该案例的开发步骤如下。

① 在 Eclipse 中新建一个项目 Sample_3_5。打开 res/values 目录下的 strings.xml，在其中输入如下代码。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">AbsoluteExample</string>
                                                <!-- 声明名为 app_name 的字符串资源 -->
    <string name="uid">用户名</string>          <!-- 声明名为 uid 的字符串资源 -->
    <string name="pwd">密 码</string>           <!-- 声明名为 pwd 的字符串资源 -->
    <string name="ok">确定</string>             <!-- 声明名为 ok 的字符串资源 -->
    <string name="cancel">取消</string>         <!-- 声明名为 cancel 的字符串资源 -->
</resources>
```

代码位置：见随书光盘中源代码/第 3 章/Sample_3_5/res/values 目录下的 strings.xml。

② 在项目 res/values 目录下新建一个文件 colors.xml，在其中输入如下代码。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="red">#fd8d8d</color>          <!-- 声明名为 red 的资源 -->
     <color name="green">#9cfda3</color>       <!-- 声明名为 green 的资源 -->
     <color name="blue">#8d9dfd</color>        <!-- 声明名为 blue 的资源 -->
     <color name="white">#FFFFFF</color>       <!-- 声明名为 white 的资源 -->
     <color name="black">#000000</color>       <!-- 声明名为 black 的资源 -->
</resources>
```

代码位置：见随书光盘中源代码/第 3 章/Sample_3_5/res/values 目录下的 colors.xml。

> 说明：colors.xml 中声明了应用程序中将会用到的颜色资源。这样将所有颜色资源统一管理有助于提高程序的可读性和可维护性。

③ 进行布局文件的开发。程序中各个控件的布局如图 3-8 所示。其中 ScrollView 中放置了一个 EditText 子控件。

图 3-8　程序各控件位置示意图

打开项目 src/wyf/jc 目录下的 main.xml，将其中已有的代码替换为如下代码。

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout android:id="@+id/AbsoluteLayout01"
    android:layout_width="fill_parent" android:layout_height="fill_parent"
    android:background="@color/white"
    xmlns:android="http://schemas.android.com/apk/res/android">
                                                <!--  声明一个绝对布局  -->
    <TextView
        android:layout_x="20dip" android:layout_y="20dip"
        android:layout_height="wrap_content" android:layout_width="wrap_content"
        android:id="@+id/TextView01" android:text="@string/uid">
                                                <!-- 声明一个 TextView 控件  -->
     </TextView>
```

```
11      <TextView
12          android:layout_x="20dip" android:layout_y="80dip"
13          android:layout_height="wrap_content" android:layout_width="wrap_content"
14          android:id="@+id/TextView02" android:text="@string/pwd">
                                                <!--  声明一个 TextView 控件 -->
15      </TextView>
16      <EditText
17         android:layout_x="80dip" android:layout_y="20dip"
18         android:layout_height="wrap_content" android:layout_width="180dip"
19         android:id="@+id/EditText01">        <!-- 声明一个 EditText 控件  -->
20      </EditText>
21      <EditText
22         android:layout_x="80dip" android:layout_y="80dip"
23         android:layout_height="wrap_content" android:layout_width="180dip"
24         android:id="@+id/EditText02" android:password="true"
25      >                                       <!--  声明一个 EditText 控件 -->
26      </EditText>
27       <Button
28          android:layout_x="155dip" android:layout_y="140dip"
29          android:layout_height="wrap_content" android:id="@+id/Button01"
30          android:layout_width="wrap_content" android:text="@string/ok"
31          >                                   <!-- 声明一个 Button 控件  -->
32       </Button>
33       <Button
34          android:layout_x="210dip" android:layout_y="140dip"
35          android:layout_height="wrap_content" android:id="@+id/Button02"
36          android:layout_width="wrap_content" android:text="@string/cancel"
37          >                                   <!-- 声明一个 Button 控件  -->
38       </Button>
39       <ScrollView
40           android:layout_x="10dip" android:layout_y="200dip"
41           android:layout_height="150dip" android:layout_width="250dip"
42           android:id="@+id/ScrollView01">    <!-- 声明一个 ScrollView 控件  -->
43          <EditText
44               android:layout_width="fill_parent" android:layout_height=
"wrap_content"
45               android:id="@+id/EditText03" android:singleLine="false"
46               android:gravity="top" >        <!-- 声明一个 EditText 控件  -->
47          </EditText>
48       </ScrollView>
49   </AbsoluteLayout>
```

代码位置：见随书光盘中源代码/第 3 章/Sample_3_5/res/layout 目录下的 main.xml。

- 代码第 2~5 行声明了一个 AbsoluteLayout，并设置了其在父容器中的显示方式。
- 代码第 6~10 行和第 11~15 行分别声明了用于显示用户名和密码的 TextView 控件，代码第 7 行和第 12 行为设置绝对布局中子控件坐标的代码。
- 代码第 16~20 行和第 21~26 行分别声明了用于输入用户名和密码的 EditText 控件，代码第 17 行和第 22 行为设置绝对布局中 EditText 控件的坐标，代码第 24 行将 android:password 属性设置为 true。
- 代码第 27~32 行和第 33~38 行分别声明了确定和取消按钮控件，其中代码第 28 行和第 34 行设置了其在绝对布局中的坐标。
- 代码第 39~48 行声明了一个 ScrollView 控件，该控件中包含一个 EditText 控件。

④ 开发应用程序的 Activity。打开项目 src/wyf/jc 目录下的 AbsoluteActivity.java，在其中输入如下代码。

```
1   package wyf.jc;                                          //声明包语句
2   import android.app.Activity;                             //引入相关类
3   import android.os.Bundle;                                //引入相关类
4   import android.view.View;                                //引入相关类
5   import android.widget.Button;                            //引入相关类
6   import android.widget.EditText;                          //引入相关类
7   public class AbsoluteActivity extends Activity {
8      @Override
9      public void onCreate(Bundle savedInstanceState) {//重写 onCreate 方法
10        super.onCreate(savedInstanceState);
11        setContentView(R.layout.main);                    //设置当前屏幕
12        final Button OkButton = (Button) findViewById(R.id.Button01);
                                                           //获取确定按钮对象
13        final Button cancelButton = (Button) findViewById(R.id.Button02);
                                                           //获取取消按钮对象
14        final EditText uid=(EditText)findViewById(R.id.EditText01);
                                                           //获取用户名文本框对象
15        final EditText pwd=(EditText)findViewById(R.id.EditText02);
                                                           //获取密码文本框对象
16        final EditText log=(EditText)findViewById(R.id.EditText03);
                                                           //获取登录日志文本框对象
17        OkButton.setOnClickListener(     //为按钮添加 OnClickListener 监听器实现
18              new View.OnClickListener(){
19                  public void onClick(View v){       //重写 onClick 方法
20                      String uidStr=uid.getText().toString();
                                                           //获取用户名文本框的内容
21                      String pwdStr=pwd.getText().toString();
                                                           //获取密码文本框的内容
22                      log.append("用户名："+uidStr+" 密码："+pwdStr+"\n");
23                  }
24              });
25        cancelButton.setOnClickListener( //为按钮添加 OnClickListener 监听器实现
26              new View.OnClickListener(){
27                  public void onClick(View v){       //重写 onClick 方法
28                      uid.setText("");               //清空用户名文本框内容
29                      pwd.setText("");               //清空密码文本框内容
30                  }
31              });
32     }
33  }
```

代码位置：见随书光盘中源代码/第 3 章/Sample_3_5/src/wyf/jc 目录下的 AbsoluteActivity.java。

- 代码第 12~16 行通过 findViewById 方法获取了布局文件中的各个控件对象。
- 代码第 17~24 行为确定按钮添加了 OnClickListener 监听器的实现。在重写的 onClick 方法中，主要进行的工作是将用户名和密码文本框中的内容添加到用于记录登录日志信息的 EditText 的内容中。
- 代码第 25~31 行为取消按钮添加了 OnClickListener 监听器的实现。在重写的 onClick 方法中，主要进行的工作是将用户名和密码文本框中的内容清空。

完成上述步骤的开发之后，运行本程序，在程序界面多次输入登录信息，如图 3-9 所示。

图 3-9　Sample_3_5 运行效果图

3.7　小结

本章主要介绍的内容是在 Android 平台下开发用户界面时使用的几种布局管理器。在介绍每种布局管理器时，都有案例进行辅助说明。本章是学习 Android 用户界面开发比较基础的一章，虽然本章的知识并不是很难，但是对于后面章节的学习是十分有帮助的。