第 3 章

Android 控件

Android 控件是 UI 的重要组成部分。常用的控件有：按钮控件 Button、ImageButton，状态开关控件 ToggleButton、Switch，单选多选按钮 RadioButton、CheckBox，图片控件 ImageView，列表控件 ListView，文本控件 TextView、EditText，下拉控件 Spinner，日期与时间选择控件 DatePicker、TimePicker。主要掌握这些控件的创建方法、类中的主要函数及事件处理机制。

3.1 类层次关系

在讲述具体控件内容之前，熟悉一下 Android 中类层次关系是非常有必要的，主要包括 View 与 ViewGroup、ViewGroup 与布局的关系等。

在 Android 知识体系中，View 是所有 UI 视图的基类，ViewGroup 是 View 的子类，所以它也具有 View 的特性，但主要用来充当 View 的容器，将其中的 View 视作自己的孩子，对它的子 View 进行管理。当然，它的孩子也可以是 ViewGroup 类型。ViewGroup（树根）和它的孩子（View 和 ViewGroup）以树形结构形成一个层次结构。View 类有接受和处理消息的功能，Android 系统所产生的消息会在这些 ViewGroup 和 View 之间传递。

所有布局类 LinearLayout、RelativeLayout、TableLayout、GridLayout、FrameLayout 等都是 ViewGroup 的子类。

总之，View、ViewGroup、部分控件类、部分布局类的 UML 类图如图 3-1 所示。

View 中常用的方法如下所示。

- View findViewById(int id)，这是 View 中最常用的方法，根据控件 id 值返回 View 对象。一般来说，若进一步使用，需要将 View 强制转换成所需对象，形如 Button obj=(Button)findViewById(id)；

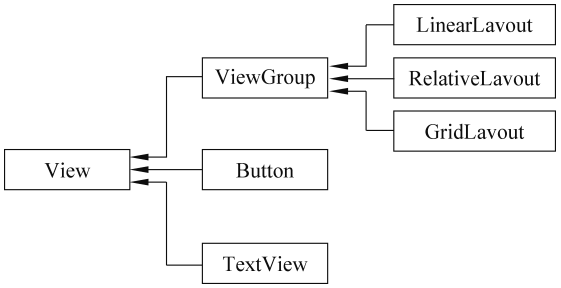

图 3-1　Android 基本类层次 UML 框图

ViewGroup 中常用的方法如下所示。

- void addView(View v),父容器添加一个子组件对象。
- void removeAllViews(),父容器删除所有子组件。
- void removeView(View v),父容器删除子组件 v。
- int getChildCount(),返回子组件数目。
- Resource getResources(),获取系统资源对象。

3.2　按钮控件

3.2.1　基本按钮 Button

Button 是 Android UI 的重要元素,其常用方法如下所示。

- Button(Context ctx),构造方法,ctx 是上下文对象。
- void setText(CharSequence cs),对应 android:text 属性,cs 是字符串序列,设置按钮文本内容。
- void setText(int resid),resid 是字符串资源 id,即原字符串定义在资源文件中。
- void setTextSize(float size),对应 android:textSize 属性,设置文本字体大小,浮点数,单位是 px。
- void setTextSize(int unit,float size),设置文本字体大小,浮点数,单位由 unit 决定。unit 是系统类 TypedValue 中定义的静态常量:COMPLEX_UNIT_PX,整数 0,代表单位是 px;COMPLEX_UNIT_DIP,整数 1,代表单位是 dp;COMPLEX_UNIT_SP,整数 2,代表单位是 sp。
- void setBackgroundColor(Color c),对应 android:background 属性,设置背景颜色。

- void setBackgroundResource(int resid)，设置背景图片，resid 是图片资源 id。
- void setEnabled(boolean mark)，对应 android:enabled 属性，按钮使能标志。

Button 按钮常用的动作有：click，短时单击事件；long click，长时单击事件；touch，触摸事件。若对按钮事件进行响应，必须完成以下步骤：①注册事件；②实现事件接口，编制响应函数。按钮事件常用的注册方法如下所示。

- void setOnClickListener(OnClickListener l)，注册 click 事件，响应方法写在实现 OnClickListener 接口类中定义的方法中。
- void setOnLongClickListener(OnLongClickListener l)，注册 long click 事件，响应方法写在实现 OnLongClickListener 接口类中定义的方法中。
- void setOnTouchListener(OnTouchListener l)，注册 click 事件，响应方法写在实现 OnTouchListener 接口类中定义的方法中。

【例 3-1】 按钮事件响应示例：定义线性布局 UI 文件，一个“开始”按钮，一个 TextView 组件。当单击“开始”按钮时，在 TextView 组件中显示“Hello”字符串。

① UI 配置文件 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:id="@+id/mystart"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="开始"
        android:textSize="25sp"/>
    <TextView
        android:id="@+id/mytext"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="25sp"/>
</LinearLayout>
```

② 事件响应：常用的方法有 4 种，如下所示。

方法 1：匿名内部类。

```
public class MainActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button btn = (Button)findViewById(R.id.mystart);
        btn.setOnClickListener(new View.OnClickListener() {
                                              //注册按钮 click 事件,定义匿名内部类
            public void onClick(View v) {     //实现响应函数
                Button srcBtn = (Button)v;    //强制转换成 button
                srcBtn.setEnabled(false);     //禁止按钮操作
                TextView tv = (TextView)MainActivity.this.findViewById(R.id
                    .mytext);                 //获得 TextView 对象
                tv.setText("Hello");          //显示 Hello
            }
        });
    }
}
```

由以上代码可知,匿名内部类一般用于函数参数中,见 btn.setOnClickListener()函数内部的参数定义。OnClickListener 是一个接口,内部仅定义了一个函数 onClick(View v),参数 v 是事件源对象,因此 v 可强制转换成 Button 对象,通过 setEnabled(false)禁止其使能状态。

由于在匿名内部类中函数获得外部类 MainActivity 中的 TextView 对象,因此一定要用如下代码:“TextView tv = (TextView)MainActivity.this.findViewById(R.id.mytext)”。注意,一定是 MainActivity.this,而不是 this。

方法 2:内部类实现。

```
public class MainActivity extends AppCompatActivity {
    class innerOper implements View.OnClickListener{
        public void onClick(View v) {
            Button srcBtn = (Button)v;
            srcBtn.setEnabled(false);
            TextView tv = (TextView)findViewById(R.id.mytext);
            tv.setText("Hello");
        }
    }
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button btn = (Button)findViewById(R.id.mystart);
```

```
        btn.setOnClickListener(new innerOper());
    }
}
```

innerOper 是内部类，且实现了 OnClickListener 接口，重写了 onClick()函数，与界面中“开始”按钮的 click 事件对应。

方法 3：外部类实现。

把方法 2 中的类 innerOper 移出 MainActivity 类，作为外部类进行编译，会出现编译错误：“findViewById()函数没定义”。这是因为当 innerOper 作为内部类时，它可以共享外部类 MainActivity 的一切变量和方法，当然就能用 findViewById()函数。有读者说把 MainActivity 对象作为参数传到外部类中，不就解决问题了吗？当然可以，但有更专业的方法，那就是应用系统 Context 上下文对象，查看 Android 帮助文档，可以看出：对许多系统类，假设抽象为 XXX 类，都有 XXX(Context ct)构造方法。本示例中，入口类 MainActivity 是 Activity 的派生类，Activity 是 Context 的派生类，因此 MainActivity 对象也是 Context 对象。基于此，编制的按钮外部响应事件类及相关类代码如下所示。

```
//外部事件响应类:OuterOper.java
public class OuterOper implements View.OnClickListener {
    Context ct;
    OuterOper(Context ct){
        this.ct = ct;
    }
    public void onClick(View v) {
        Button srcBtn = (Button)v;
        srcBtn.setEnabled(false);
        MainActivity obj = (MainActivity)ct;
                                        //将 Context 对象转化为 MainActivity 对象
        TextView tv = (TextView)obj.findViewById(R.id.mytext);
        tv.setText("Hello");
    }
}
//入口类:MainActivity.java
public class MainActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button btn = (Button)findViewById(R.id.mystart);
```

```
        btn.setOnClickListener(new OuterOper(this));
    }
}
```

方法 4：配置文件方法。

在 main.xml 配置文件中的 Button 配置部分增加 onClick 属性，如下所示（仅列出 Button 配置部分）。

```
<Button
    android:id="@+id/mystart"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="开始"
    android:textSize="25sp"
    android:onClick="myClick" />
```

然后，在 MainActivity 类中增加 myClick(View v)函数，代码如下所示。

```
public class MainActivity extends AppCompatActivity {
    public void onClick(View v) {
        Button srcBtn = (Button)v;
        srcBtn.setEnabled(false);
        TextView tv = (TextView)findViewById(R.id.mytext);
        tv.setText("Hello");
    }
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

3.2.2　图像按钮 ImageButton

为丰富 Android UI，有时需要图像按钮。图像按钮可以给设计者更广阔的想象空间。ImageButton 与 Button 最大的区别是，前者不需要文字，android:text 属性对 ImageButton 是不起作用的。

ImageButton 类常用的函数如下所示。

- ImageButton(Context ct)，构造方法，ct 是上下文对象。
- void setImageResource(int resid)，对应 android:src 属性，设置按钮图像，resid 是

图像资源号，图像一般保存在 res\drawable\目录下。

【例 3-2】 图像按钮事件响应示例：定制线性布局页面，有一个图像按钮，初始时显示图像 a.png。当手按下后显示图像 b.png、手抬起后显示图像 a.png。

一般来说，将图像 a.png、b.png 保存在 res\drawable\目录下。

① 界面配置文件 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ImageButton
        android:id="@+id/mybtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/a"/>
</LinearLayout>
```

由代码可知，默认初始图像按钮显示图像 res\drawable\a.png，其路径用@drawable/a 描述，不带文件名后缀，不要写成 a.png。

② MainActivity.java。

```
public class MainActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final ImageButton btn = (ImageButton)findViewById(R.id.mybtn);
        btn.setOnTouchListener(new View.OnTouchListener() {  //注册触摸事件
            public boolean onTouch(View v, MotionEvent event) {
                int r = event.getAction();                    //获得特征值
                if(r==MotionEvent.ACTION_DOWN){               //动作是按下
                    btn.setImageResource(R.drawable.b);       //设置 b 图像
                }
                if(r==MotionEvent.ACTION_UP){                 //动作是抬起
                    btn.setImageResource(R.drawable.a);       //设置 a 图像
                }
                return true;
            }
        });
```

```
    }
}
```

很明显，需对图像按钮注册 touch 触摸事件，通过 setOnTouchListener()函数完成，响应函数是匿名内部类中的 onTouch()。touch 触摸事件一般包括 3 部分：按住、移动、松开，它们均属于 MotionEvent 对象，具体哪一个是由 MotionEvent 类中的 getAction()函数的返回值决定。返回值等于 MotionEvent.ACTION_DOWN，表明是按住事件；返回值等于 MotionEvent.ACTION_MOVE，表明是移动事件；返回值等于 MotionEvent.ACTION_UP，表明是松开事件。

3.3　状态开关

3.3.1　ToggleButton 开关

ToggleButton 是一个具有选中和未选中双状态的按钮，并且需要为不同的状态设置不同的显示文本。其常用函数如下所示。

- ToggleButton(Context ct)，构造方法，ct 是上下文对象。
- void setTextOn(CharSequence ontxt)，对应 android:textOn 属性，设置开关打开时的文字内容。
- void setTextOff(CharSequence offtxt)，对应 android:textOff 属性，设置开关关闭时的文字内容。
- boolean isChecked()，对应 android:checked 属性，返回开关状态布尔值。

ToggleButton 按钮的常用事件注册函数如下所示。

- setOnCheckedChangeListener(OnCheckedChangeListener l)，注册 check 变化侦听事件，OnCheckedChangeListener 是系统接口，响应函数是该接口中定义的 onCheckedChange()函数。

【例 3-3】 ToggleButton 示例：定制线性布局页面，有一个 ToggleButton 双态按钮(打开、关闭)，初始时处于“打开”状态。当不断地“按下-松开”该按钮，在手机屏幕上利用 Toast 交替显示“关闭状态”“打开状态”。

① 在 UI 配置文件 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
```

```
    <ToggleButton
        android:id="@+id/mytoggle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="30sp"
        android:checked="true"
        android:textOff="关闭"
        android:textOn="打开" />
</LinearLayout>
```

② MainActivity.java。

```
public class MainActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ToggleButton btn = (ToggleButton)findViewById(R.id.mytoggle);
        btn.setOnCheckedChangeListener(new CompoundButton.OnCheckedChange-
Listener() {
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                if(isChecked)
                    Toast.makeText(MainActivity.this,"开关打开",Toast.LENGTH_
LONG).show();
                else
                    Toast.makeText(MainActivity.this,"开关关闭",Toast.LENGTH_
LONG).show();
            }
        });
    }
}
```

其界面如图 3-2 所示。

图 3-2　ToggleButton 示例

3.3.2　Switch 开关

Switch 开关也是一个双态开关，与 ToggleButton 相比，从界面来说，它更美观。其常用函数及事件注册函数与 ToggleButton 几乎是一致的，增量的知识点如下所示。

- Switch(Context ct)，构造方法，ct 是上下文对象。
- void setShowText(boolean mark)，对应 android:showText。当 mark 为 true 时，android:textOn 及 android:textOff 设置的字符串内容才能显示在屏幕上。
- void setTrackResource(int resid)，设置 Track 滑道图片资源，对应 android:track 属性，resid 是资源号，图片资源一般保存在 res\drawable\目录下。
- void setThumbResource(int resid)，设置 Thumb 滑块图片资源，对应 android:thumb 属性，resid 是资源号，图片资源一般保存在 res\drawable\目录下。

Switch 开关也可叫作滑动开关，当“关闭”时，一般滑块(Thumb)位于滑道(Track)的左侧；当“打开”时，一般滑块位于滑道的右侧，同时将其“高亮”显示。

【例 3-4】 系统 Switch 开关示例：定制线性布局页面，有一个 Switch 双态按钮(打开、关闭)，初始时处于“关闭”状态。当不断地“按下-松开”该按钮，在手机屏幕上利用 Toast 交替显示“打开状态”“关闭状态”。

① 在 UI 配置文件 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Switch
        android:id="@+id/myswitch"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="30sp"
        android:text="开关状态"
        android:textOn="打开"
        android:textOff="关闭"
        android:showText="true" />
</LinearLayout>
```

② MainActivity.java。

```
public class MainActivity extends AppCompatActivity {
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Switch btn = (Switch)findViewById(R.id.myswitch);
        btn.setOnCheckedChangeListener(new CompoundButton
.OnCheckedChangeListener() {
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                if(isChecked)
                    Toast.makeText(MainActivity.this,"开关打开",Toast.LENGTH_
LONG).show();
                else
                    Toast.makeText(MainActivity.this,"开关关闭",Toast.LENGTH_
LONG).show();
            }
        });
    }
}
```

其界面如图 3-3 所示。

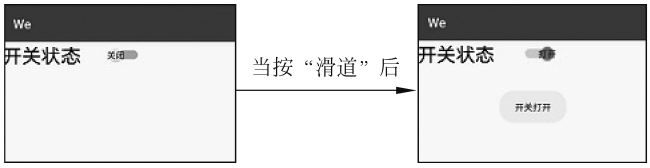

图 3-3　系统 Switch 开关示例

可以看出，Android 系统默认的滑道图片形似长方形，滑块图片是一个圆形。当 Switch 开关关闭时，圆形滑块位于左侧，且是白色；当 Switch 开关打开时，圆形滑块位于右侧，且是红色（表示高亮）。

实际上，从此例可推出：ToggleButton 开关必须在按钮上显示文字，用以区分打开、关闭状态，而 Switch 开关完全可以用滑道、滑块颜色的变化表示开关的打开、关闭。读者可以看看自己手机的设置部分，许多双态按钮都类似 Switch 开关，完全是由颜色变化区分状态的。

【例 3-5】 自定义 Switch 开关示例：主要对 track 及 thumb 参数进行设置。track 有打开、关闭状态；thumb 也有打开、关闭状态，因此每个 track、thumb 图片资源最多定义 2 个。为了方便，令 track 图片在打开、关闭两个状态下不变化，定义为一个资源即可，假设

为 track.jpg。thumb 图片定义为 2 个：一个为 off.jpg(对应关闭状态)；一个为 on.jpg(对应打开状态)。

注意：一般来说，滑块、滑道图片的高度应一致，滑道图片的宽度要大于或等于 2 倍滑块图片的宽度。

① 在 res\drawable\目录下，按题意建立滑道图片 track.jpg 和滑块图片 off.jpg、on.jpg。

② 在 UI 配置文件 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Switch
        android:id="@+id/myswitch"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="开关状态" />
</LinearLayout>
```

配置文件中仅定义了 Switch 组件，其滑道、滑块图片动态设置是在程序中实现的，勿需在此设置。

③ MainActivity.java。

```
public class MainActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Switch btn = (Switch)findViewById(R.id.myswitch);
        btn.setTrackResource(R.drawable.track);//设置滑道图片
        btn.setThumbResource(R.drawable.off);  //设置滑块图片(初始为关闭状态)
        btn.setOnCheckedChangeListener(new CompoundButton
.OnCheckedChangeListener() {
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                if(isChecked) {
                    btn.setThumbResource(R.drawable.on);
                                                    //若开关打开,则设置 on 图片
```

```
                Toast.makeText(MainActivity.this, "开关打开", Toast.LENGTH_
LONG).show();
            }
            else {
                btn.setThumbResource(R.drawable.off);
                            //若开关关闭,则设置 off 图片
                Toast.makeText(MainActivity.this, "开关关闭", Toast.LENGTH_
LONG).show();
            }
        }
    });
  }
}
```

3.4 单选按钮和多选按钮

3.4.1 RadioButton 单选按钮

RadioButton 是常用的单选按钮。为方便实现单选功能，一般将 RadioGroup 作为父容器，内部放置多个 RadioButton 子控件，这些子控件要么没有选中，要么同时只能有一个 RadioButton 子控件被选中。

RadioGroup 类常用的函数如下所示。

- void setOrientation(int mark)，设置添加子组件方式，对应 android:orientation 属性。当 mark = RadioGroup.VERTICAL 时，垂直添加子组件；当 mark = RadioGroup.HORIZONTAL 时，水平添加子组件。
- void getChildCount()，获得子组件数量。
- int getCheckedRadioButtonId()，获得选中的 RadioButton 对象 ID 值，若无选中的单选按钮对象，则返回－1。

RadioButton 类常用的函数如下所示。

- boolean isChecked()，获得布尔值，若为 true，则表示选中；若为 false，则表示未选中。

【例 3-6】 RadioButton 示例：定义一组单选按钮选项，分别是语文、数学、英语；再定义一个 OK 按钮，当单击 OK 按钮时，利用 Toast 显示选中单选按钮的内容，若没有选中的按钮，则显示“无”。

① 在 UI 布局文件 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
<RadioGroup
        android:id="@+id/mygrp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical">
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="语文"/>
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="数学"/>
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="英语"/>
</RadioGroup>
<Button
        android:id="@+id/myok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="OK"/>
</LinearLayout>
```

代码中仅对 RadioGroup 节点定义了 android:id 属性，各单选 RadioButton 子节点没有定义 android:id 属性。这是因为根据 id 属性值可获取父节点 RadioGroup 对象。当然，可遍历其子节点对象，因此可不对各 RadioButton 子节点设置 android:id 值。

② MainActivity.java。

```
public class MainActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button btn = (Button)findViewById(R.id.myok);
        btn.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                String s = "";
                RadioGroup rg = (RadioGroup)MainActivity.this.findViewById
(R.id.mygrp);
                int id = rg.getCheckedRadioButtonId();
                if(id==-1)
                    s = "你没有选中任何科目!";
                else{
                    Button sel = (Button)MainActivity.this.findViewById(id);
                    s = "你选中的科目是:" +sel.getText();
                }
                Toast.makeText(MainActivity.this, s, Toast.LENGTH_LONG).show();
            }
        });
    }
}
```

示例中获取选中单选按钮的步骤是：①根据 id 值获取 RadioGroup 对象 rg；②通过调用 RadioGroup 类中的 getCheckedRadioButtonId()，获取选中的单选按钮 id 值；③根据 id 值，获得选中的单选按钮对象 sel；④根据 sel，调用 getText()，获取选中单选按钮的文本字符串值。

当然，也可用第二种方法确定选中的单选对象，步骤是：①根据 id 值，获取 RadioGroup 对象 rg；②遍历 rg 的各 RadioButton 子对象，若某子对象的 isChecked()函数返回 true，则该单选按钮被选中。部分关键代码如下所示。

```
String s = "你没有选中任何科目!";
RadioGroup rg = (RadioGroup)MainActivity.this.findViewById(R.id.mygrp);
for(int i=0; i<rg.getChildCount(); i++) {              //遍历各子节点
    RadioButton rb = (RadioButton)rg.getChildAt(i);    //获取各具体子节点
if(rb.isChecked()){                                    //若选中
        s = "你选中的科目是:" + rb.getText(); break;
    }
}
Toast.makeText(MainActivity.this, s, Toast.LENGTH_LONG).show();
```

该示例界面如图 3-4 所示。

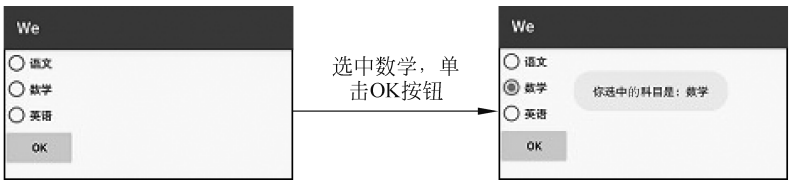

图 3-4　单选按钮示例

3.4.2　深入探究

1. 动态生成界面

人们经常利用手机做单选的调查问卷，有的读者说利用布局配置文件很容易实现这样的界面，有 N 个调查问卷，就定制 N 个单选布局文件。诚然是可以的，但有无其他方法？当然有！即主要由程序动态生成界面，分析如下所示。

- 每个单选题目由题目内容＋选项组成，需求分析是稳定的，完全可以利用循环生成 N 个单选题目的界面。
- 为了方便，可以借助 UI 布局文件，仅定制一道题目（包括题目内容及选项）的样式，包括字体、边距等，作为生成其他题目的样式模板。

仿真单选调查问卷功能的关键代码如下所示。

① 在 UI 布局文件 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/myline"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <TextView
        android:id="@+id/mytext"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:visibility="gone"
        android:layout_marginBottom=”10dp”/>
    <RadioGroup
        android:id="@+id/mygrp"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:visibility="gone"
        android:layout_marginBottom="15dp">
        <RadioButton
            android:id="@+id/mybtn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="语文"/>
    </RadioGroup>
</LinearLayout>
```

该文件主要定义了题目内容与选项、相邻两道题目之间的位置关系模板。模板主要涉及以 layout_为前缀的所有属性内容。TextView 节点是题目内容节点模板，由 android:layout_marginBottom＝"10dp"得出：题目内容与选项之间的间距为 10dp；RadioGroup 节点定义了该题与下一道题目的间距，由 android:layout_marginBottom＝“15dp”得出：两道相邻题目之间的间距是 15dp。

由于 TextView、RadioGroup 节点都不是真实的调查问卷所需内容，因此把这两个节点的 android:visibility 均设置为 gone，表示此两个节点在实际界面中是隐藏的，且不占空间，若设置成 invisible，也是不显示，但占用实际空间。

本例仅对 margin_bottom 属性做了设置，实际中可对 padding、margin 等所有需要的以 layout_为前缀的属性进行设置。这样，在程序中读一次模板参数，为所需要的 UI 控件进行设置，程序会非常简洁。看下述的实际代码。

② MainActivity.java。

```
public class MainActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView tv = (TextView)findViewById(R.id.mytext);
                                                        //获得题目内容模板对象
        RadioGroup rg = (RadioGroup)findViewById(R.id.mygrp);
                                                        //获得题目选项模板对象
        RadioButton rb = (RadioButton)findViewById(R.id.mybtn);
                                                        //获得单选按钮模板对象
        LinearLayout parent = (LinearLayout)findViewById(R.id.myline);
                                                        //获得总的父容器对象
```

```
        //二维字符串数组模拟题目内容和选项内容
        String s[][] = {{"你的性别","男","女"},
                {"你最喜欢的科目？","语文","数学","外语"},
                {"你最喜欢的乐器","钢琴","笛子","手风琴","二胡"}};
        for(int i=0; i<s.length; i++){
            TextView text = new TextView(this);          //产生题目内容对象
            text.setLayoutParams(tv.getLayoutParams());  //设置位置对象
            text.setText(s[i][0]);                       //设置题目内容
            RadioGroup grp = new RadioGroup(this);       //产生选项组对象
            grp.setLayoutParams(rg.getLayoutParams());   //设置位置对象
            for(int j=1; j<s[i].length; j++){
                RadioButton btn = new RadioButton(this);  //产生第 j 个单选按钮对象
                RadioGroup.LayoutParams param2 = new RadioGroup.LayoutParams
(rb.getLayoutParams());
                btn.setLayoutParams(param2);              //设置位置对象
                btn.setText(s[i][j]);                     //设置单选按钮文本内容
                grp.addView(btn);                         //单选按钮加入选项组
            }
            parent.addView(text);                        //加入第 i 道题目内容对象
            parent.addView(grp);                         //加入第 i 道题目选项组对象
        }
    }
}
```

示例中利用不规则二维数组 s 模仿了题目及选项内容，每行中第 0 项表示题目内容，第 1～n 项表示选项文本内容。例如，第 1 道题目的内容是“你的性别?”，选项有“男”和“女”。改变字符串内容，其他不用修改，即可生成新的调查问卷。或者从数据源（文件或数据库）按规则动态读入，即可形成新的所需界面。

代码中，核心思想是：利用循环添加题目内容及选项组对象，产生每个 UI 组件时，都要调用“组件.setLayoutParams(模板对象.getLayoutParams())”，确定组件的安放位置后，代码非常简洁。反之，知道每个 UI 组件以 layout_为前缀的属性非常多，如果一个个设置，代码会显得非常臃肿。从中也可体会出：每个组件都由“位置＋其他”属性组成，把“位置”单独提出来很有必要，可减少代码规模，而这一点单纯从布局配置文件中是体会不出来的，只有深入动态编程时，才可体会其中的精髓。

因此，一定不要被布局配置文件所左右，要灵活应用配置文件，让静态的配置文件动起来。

2. 属性增加问题

RadioButton 有属性 text，即显示在 UI 上的文本，但缺少一个属性 value。如何为 RadioButton 增加属性 value？如何在 UI 布局 XML 文件中应用 value 属性？步骤如下所示。

① 定义 RadioButton 派生类 MyRadioButton，value 成员变量（字符串类型）以及 setter()和 getter()方法。

② 为了能在布局配置文件中应用 MyRadioButton，要在工程 res\values\attrs.xml 中增加如下定义。当然，也可命名其他名字的文件，增加相同的内容。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<declare-styleable name="MyRadioButton"><!-- 控件名称-->
<attr name="value" format="string"/><!-- 属性名称,类型-->
</declare-styleable>
</resources>
```

此文件中定义了名为 MyRadioButton 的按钮，并定义了一个属性，名字为 value，类型是字符串 string。名字 value 必须与 MyRadioButton 定义的成员变量 value 的名称一致。若再增加其他属性，只增加相应的＜attr /＞节点即可。

③ 在②的基础上，完善的 MyRadioButton 类关键代码如下所示。

```
public class MyRadioButton extends RadioButton {
    String value;
    public MyRadioButton(Context context, AttributeSet attrs) {
        super(context, attrs);
        try {
            //获得 MyRadioButton 的属性-值集合
            TypedArray a = context.obtainStyledAttributes(attrs,R.styleable.
MyRadioButton);
            this.value = a.getString(R.styleable.MyRadioButton_value);
                                                                    //获得 value 值
            a.recycle();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public String getValue() {return value;}
```

```
    public void setValue(String value) {this.value = value;}
}
```

构造方法 MyRadioButton()不是主动调用的，而是 Android 系统自动调用的。一些关键代码的解释如下所示。

- super(context, attrs)，调用基类 RadioButton 构造方法，将 RadioButton 定义的“属性-值”集合保存在内存中。
- TypedArray a = context.obtainStyledAttributes(attrs,R.styleable.MyRadioButton)；

一方面将 MyRadioButton 自定义的“属性-值”集合添加到 attrs 对象中，也就是说，attrs 包含了 RadioButton 及派生类 MyRadioButton 的“属性-值”集合；另一方面获得 MyRadioButton 自定义“属性-值”集合对象（由 TypedArray a 描述）。代码中，obtainStyleAttributes()函数的第 2 个参数前缀 R.styleable 是固定的，末尾的 MyRadioButton 必须与上文 attrs.xml 中＜declare-styleable name＝"MyRadioButton"＞定义的 name 属性值相一致。

- this.value = a.getString(R.styleable.MyRadioButton_value)

获取 MyRadioButton 自定义属性 value 值。代码中，getString()函数的第 2 个参数前缀 R.styleable 是固定的，MyRadioButton 是控件名称，末尾的 value 必须与上文 attrs.xml 中＜attr name＝"value" format＝"string"/＞定义的 name 属性值相一致。

④ 简单测试。

前提条件是上文论述的 MyRadioButton 类，attrs.xml 已存在。测试功能是：定义布局文件 main.xml，仅包含 MyRadioButton 节点，并定义 value 属性值，在程序中获得该属性值并利用 Toast 输出该值。

//布局文件 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:myattr="http://schemas.android.com/apk/res/com.example.dqjbd.we"
    android:id="@+id/myline"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <com.example.dqjbd.we.MyRadioButton
        android:id="@+id/myradio"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Test"
```