

第2章 *Android*用户界面设计

石刚

2025年3月

2.1 用户界面组件包widget 和View类

1、用户界面组件包widget

- Android系统的大多数用户界面组件均放置在widget包中。

可视化组件	说明
Button	按钮
CalendarView	日历视图
CheckBox	复选框
EditText	可编辑的文本输入框
ImageView	显示图像或图标，并提供缩放、着色等各种图像处理方法
ListView	列表框视图
MapView	地图视图。
RadioGroup	单选按钮组件
Spinner	下拉列表
TextView	文本标签
WebView	网页浏览器视图
Toast	消息提示

2、View类

- **View**是用户界面组件的共同父类，几乎所有的用户界面组件都是继承**View**类而实现的，如**TextView**、**Button**、**EditText**等。
- 对**View**类及其子类的属性进行设置，可以在布局文件**XML**中设置，也可以通过成员方法在**Java**代码文件中动态设置。

View类的常用属性与方法

属 性	对 应 方 法	说 明
android:background	<code>setBackgroundColor(int color)</code>	设置背景颜色
android:id	<code>setId(int)</code>	为组件设置可通过findViewById方法获取的标识符
android:alpha	<code>setAlpha(float)</code>	设置透明度，取值[0, 1]之间
	<code>findViewById(int id)</code>	与id所对应的组件建立关联
android:visibility	<code>setVisibility(int)</code>	设置组件的可见性
android:clickable	<code>setClickable(boolean)</code>	设置组件是否响应单击事件

2.2 文本标签TextView 与按钮Button

2.2.1 文本标签TextView

- 文本标签TextView用于显示文本内容，是最常用的组件之一。

方 法	功 能
getText();	获取文本标签的文本内容
setText(CharSequence text);	设置文本标签的文本内容
setTextSize(float);	设置文本标签的文本大小
setTextColor(int color);	设置文本标签的文本颜色

文本标签**TextView**常用的**XML**文件元素

元 素 属 性	说 明
<code>android:id</code>	文本标签标识
<code>android:layout_width</code>	文本标签 TextView 的宽度，通常取值" <code>fill_parent</code> "（屏幕宽度）或以像素为单位 pt 的固定值
<code>android:layout_height</code>	文本标签 TextView 的高度，通常取值" <code>wrap_content</code> "（文本的高）或以像素 px 为单位的固定值。
<code>android:text</code>	文本标签 TextView 的文本内容
<code>android:textSize</code>	文本标签 TextView 的文本大小

【例2-1】设计一个文本标签组件程序。

创建名称为**Ex2_01**的新项目，包名为**com.ex2_01**。打开系统自动生成的项目框架，需要设计的文件为：

- (1) 设计布局文件**activity_main.xml**;
- (2) 设计控制文件**MainActivity.java**;
- (3) 设计资源文件。

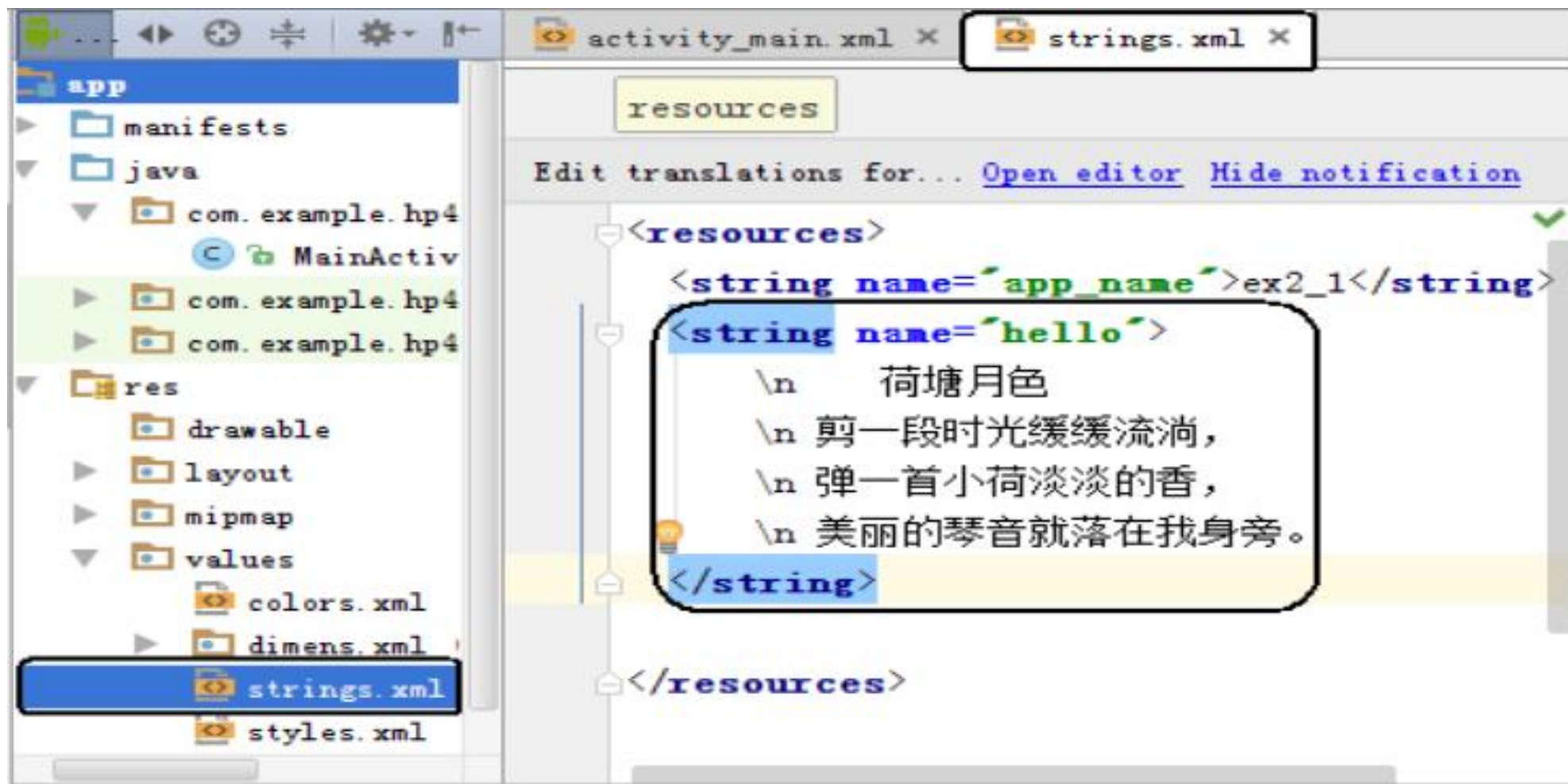
(1) 设计界面布局文件activity_main.xml

在界面布局文件activity_main.xml中加入文本标签组件TextView, 设置文本标签组件的id属性。

```
<!--设置文本标签的属性值-->
```

```
<TextView  
    android:id="@+id/textView1 "  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:text="@string/hello" />
```

在界面布局中设置文本标签



(2) 界面布局文件activity_main.xml



(3) 设计控制文件MainActivity.java

在控制文件MainActivity.java源文件在添加文本标签组件，并将布局文件中所定义的文本标签元素属性值赋值给文本标签，与布局文件中文本标签建立关联。

```
public class MainActivity extends Activity
{
```

```
    private TextView txt;
```

← 声明文本标签对象

```
    public void onCreate(Bundle savedInstanceState)
```

```
    {
```

```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.main);
```

```
        txt = (TextView)findViewById(R.id.textView1);
```

← 与布局文件文本
标签建立关联

```
        txt.setTextColor(Color.WHITE);
```

← 设置文本颜色

```
    }
```

```
}
```



2.2.2 按钮Button及按钮处理事件

- 按钮Button用于处理人机交互的事件，在一般应用程序中常常会用到。由于按钮Button是文本标签TextView的子类，按钮Button继承了文本标签TextView所有的方法和属性。其继承关系如图所示。

```
java.lang.Object
├── android.view.View
│   ├── android.widget.TextView
│       └── android.widget.Button
```

- 按钮Button在程序设计中最常用的方式是实现OnClickListener监听接口，当单击按钮时，通过OnClickListener监听接口触发onClick（）事件，实现用户需要的功能。
- OnClickListener接口有一个onClick（）方法，在按钮Button实现OnClickListener接口时，一定要重写这个方法。
- 按钮Button调用OnClickListener接口对象的方法如下：
按钮对象. setOnClickListener(OnClickListener对象);

【例2-2】 编写程序，当点击按钮命令后，页面标题及文本组件的文字内容发生变化。



(点击按钮前)



(点击按钮后)

(1) 设计布局文件: **activity_main.xml**
在布局文件中添加一个按钮, 其id设置为**button1**。

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="fill_parent"
4     android:layout_height="fill_parent"
5     android:orientation="vertical" >
6     <TextView
7         android:id="@+id/textView1"
8         android:layout_width="fill_parent"
9         android:layout_height="wrap_content"
10        android:text="@string/hello" />
11    <Button
12        android:id="@+id/button1"
13        android:layout_width="fill_parent"
14        android:layout_height="wrap_content"
15        android:text="@string/button" />
16 </LinearLayout>
```

(2) 控制文件: Ex2_2Activity.java

在控制文件Ex2_2Activity.java中, 设计一个实现按钮监听接口的内部类mClick, 当点击按钮时, 触发onClick()事件。

```
1 package com.ex2_2;  
2 import android.app.Activity;  
3 import android.os.Bundle;  
4 import android.view.View;  
5 import android.view.View.OnClickListener;  
6 import android.widget.TextView;  
7 import android.widget.Button;  
8
```

```
9 public class Ex2_2Activity extends Activity
10 {
11     private TextView txt;
12     private Button btn;
13     public void onCreate(Bundle savedInstanceState)
14     {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.main);
17         txt = (TextView) findViewById(R.id.textView1);
18         btn = (Button) findViewById(R.id.button1);
19         btn.setOnClickListener(new mClick());
20     }
21     class mClick implements OnClickListener
22     {
23         public void onClick(View v)
24         {
25             Ex2_2Activity.this.setTitle("改变标题");
26             txt.setText(R.string.newStr);
27         }
28     }
29 }
```

(3) 资源文件: strings.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3   <string name="hello">Hello World, 这是Ex03_02的界面!</string>
4   <string name="app_name">Ex2_2</string>
5   <string name="button">点击我! </string>
6   <string name="newStr">改变了文本标签的内容</string>
7 </resources>
```

【例2-3】编写程序，点击按钮命令，改变文本组件的文字及背景颜色。如图所示。



（点击按钮前）



（点击按钮后）

- 本例题涉及到颜色定义，Android系统在android.graphics.Color里定义了12种常见的颜色常数，其颜色常数见表2-5。

颜色常数	十六进制数色码	意 义
Color.BLACK	0xff000000	黑色
Color.BLUE	0xff00ff00	蓝色
Color.CYAN	0xff00ffff	青绿色
Color.DKGRAY	0xff444444	灰黑色
Color.GRAY	0xff888888	灰色
Color.GREEN	0xff0000ff	绿色
Color.LTGRAY	0xffcccccc	浅灰色
Color.MAGENTA	0xffff00ff	红紫色
Color.RED	0xffff0000	红色
Color.TRANSPARENT	0x00ffffff	透明
Color.WHITE	0xffffffff	白色
Color.YELLOW	0xffffffff00	黄色

创建名称为Ex2_3的新项目，包名为com.ex2_3。

(1) 设计布局文件main.xml

在XML文件中表示颜色的方法有多种：

- #RGB：用三位十六进制数分别表示红、绿、蓝颜色。
- #ARGB：用四位十六进制数分别表示透明度、红、绿、蓝颜色。
- #RRGGBB：用六位十六进制数分别表示红、绿、蓝颜色。
- #AARRGGBB：用八位十六进制数分别表示透明度、红、绿、蓝颜色。


```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   android:layout_width="fill_parent"
4   android:layout_height="fill_parent"
5   android:background="#ff7f7c"
6   android:orientation="vertical" >
7   <TextView
8     android:id="@+id/textView1"
9     android:layout_width="fill_parent"
10    android:layout_height="wrap_content"
11    android:textColor="#ff000000"
12    android:text="@string/hello" />
13   <Button
14     android:id="@+id/button1"
15     android:layout_width="wrap_content"
16     android:layout_height="wrap_content"
17     android:text="@string/button" />
18 </LinearLayout>
```

采用八位十六进制数表示颜色

(2) 控制文件: Ex2_3Activity.java

```
1 package com.ex2_3;  
2 import android.app.Activity;  
3 import android.graphics.Color;  
4 import android.os.Bundle;  
5 import android.view.View;  
6 import android.view.View.OnClickListener;  
7 import android.widget.Button;  
8 import android.widget.TextView;  
9
```

```
10 public class Ex2_3Activity extends Activity
11 {
12     /** Called when the activity is first created. */
13     private TextView txt;
14     private Button btn;
15     @Override
16     public void onCreate(Bundle savedInstanceState)
17     {
18         super.onCreate(savedInstanceState);
19         setContentView(R.layout.main);
20         btn=(Button)findViewById(R.id.button1);
21         txt=(TextView)findViewById(R.id.textView1);
22         btn.setOnClickListener(new click());
23     }
```

定义实现监听接口的内部类注册监听接口

```
24 class click implements OnClickListener
25 {
26     public void onClick(View v)
27     {
28         int BLACK = 0xffcccccc;
29         txt.setText("改变了文字及背景颜色");
30         txt.setTextColor(Color.YELLOW);
31         txt.setBackgroundColor(BLACK);
32     }
33 }
34 }
```

采用颜色常数设置文字颜色

设置文本标签背景颜色

(3) 资源文件: strings.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3   <string name="hello">
4     Hello World, MainActivity!</string>
4   <string name="app_name">Ex2_3</string>
5   <string name="button">点击我, 改变文字背景颜色</string>
6 </resources>
```

2.3 文本编辑框EditText

- 文本编辑框EditText用于接收用户输入的文本信息内容。文本编辑框EditText继承于文本标签TextView, 其继承关系如图所示。

```
android.view.View
└─ android.widget.TextView
    └─ android.widget.EditText
```

文本编辑框**EditText**主要继承文本标签**TextView**的方法，其常用的方法见表2-6。

方法	功能
EditText(Context context)	构造方法，创建文本编辑框对象
getText()	获取文本编辑框的文本内容
setText(CharSequence text)	设置文本编辑框的文本内容

文本编辑框EditText常用的XML文件元素

元素属性	说明
android:editable	设置是否可编辑，其值为“true”或“false”
android:numeric	设置TextView 只能输入数字，其参数默认值为假
android:password	设置密码输入，字符显示为圆点，其值为“true”或“false”
android:phoneNumber	设置只能输入电话号码，其值为“true”或“false”

定义框EditText元素的android: numeric属性，其取值只能是下列常量（可由“|”连接多个常量）：

- integer 可以输入数值.
- signed 可以输入带符号的数值.
- decimal 可以输入带小数点的数值.

【例2-4】设计一个密码验证程序。



- 创建名称为Ex2_4的新项目，包名为com.ex2_4。

(1) 设计布局文件activity_main.xml

在界面布局中，设置一个编辑框，用于输入密码，再设置一个按钮，判断密码是否正确，设置两个文本标签，其中一个显示提示信息“请输入密码”，另一个用于显示密码正确与否。

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   android:layout_width="fill_parent"
4   android:layout_height="fill_parent"
5   android:orientation="vertical" >
6   <TextView
7     android:id="@+id/myTextView01"
8     android:layout_width="fill_parent"
9     android:layout_height="41px"
10    android:layout_x="33px"
11    android:layout_y="106px"
12    android:text="请输入密码:"
13    android:textSize="24sp"
14  />
```

<!-- 建立一个EditText -->

15 <EditText

16 android:id="@+id/myEditText"

17 android:layout_width="180px"

18 android:layout_height="wrap_content"

19 android:layout_x="29px"

20 android:layout_y="33px"

21 android:inputType="text"

22 android:textSize="24sp" />

<!-- 建立一个Button -->

23 <Button

24 android:id="@+id/myButton"

25 android:layout_width="100px"

26 android:layout_height="wrap_content"

27 android:text="确定"

28 android:textSize="24sp"

29 />

<!--建立一个TextView -->

30 <TextView

31 android:id="@+id/myTextView02"

32 android:layout_width="180px"

33 android:layout_height="41px"

34 android:layout_x="33px"

35 android:layout_y="106px"

36 android:textSize="24sp"

37 />

38 </LinearLayout>

(2) 设计控制文件MainActivity.java

在控制文件MainActivity.java中，主要是设计按钮的监听事件，当单击按钮后，从文本编辑框中获取输入的文本内容，与密码“abc123”进行比较。

```
1 package com.ex2_4;  
2 import android.app.Activity;  
3 import android.os.Bundle;  
4 import android.view.View;  
5 import android.view.View.OnClickListener;  
6 import android.widget.EditText;  
7 import android.widget.TextView;  
8 import android.widget.Button;
```

```
9 public class MainActivity extends Activity
10 {
11     private EditText edit;
12     private TextView txt1,txt2;
13     private Button  mButton01;
14     @Override
15     public void onCreate(Bundle savedInstanceState)
16     {
17         super.onCreate(savedInstanceState);
18         setContentView(R.layout.main);
19         txt1 = (TextView)findViewById(R.id.myTextView01);
20         txt2 = (TextView)findViewById(R.id.myTextView02);
21         edit = (EditText)findViewById(R.id.myEditText);
22         mButton01 = (Button)findViewById(R.id.myButton);
23         mButton01.setOnClickListener(new mClick());
24     }
```


定义实现监听接口的内部类

```
25 class mClick implements OnClickListener
26 {
27     public void onClick(View v)
28     {
29         String passwd;
30         passwd=edit.getText().toString();
31         if(passwd.equals("abc123"))
32             txt2.setText("欢迎进入快乐大本营!");
33         else
34             txt2.setText("非法用户,请立刻离开!");
35     }
36 }
37 }
```

获取文本编辑框中的文本内容

equals()方法比较两个字符串是否相等

2.4 Android布局管理

2.4 Android布局管理

Android系统的布局管理指的是在XML布局文件中设置组件的大小、间距、排列及对齐方式等。

Android系统中常见的布局方式有6种，它们分别是：

LinearLayout、FrameLayout、TableLayout、GridLayout、RelativeLayout、AbsoluteLayout。

2016年推出**Android新布局-ConstraintLayout**

2.4.1 布局文件的规范与重要属性

1. 布局文件的规范

Android 系统应用程序的 XML 布局文件有如下规范：

- (1) 布局文件作为应用项目的资源存放在 `res/layout` 目录下，其扩展名为 `.xml`；
- (2) 布局文件的根节点通常是一个布局方式，在根节点内可以添加组件作为节点；
- (3) 布局文件的根节点必须包含一个命名空间：

`xmlns:android="http://schemas.android.com/apk/res/android"`

- (4) 如果要在实现控制功能的 Java 程序中控制界面中的组件，则必须为界面文件中的组件定义一个 ID，其定义格式为：

`android:id="@+id/<组件ID>"`

2. 布局文件的重要属性值

(1) 设置组件大小的属性值

- `wrap_content` : 根据组件内容的大小来决定组件的大小;
- `match_parent` : 使组件填充所在容器的所有空间。

(2) 设置组件大小的单位

- `px (pixels)` 像素: 即屏幕上的发光点;
- `dp (或 dip device independent pixels)` 设备独立像素: 一种支持多分辨率设备的抽象单位, 和硬件相关;
- `sp (scaled pixels)` 比例像素: 设置字体大小。

(3) 设置组件的对齐方式

在布局文件中，由 “`android:gravity`” 属性控制组件的对齐方式，其属性值有上 (`top`)、下 (`bottom`)、左 (`left`)、右 (`right`)、水平方向居中 (`center_horizontal`)、垂直方向居中 (`center_vertical`) 等。

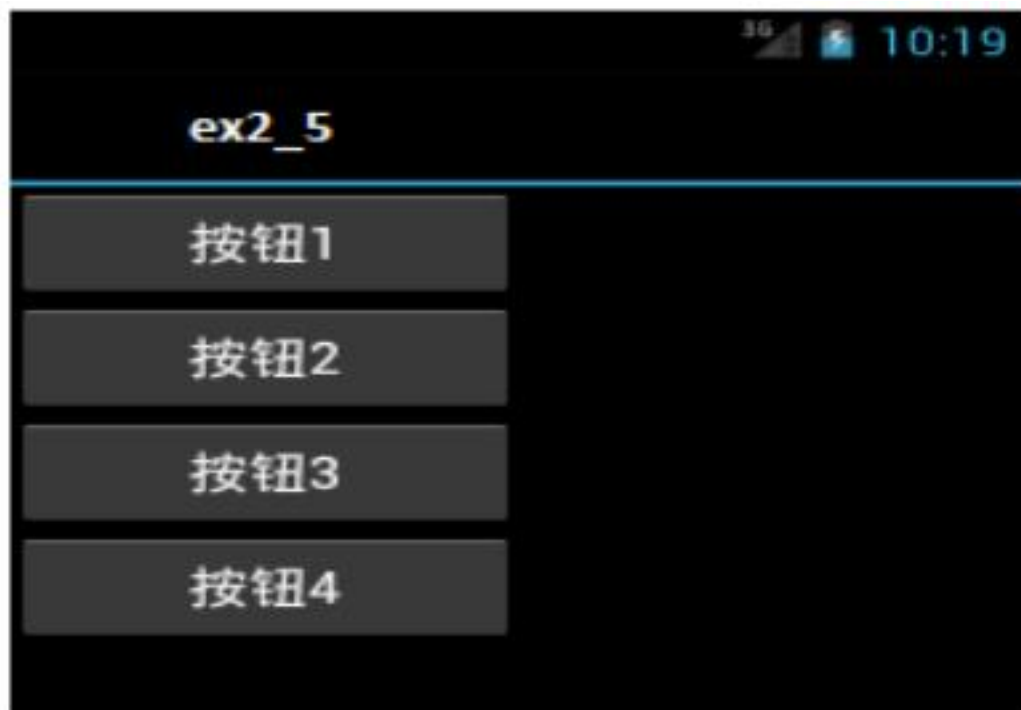
2.4.2 常见的布局方式

1、线性布局LinearLayout

- 线性布局LinearLayout将组件按照水平或垂直方向排列。在XML布局文件中，由根元素LinearLayout来标识线性布局。
- 在布局文件中，由“android:orientation”属性来控制排列方向，其属性值有水平（horizontal）和垂直（vertical）两种：
 - ◆ 设置线性布局为水平方向
android:orientation="horizontal"
 - ◆ 设置线性布局为垂直方向
android:orientation="vertical"

- 在布局文件中，由 “**android:gravity**” 属性控制组件的对齐方式，其属性值有
 - ◆ 上 (**top**) 、
 - ◆ 下 (**bottom**) 、
 - ◆ 左 (**left**) 、
 - ◆ 右 (**right**) 、
 - ◆ 水平方向居中 (**center_horizontal**) 、
 - ◆ 垂直方向居中 (**center_vertical**) 。

【例2-5】线性布局应用示例。



(a) 垂直方向的线性布局



(b) 水平方向的线性布局

线性布局的核心代码:

线性布局的组件垂直排列

```
<LinearLayout android:orientation="vertical" >
```

或: 线性布局的组件水平排列

```
< LinearLayout android:orientation="horizontal" >
```

2、帧布局FrameLayout

- 帧布局FrameLayout是将组件放置到左上角位置，当添加多个组件时，后面的组件将遮盖之前的组件。在XML布局文件中，由根元素FrameLayout来标识帧布局。

【例2-6】帧布局应用示例。

(1) 布局文件activity_main.xml

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

```
<FrameLayout
```

```
xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent">
```

```
    <ImageView
```

```
        android:id="@+id/mImageView"
```

```
        android:layout_width="60px"
```

```
        android:layout_height="wrap_content"
```

```
    />
```

```
    <TextView
```

```
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
```

```
        android:text="快乐大本营"
```

```
        android:textSize="18sp"
```

```
    />
```

```
</FrameLayout>
```



(2) 控制文件MainActivity.java

```
package com.ex2_6;
import android.app.Activity;
import android.os.Bundle;
import android.widget.ImageView;
public class MainActivity extends Activity
{
    ImageView imageview;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        imageview = (ImageView) findViewById(R.id.mImageView);
        imageview.setImageResource(R.drawable.img);
    }
}
```

3、表格布局TableLayout

- 表格布局TableLayout是将页面划分成行列构成的单元格。
- 在XML布局文件中，由根元素TableLayout来标识表格布局。
- 表格的列数由android:shrinkColumns定义，例如，
android:shrinkColumns = “0, 1, 2”，即表格为3列，其列编号为第1、2、3列。
- 表格的行由<TableRow> </TableRow>定义。组件放置到哪一列，由android:layout_column指定列编号。

【例2-7】 表格布局应用示例。设计一个3行4列的表格布局， 组件安排如图所示。

- 创建名称为E2_7的新项目。
- 生成项目框架后， 将准备好的图像文件img1.png、img2.png、img3.png、img4.png、img5.png复制到res\drawable-hdpi目录下。



(1) 表格布局的布局文件activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
<TableRow> <!-- 第1行 -->
    <ImageView android:id="@+id/mImageView1"
        android:layout_width="60px"
        android:layout_height="wrap_content"
        android:layout_column="0" />
    <ImageView android:id="@+id/mImageView2"
        android:layout_width="60px"
        android:layout_height="wrap_content"
        android:layout_column="1" />
</TableRow>
```



```
<TableRow><!-- 第2行 -->
    <ImageView android:id="@+id/mImageView3"
        android:layout_width="60px"
        android:layout_height="wrap_content"
        android:layout_column="1" />
    <ImageView android:id="@+id/mImageView4"
        android:layout_width="60px"
        android:layout_height="wrap_content"
        android:layout_column="2" />
</TableRow>
<TableRow> <!-- 第3行 -->
    <ImageView android:id="@+id/mImageView5"
        android:layout_width="60px"
        android:layout_height="wrap_content"
        android:layout_column="3" />
</TableRow>
</TableLayout>
```

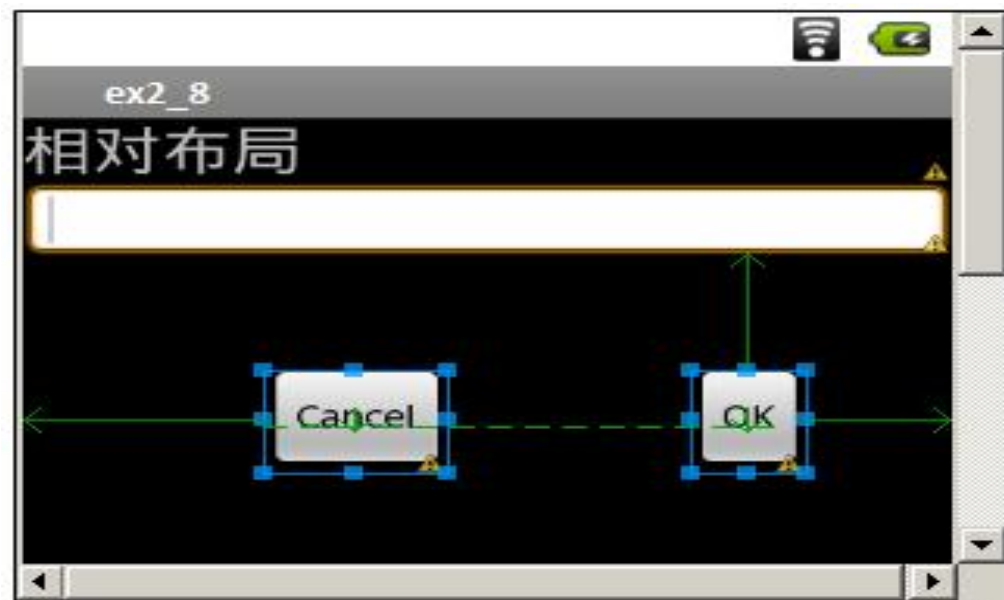
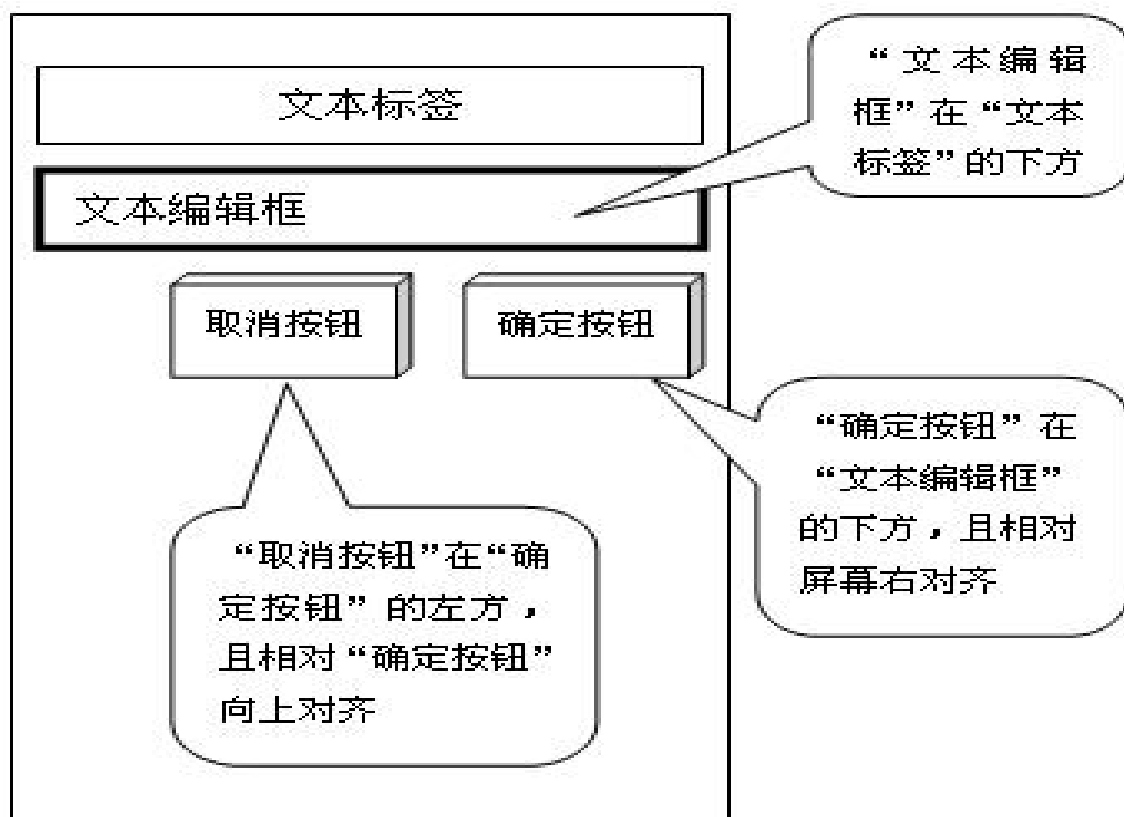
(2) 控制文件Ex2_7Activity.java

```
package com.ex03_07;
import android.app.Activity;
import android.os.Bundle;
import android.widget.ImageView;
public class Ex03_07Activity extends Activity
{
    ImageView img1, img2, img3, img4, img5;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        img1 = (ImageView) this.findViewById(R.id.mImageView1);
        img2 = (ImageView) this.findViewById(R.id.mImageView2);
        img3 = (ImageView) this.findViewById(R.id.mImageView3);
        img4 = (ImageView) this.findViewById(R.id.mImageView4);
        img5 = (ImageView) this.findViewById(R.id.mImageView5);
        img1.setImageResource(R.drawable.img1);
        img2.setImageResource(R.drawable.img2);
        img3.setImageResource(R.drawable.img3);
        img4.setImageResource(R.drawable.img4);
        img5.setImageResource(R.drawable.img5);
    }
}
```

4、相对布局RelativeLayout

- 相对布局RelativeLayout是采用相对其它组件的位置的布局方式。在相对布局中，通过指定ID关联其他组件，与之右对齐、上下对齐或屏幕中央等方式来排列组件。

【例2-8】应用相对布局设计一个组件排列如图所示的应用程序。



```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:id="@+id/label"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textSize="24sp"
        android:text="相对布局"/>
    <EditText
        android:id="@+id/edit"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"

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

        android:layout_below="@id/label"/> <!-- 指定在文本标签
    的下方 -->

```

```

<Button
    android:id="@+id/ok"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/edit" <!-- 指定在文本
编辑框的下方 -->
    android:layout_alignParentRight="true"
    android:layout_marginLeft="10dip"
    android:text="OK" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/ok" <!-- 指定在OK
按钮的左方 -->
        android:layout_alignTop="@id/ok"
        android:text="Cancel" />
</RelativeLayout>

```

5、 网格布局GridLayout

- 网格布局GridLayout是把设置区域划分为若干行和若干列的网格，网格中的一个组件可以占据多行或多列。

网格布局 GridLayout 的主要属性:

alignment Mode

设置布局管理器的对齐方式

columnCount

设置网格列数量

rowCount

设置网格行数量

layout_columnSpan

设置组件占据列数

layout_rowSpan

设置组件占据行数

【例2-9】应用网格布局设计一个计算器界面。

计算器的设计界面如图所示。在界面设计区域中，设置一个6行4列的网格布局，第一行为显示数据的文本标签，第二行为清除数据的按钮，第3~6行均划分为4列，共安排16个按钮，分别代表数字0、1、2、……、9及加、减、乘、除、等号等符号。



<But t on

androi d: l ayo ut _wi dt h="mat ch_ par ent "

androi d: l ayo ut _hei ght ="wr ap_ cont ent "

androi d: l ayo ut _col umSpan="4"



该组件占据 4 列的位置

androi d: t ext ="清除"

androi d: t ext Si ze="26sp" />

<But t on androi d: t ext ="1" androi d: t ext Si ze="26sp" />

<But t on androi d: t ext ="2" androi d: t ext Si ze="26sp" />

<But t on androi d: t ext ="3" androi d: t ext Si ze="26sp" />

0 <But t on androi d: t ext ="+" androi d: t ext Si ze="26sp" />

1 <But t on androi d: t ext ="4" androi d: t ext Si ze="26sp" />

2 <But t on androi d: t ext ="5" androi d: t ext Si ze="26sp" />

3 <But t on androi d: t ext ="6" androi d: t ext Si ze="26sp" />

4 <But t on androi d: t ext ="-" androi d: t ext Si ze="26sp" />

5 <But t on androi d: t ext ="7" androi d: t ext Si ze="26sp" />

6 <But t on androi d: t ext ="8" androi d: t ext Si ze="26sp" />

7 <But t on androi d: t ext ="9" androi d: t ext Si ze="26sp" />

8 <But t on androi d: t ext ="*" androi d: t ext Si ze="26sp" />

9 <But t on androi d: t ext ="." androi d: t ext Si ze="26sp" />

0 <But t on androi d: t ext ="0" androi d: t ext Si ze="26sp" />

1 <But t on androi d: t ext ="=" androi d: t ext Si ze="26sp" />

2 <But t on androi d: t ext ="/" androi d: t ext Si ze="26sp" />

3 </ Gr i dLayout >

修改布局文件activity_main.xml如下:

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:rowCount="6"
    android:columnCount="4"
>
<!-- 文本标签 -->
<TextView
0     android:layout_width="wrap_content"
1     android:layout_height="wrap_content"
2     android:layout_columnSpan="4"
3     android:layout_marginLeft="4px"
4     android:gravity="left"
5     android:text="0"
6     android:textSize="50dp"
7 />
```

设置网格为 6 行 4 列

该组件占据 4 列的位置

ConstraintLayout布局

默认布局，新版的都是**ConstraintLayout**

优点

- **Constraint Layout**可以在不嵌套view group的情况下实现非常庞大、复杂的布局。实现扁平化。
- **Constraint Layout**同时具有**Relative Layout**和**Linear Layout**的优点、特性。功能强大。使用**Constraint Layout**来布局时性能要比其他布局方式高。
- **Constraint Layout**无论是通过布局管理器拖拽，鼠标控制的形式实现还是使用**XML**代码去写，都比较方便。

- `app:layout_constraintLeft_toLeftOf="parent"`
`app:layout_constraintTop_toTopOf="parent"`
- 与左边的Parent对齐，与Parent的顶部对齐

这个相对位置的设置有点类似RelativeLayout的`layout_toLeftOf`、`alignParentLeft`等这些属性。ConstraintLayout一共支持相对位置的属性在此：

`layout_constraintLeft_toLeftOf`

`layout_constraintLeft_toRightOf`

`layout_constraintRight_toLeftOf`

`layout_constraintRight_toRightOf`

`layout_constraintTop_toTopOf`

`layout_constraintTop_toBottomOf`

`layout_constraintBottom_toTopOf`

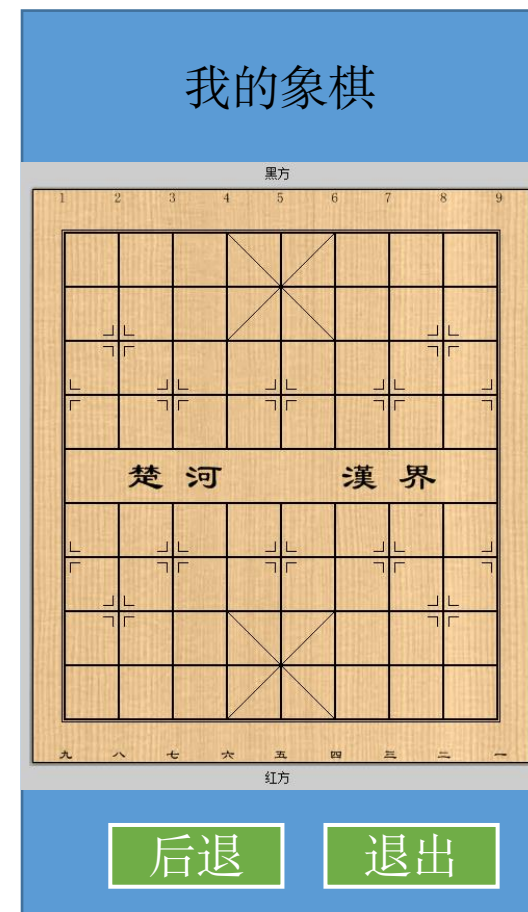
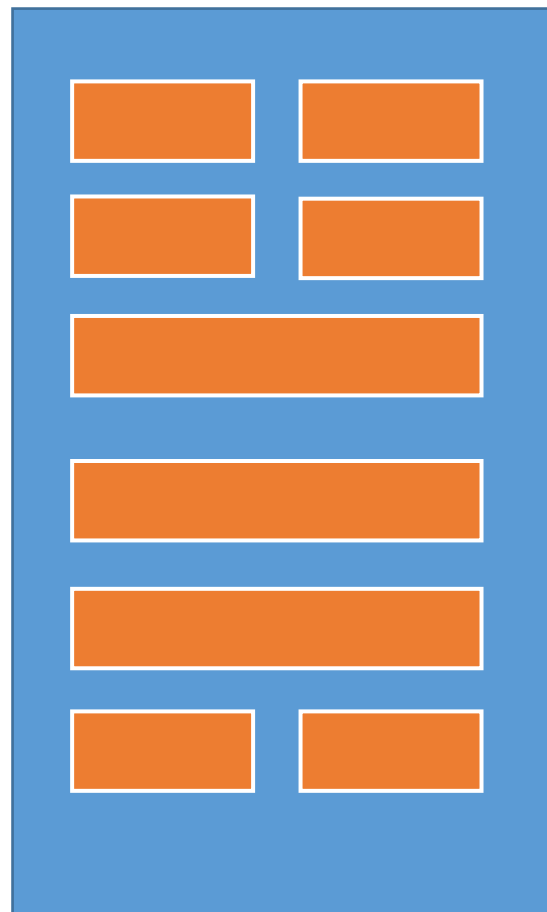
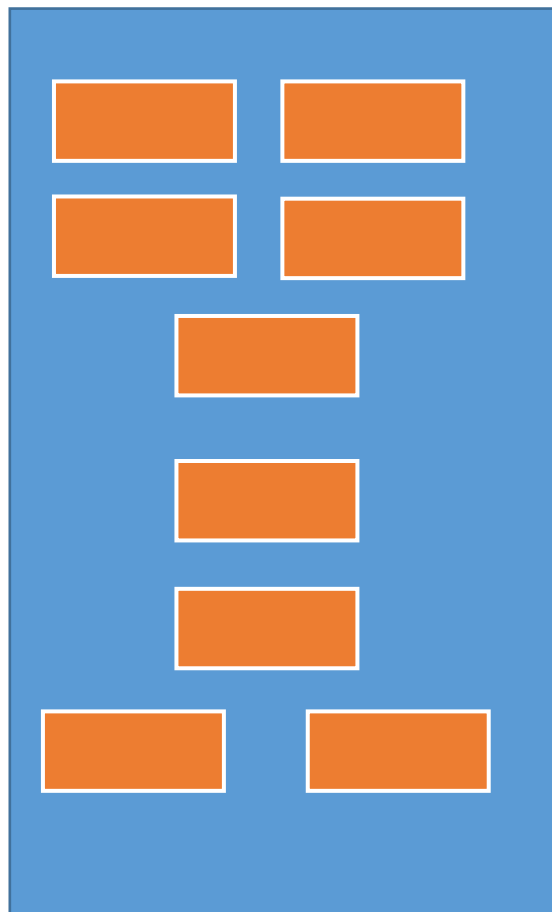
`layout_constraintBottom_toBottomOf` `layout_constraintBaseline_toBaselineOf`

`layout_constraintStart_toEndOf` `layout_constraintStart_toStartOf`

`layout_constraintEnd_toStartOf` `layout_constraintEnd_toEndOf`

拿第一个属性来说，`layout_constraintLeft_toLeftOf="@+id/id_first"`，表示当前View的左边与`id_first`的View的左边对齐。其实这个属性翻译成中文就是

练习：要完成下面界面，可选择的布局



2.5 选项按钮和进度条

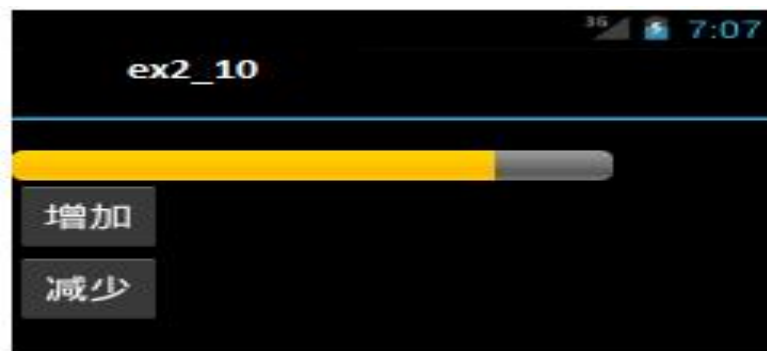
2.5.1 进度条

- 进度条ProgressBar能以形象地图示方式直观显示某个过程的进度。进度条ProgressBar的常用属性和方法见表2-8。

属 性	方 法	功 能
android:max	setMax(int max)	设置进度条的变化范围为0 ~ max
android:progress	setProgress(int progress)	设置进度条的当前值（初始值）
	incrementProgressBy (int diff)	进度条的变化步长值

【例2-10】进度条应用示例。

- 在界面设计中，安排一个进度条组件，并设置两个按钮，用于控制进度条的进度变化。
- 程序设计步骤：
 - ◆ (1) 在布局文件中声明ProgressBar。
 - ◆ (2) 在Activity中获得ProgressBar实例。
 - ◆ (3) 调用ProgressBar的incrementProgressBy()方法增加和减少进度。



(1) 布局文件activity_main.xml

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

```
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/an  
droid"
```

```
    android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent"
```

```
    android:orientation="vertical" >
```

```
<TextView        android:layout_width="fill_parent"
```

```
    android:layout_height="wrap_content"
```

```
    android:text="" />
```

```
<ProgressBar     android:id="@+id/ProgressBar01"
```

```
    style="@android:style/Widget.ProgressBar.Horizontal"
```

```
    android:layout_width="250dp"
```

```
    android:layout_height="wrap_content"
```

```
    android:max="200"
```

```
    android:progress="50" >
```

```
</ProgressBar>
```

```
<Button          android:id="@+id/button1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/btn1" />
```

```
<Button          android:id="@+id/button2"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/btn2" />
```

```
</LinearLayout>
```


(2) 事件处理文件MainActivity.java

```
package com.ex2_10;
import android.app.Activity; . . .
public class Ex03_10Activity extends Activity {
    ProgressBar progressBar;
    Button btn1,btn2;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        progressBar =
        (ProgressBar)findViewById(R.id.ProgressBar01);
        btn1=(Button)findViewById(R.id.button1);
        btn2=(Button)findViewById(R.id.button2);
        btn1.setOnClickListener(new click1());
        btn2.setOnClickListener(new click2());
    }
    class click1 implements OnClickListener {
        public void onClick(View v) {
            progressBar.incrementProgressBy(5);
        }
    }
}
```

```
class click2 implements OnClickListener {
    public void onClick(View v) {
        progressBar.incrementProgressBy(-5);
    }
}
```

3.5.2 选项按钮

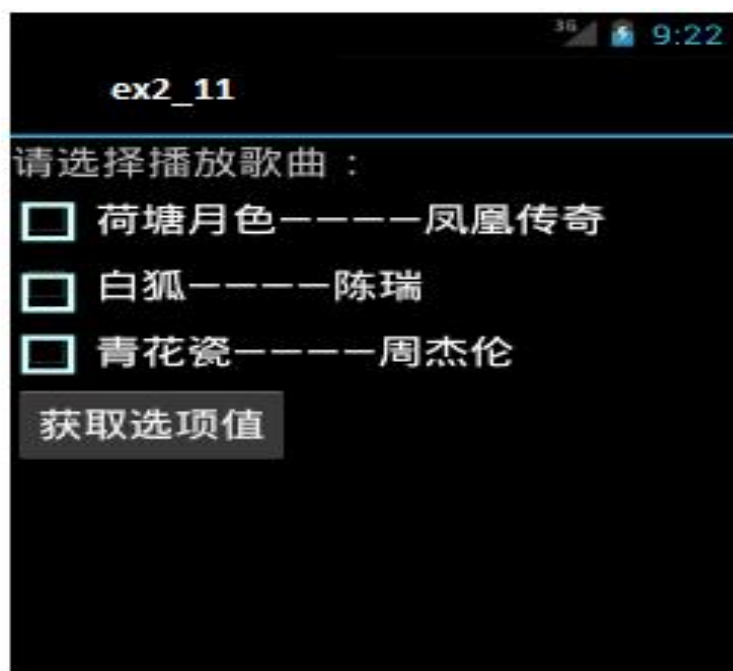
1、复选按钮CheckBox

复选按钮CheckBox用于多项选择的情形，用户可以一次性选择多个选项。复选按钮CheckBox是按钮Button的子类，其属性与方法继承于按钮Button。复选按钮CheckBox常用方法见表3-9。

方 法	功 能
<code>isChecked()</code>	判断选项是否被选中
<code>getText()</code>	获取复选按钮的文本内容

【例2-11】复选按钮应用示例。

- 在界面设计中，安排3个复选按钮和1个普通按钮，选择选项后，点击按钮，在文本标签中显示所选中的选项文本内容。如图所示。



程序设计步骤:

- (1) 在布局文件中声明复选按钮CheckBox。
- (2) 在Activity中获得复选按钮CheckBox实例。
- (3) 调用CheckBox的isChecked()方法判断该选项是否被选中。
如果选项被选中，则调用getText()方法获取选项的文本内容。

(程序代码见教材)

3.5.2 选项按钮

2、单选组件RadioGroup与单选按钮RadioButton

单选组件RadioGroup用于多项选择中只允许任选其中一项的情形。单选组件RadioGroup由一组单选按钮RadioButton组成。单选按钮RadioButton常用方法见表2-10。

方 法	功 能
<code>isChecked()</code>	判断选项是否被选中
<code>getText()</code>	获取单选按钮的文本内容

【例2-12】单选按钮应用示例。

- 在界面设计中，安排2个单选按钮、1个文本编辑框和1个普通按钮，选择选项后，点击按钮，在文本标签中显示文本编辑框及所选中的选项文本内容。如图所示。



程序设计步骤:

- (1) 在布局文件中声明单选组件RadioGroup和单选按钮RadioButton。
- (2) 在Activity中获得单选按钮RadioButton实例。
- (3) 调用RadioButton的isChecked()方法判断该选项是否被选中。如果选项被选中，则调用getText()方法获取选项的文本内容。

(程序代码见教材)

2.6 图像显示ImageView与画廊组件 Gallery

2.6.1 图像显示ImageView类

- ImageView类用于显示图片或图标等图像资源，并提供图像缩放及着色（渲染）等图像处理功能。

ImageView类的常用方法

元素属性	对应方法	说 明
android:maxHeight	setMaxHeight(int)	为显示图像提供最大高度的可选参数。
android:maxWidth	setMaxWidth(int)	为显示图像提供最大宽度的可选参数。
android:scaleType	setScaleType(ImageView.ScaleType)	控制图像适合 ImageView大小的显示方式。（参见表2-12）
android:src	setImageResource(int)	获取图像文件路径

ImageView类的scaleType属性值(表2-12)

scaleType属性值常量	值	说 明
matrix	0	用矩阵来绘图。
fitXY	1	拉伸图片（不按宽高比例）以填充View的宽高。
fitStart	2	按比例拉伸图片，拉伸后图片的高度为View的高度，且显示在View的左边。
fitCenter	3	按比例拉伸图片，拉伸后图片的高度为View的高度，且显示在View的中间。
fitEnd	4	按比例拉伸图片，拉伸后图片的高度为View的高度，且显示在View的右边。
center	5	按原图大小显示图片，但图片宽高大于View的宽高时，截图图片中间部分显示。
centerCrop	6	按比例放大原图直至等于某边View的宽高显示。
centerInside	7	当原图宽高或等于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。

【例2-13】 显示图片示例。

程序设计步骤:

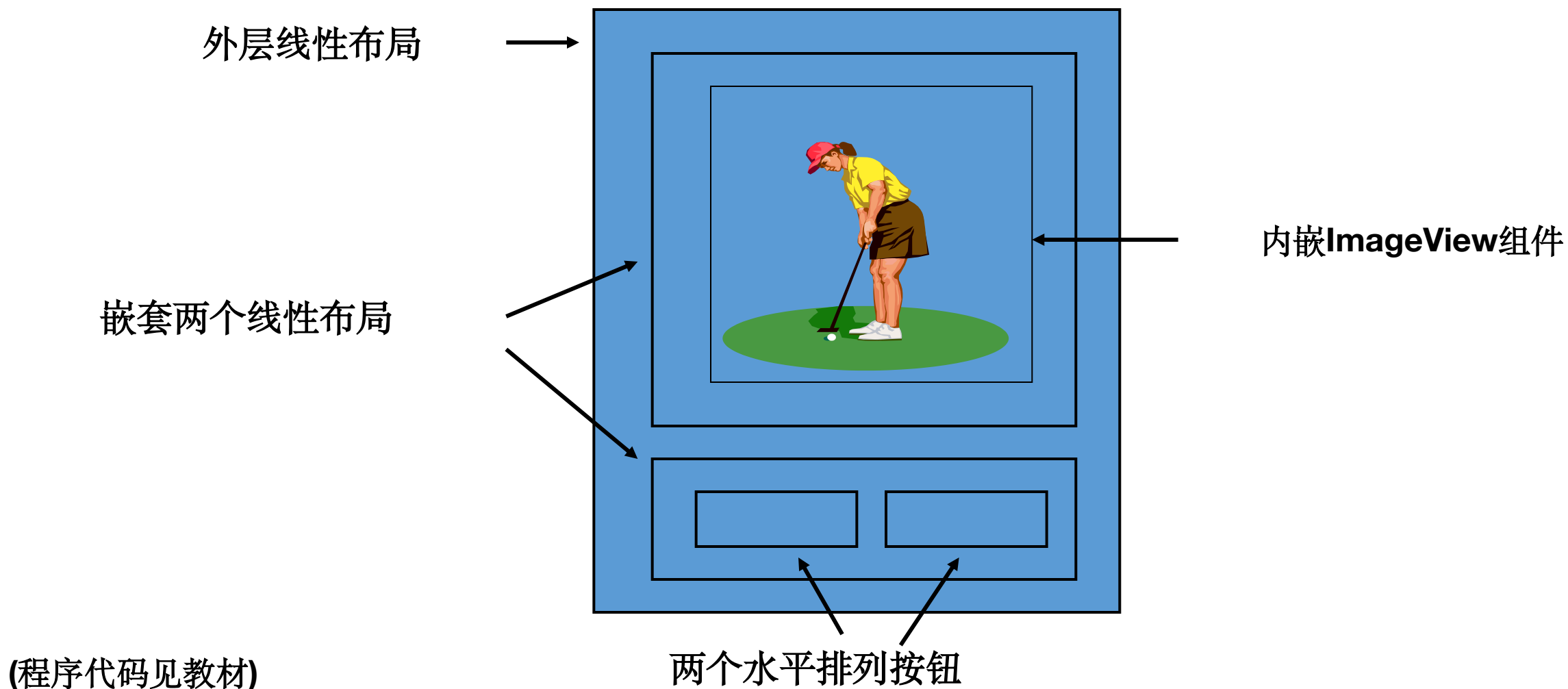
- (1) 将事先准备好的多张图片序列img1.jpg、img2.jpg、.....、img6.jpg复制到资源/res/drawable-hdpi目录下。
- (2) 在布局文件中声明图像显示组件ImageView。
- (3) 在Activity中获得相关组件实例。
- (4) 通过触发按钮事件，调用OnClickListener接口的onClick()方法显示图像。

【例2-13】显示图片示例。

- 在界面设计中，安排2个按钮和1个图像显示组件 **ImageView**，点击按钮，可以翻阅浏览图片。

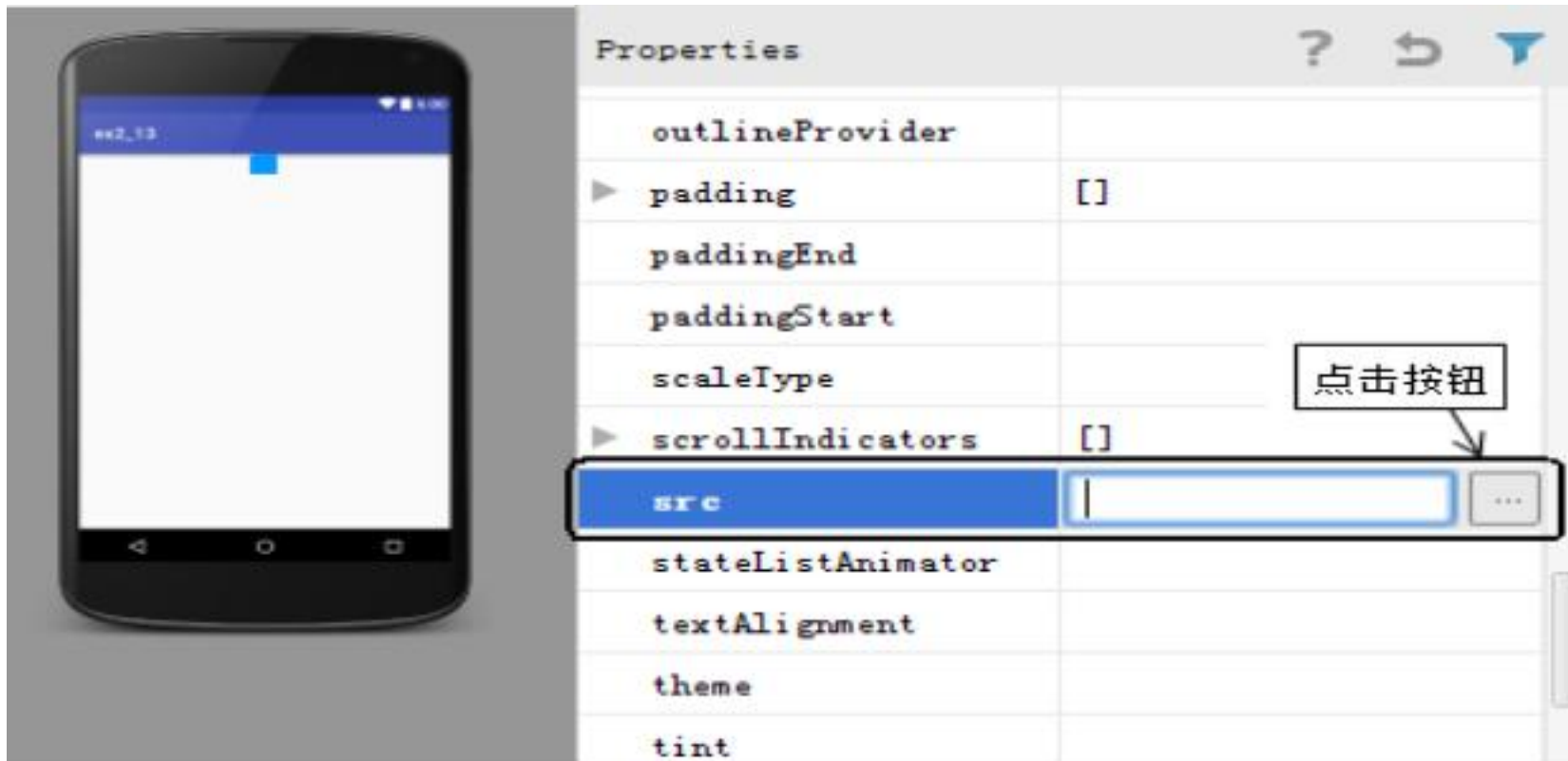


【例2-13】的界面布局:

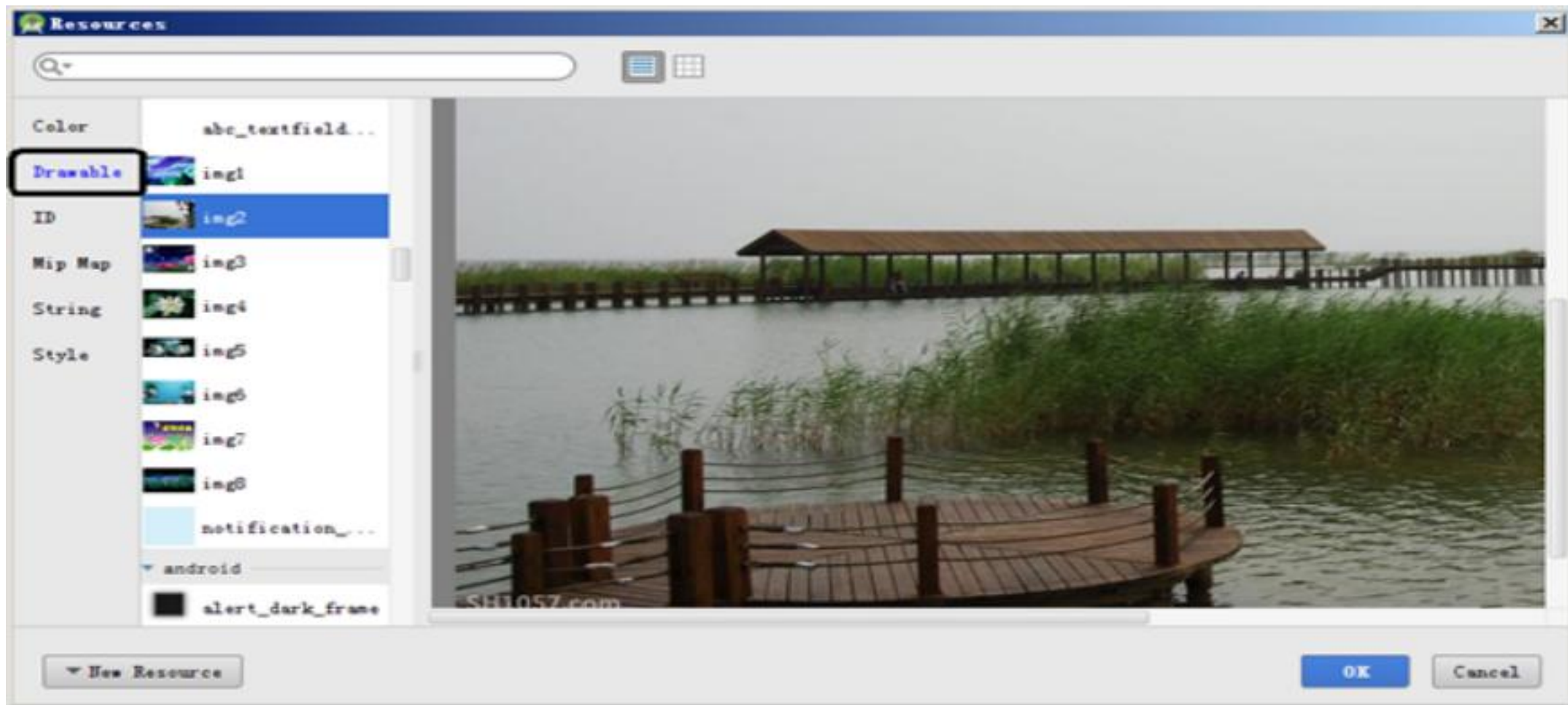


设置图像显示组件的数据源，其步骤如下：

- a) 选取ImageView组件，在属性（Properties）窗口中，选择src项，点击右边的按钮，设置显示图像的数据源，如图2.20所示。



- b) 在弹出的“Resources”窗口中，选择“Drawable”项，选取复制在/res/drawable目录下的图像，如图2.21所示。



2.6.2 画廊组件Gallery与 图片切换器ImageSwitcher

- Gallery类是Android中控制图片展示的一个组件，它可以横向显示一系列图像。
- Gallery类经常与图片切换器ImageSwitcher配合使用，用图片切换器ImageSwitcher类展示图片效果。使用ImageSwitcher时必须用ViewFactory接口的makeView()方法创建视图。

2.7 消息提示Toast

- 在Android系统中，可以用**Toast**来显示帮助或提示消息。该提示消息以浮于应用程序之上的形式显示在屏幕上。因为它并不获得焦点，不会影响用户的其他操作，使用消息提示组件**Toast**的目的就是为了尽可能不中断用户操作，并使用户看到提供的信息内容。

Toast类的常用方法

对应方法	说 明
<code>Toast(Context context)</code>	Toast的构造方法，构造一个空的Toast对象。
<code>makeText(Context context, CharSequence text, int duration)</code>	以特定时长显示文本内容，参数text为显示的文本，参数duration为显示时间，较长时间取值LENGTH_LONG，较短时间取值LENGTH_SHORT。
<code>getView()</code>	返回视图。
<code>setDuration(int duration)</code>	设置存续时间。
<code>setView(View view)</code>	设置要显示的视图。
<code>setGravity(int gravity, int xOffset, int yOffset)</code>	设置提示信息在屏幕上的显示位置。
<code>setText(int resId)</code>	更新makeText () 方法的所设置的文本内容。
<code>show()</code>	显示提示信息。
LENGTH_LONG	提示信息显示较长时间的常量。
LENGTH_SHORT	提示信息显示较短时间的常量。

【例2-15】消息提示Toast分别按默认方式、自定义方式和带图标方式显示的示例。

- 将事先准备好的图标文件icon.jpg复制到资源/res/drawable-hdpi目录下，以备作提示消息的图标之用。

(程序代码见教材)



2.8 列表组件 ListView 和ListActivity

2.8.1 列表组件ListView类

- ListView类是Android程序开发中经常用到组件，该组件时必须与适配器配合使用，由适配器提供显示样式和显示数据。

ListView类的常用属性和方法

对应方法	说 明
<code>ListView(Context context)</code>	构造方法
<code>setAdapter(ListAdapter adapter)</code>	设置提供数组选项的适配器
<code>addHeaderView (View v)</code>	设置列表项目的头部
<code>addFooterView (View v)</code>	设置列表项目的底部
<code>setOnClickListener(AdapterView.OnItemClickListener listener)</code>	注册单击选项时执行的方法，该方法继承于父类 <code>android.widget.AdapterView</code> 。

【例2-16】列表组件示例。

- 在界面设计中，设置1个文本标签和1个列表组件ListView。
- 程序设计步骤：
 - (1) 在布局文件中声明列表组件ListView。
 - (2) 在Activity中获得相关组件实例。
 - (3) 通过触发列表的选项事件，调用mClick类的onClick()方法显示相应提示内容。



(程序代码见教材)

2.8.2 ListActivity类

- 当整个Activity中只有一个ListView组件时，可以使用ListActivity。
- ListActivity类继承于Activity类，默认绑定了一个ListView组件，并提供一些与ListView处理相关的操作。
- ListActivity类常用的方法为getListView()，该方法返回绑定的ListView组件。

【例2-17】 ListActivity应用示例。

- (程序代码见教材)



2.9 滑动抽屉组件SlidingDraw

- 在Android系统中，也可以把多个程序放到一个应用程序的抽屉里。
 - 如图2.27 (a) 所示，单击“向上”图标按钮（称为手柄）时，打开抽屉；如图2.27 (b) 所示，单击“向下”图标按钮，关闭抽屉。

SlidingDraw类重要的XML属性

属 性	说 明
android:allowSingleTap	设置通过手柄打开或关闭滑动抽屉
android:animateOnClick	单击手柄时，是否加入动画，默认为true
android:handle	指定抽屉的手柄handle
android:content	隐藏在抽屉里的内容
android:orientation	滑动抽屉内的对齐方式

SlidingDrawer类的重要方法

方 法	说 明
animateOpen()	关闭时实现动画
animateOpen()	打开时实现动画
getContent()	获取内容
getHandle()	获取手柄
setOnDrawerOpenListener(SlidingDrawer.OnDrawerOpenListener onDrawerOpenListener)	打开抽屉的监听器
setOnDrawerCloseListener(SlidingDrawer.OnDrawerCloseListener onDrawerCloseListener)	关闭抽屉的监听器
setOnDrawerScrollListener(SlidingDrawer.OnDrawerScrollListener onDrawerScrollListener)	打开/关闭切换时的监听器

【例2-18】实现如图2.27所示的滑动抽屉SlidingDraw组件应用示例。

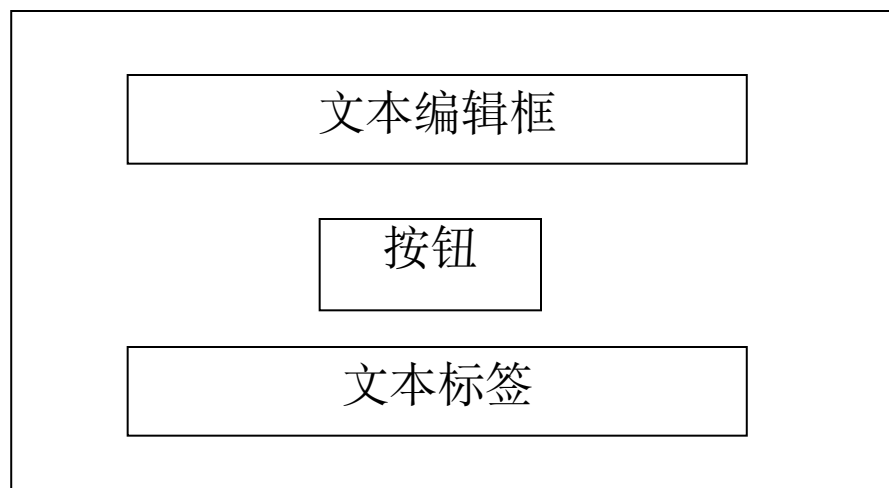
- 事先准备好两个图标文件，分别命名为up.jpg和down.jpg，将它们复制到/res/drawable-hdpi目录下，以备作滑动抽屉的手柄之用。

(代码见教材)



习题三

1、编写程序，如图所示，单击按钮，文本编辑框中输入的文字内容，显示到文本标签。



2、设计一个加法计算器，如图所示，在前二个文本编辑框中输入整数，单击按钮“=”时，在第三个文本编辑框中显示这两个数之和。

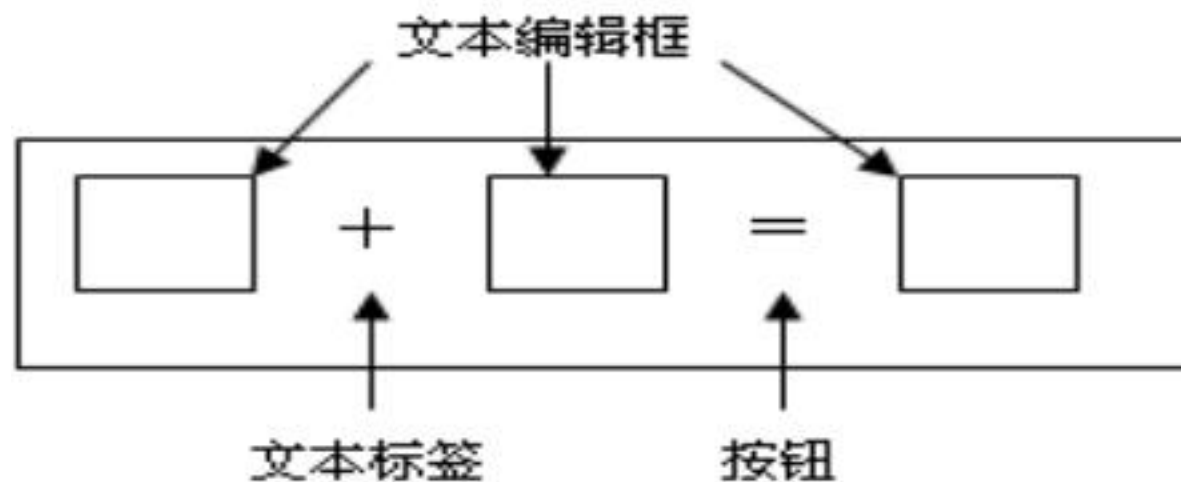
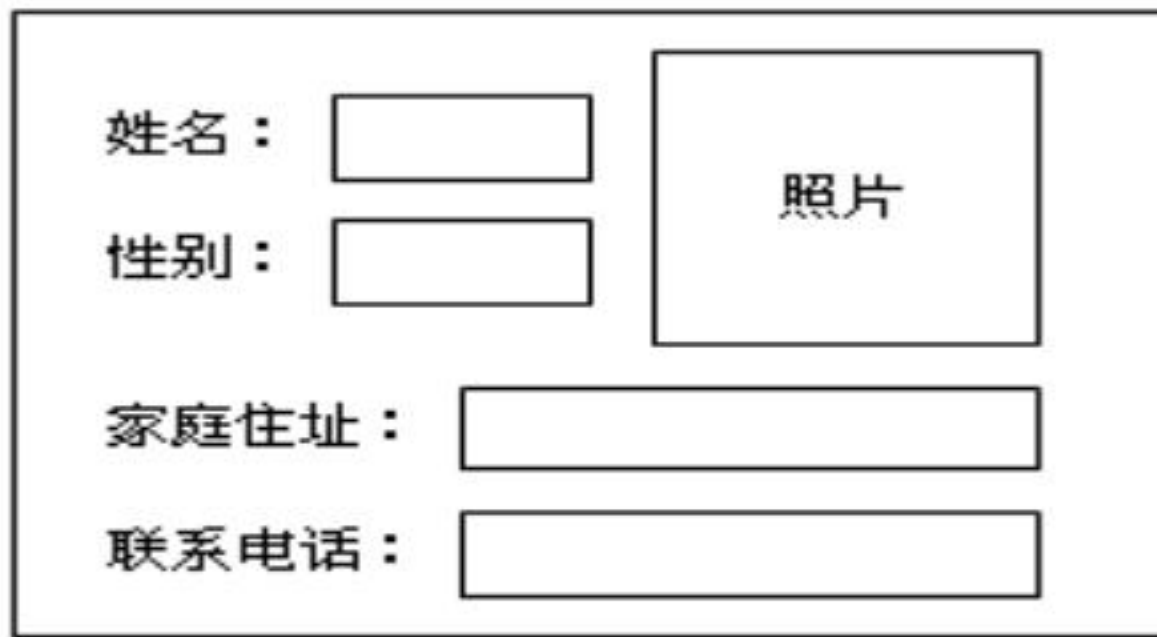


图 3.25 加法计算器

3、设计如图所示的用户界面布局。



A diagram of a user interface layout within a rectangular frame. On the left side, there are four labels with corresponding input fields: '姓名:' followed by a small rectangular box, '性别:' followed by a small rectangular box, '家庭住址:' followed by a long horizontal rectangular box, and '联系电话:' followed by a long horizontal rectangular box. To the right of these labels is a large square box containing the text '照片' (Photo).

图 3.26 设计用户界面

Q&A

谢谢大家！