实验二

用户界面设计

2017 年 10 月 8 日

实验介绍

用户界面设计是 Android 应用开发的一项重要内容，一个优秀的 Android 应用应该提供友好的图形用户界面（Graphics User Interface， GUI）。Android 提供了非常丰富的用户界面组件，借助于这些用户界面组件，开发者可以非常方便地进行用户界面开发，而且可以开发出非常优秀的用户界面。

控制用户界面有三种方法：

1） 使用 XML 布局文件控制 UI 界面：Android 推荐使用的一种方法，这种方法简单、明了，而且可以将应用的视图控制逻辑从 Java 代码中分离出来，放入 XML 文件中控制，很好地体现了 MVC 原则。

2） 使用代码控制 UI 界面：Android 允许开发者像开发 Swing 应用一样，完全在 Java 代码中控制 UI 界面。这样所有的 UI 组件都将通过 new 关键字创建出来，然后以合适的方式“搭建”在一起即可。

3） 使用 XML 布局文件和 Java 代码混合控制 UI 界面：完全使用 XML 布局文件控制 UI 界面虽然便捷，但有失灵活；完全通过 Java 代码控制 UI 界面虽然灵活，但过程烦琐，且不利于解耦。因此有些时候，需要混合使用 XML 布局文件和 Java 代码控制 UI 界面。习惯上把变化小、行为比较固定的组件放在 XML 布局文件中管理，而那些变化较多、行为控制比较复杂的组件则交给 Java 代码来管理。

在开始实验之前，需要注意以下几点：

- 该实验教程采用的 IDE 为 Android Studio 2.2，安卓虚拟设备即模拟器为 Nexus 5，系统镜像为 Android 7.0。其中，变更系统镜像为 Android 5.1 不会影响实验结果，变更模拟器可能会影响实验结果，但仅限于对用户界面的影响，就是说变更模拟器后，你可能会发现用户界面中一些组件的大小、清晰度发生了改变，但不会发现组件的功能发生改变或者程序突然有了 bug 不能运行，这是因为不同的模拟器只有尺寸、分辨率有所差别。
- 该教程中所有 Java 代码都没有对包的声明，因为包名取决于公司域名和工程名。在使用该教程的 Java 代码时，不要覆盖掉你自己 Java 代码中对包的声明，不然，会出现程序包 R 不存在的错误。
- 该教程中所有实验均只涉及一个 Activity，类别是 Empty Activity，命名采用默认名字 MainActivity，对应的布局文件命名为 activity_main。这两个名字可以自己修改，但修改后如果要使用该教程中代码，请记得对教程中的代码做出这样的修改：在布局文件中修改最外层布局的 tool：context 这一属性的值，把其中的“MainActivity”改为你自己的 Activity 的名字；在 Activity 的 Java 代码中修改 onCreate 函数中 setContentView 调用语句，把调用参数中的“activity_main”改为你自己的布局文件名。

- 教程中所有布局文件出现的所有 text 属性的字符串取值为了方便都是直接写在“”里，但是这种做法不提倡，一般要把字符串写到 string.xml 文件中，然后通过 @string/xxx 取得对应的字符串内容。类似的还有尺寸、颜色等资源。

1. 布局管理器

1.1 线性布局（LinearLayout）

线性布局控制其中的组件一个挨着一个地横向或纵向排列，每一行或列只能放一个组件，并且不会换行，当组件排列到窗体边缘后，剩下的组件将不会被显示出来。

下面通过一个简单实验熟悉一下线性布局的使用：

修改新建项目 res\layout 目录下的布局文件 activity_main.xml，设置线性布局排列方式为垂直，组件的对齐方式为 bottom|center_horizontal，背景颜色为#98FB98，长度和宽度都为 match_parent，并在线性布局中添加 3 个按钮，按钮 1 的长度和宽度都为 wrap_content，按钮 2 的长度和宽度分别为 match_parent、wrap_content，按钮 3 的长度和宽度分别为 100dp、50dp。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    android:gravity="bottom|center_horizontal"
    android:orientation="vertical"
    android:background="#98FB98"
    tools:context=".MainActivity">

    <Button
        android:text="按钮 1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <Button
        android:text="按钮 2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <Button
        android:text="按钮 3"
        android:layout_width="100dp"
        android:layout_height="50dp" />

</LinearLayout>
```

运行结果如下图所示：

1.2 表格布局（TableLayout）

表格布局继承了线性布局，采用行、列的形式管理放入其中的 UI 组件。TableLayout 通过添加 TableRow 和其他组件来控制表格的行数和列数。每次添加一个 TableRow，就是添加一个表格行，TableRow 也是容器，因此它也可以添加其他组件，每添加一个子组件，该表格就增加一列。如果直接向 TableLayout 中添加组件，那么这个组件将直接占用一行。在表格布局中，列的宽度由该列中最宽的那个单元格决定，整个表格布局的宽度则取决于父容器的宽度（默认总是占满父容器本身）。

下面通过一个实验来熟悉一下表格布局的使用：

修改新建项目 res\layout 目录下的布局文件 activity_main.xml，定义 3 个 TableLayout，其中，第一个 TableLayout 指定第二列被隐藏，第二个 TableLayout 指定第二、三列允许拉伸，第三个 TableLayout 指定第二列允许收缩。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TableLayout
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
```

```
        android:collapseColumns="1">
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="独自一行的按钮"/>
        <TableRow>
            <Button android:layout_height="wrap_content"
                android:layout_width="wrap_content"
                android:text="普通按钮 1"/>
            <Button android:layout_height="wrap_content"
                android:layout_width="wrap_content"
                android:text="普通按钮 2"/>
            <Button android:layout_height="wrap_content"
                android:layout_width="wrap_content"
                android:text="普通按钮 3"/>
        </TableRow>
    </TableLayout>

    <TableLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:stretchColumns="1,2">
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="独自一行的按钮"/>
        <TableRow>
        <Button android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:text="普通按钮"/>
        <Button android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:text="拉伸的按钮"/>
        <Button android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:text="拉伸的按钮"/>
        </TableRow>
    </TableLayout>

    <TableLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:shrinkColumns="1">
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="独自一行的按钮"/>
```

```
        <TableRow>
        <Button android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:text="普通按钮"/>
        <Button android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:text="收缩的按钮"/>
        <Button android:layout_height="wrap_content"
            android:layout_width="200dp"
            android:text="普通按钮"/>
        </TableRow>
    </TableLayout>
</LinearLayout>
```

运行结果如下图所示：

1.3 帧布局（FrameLayout）

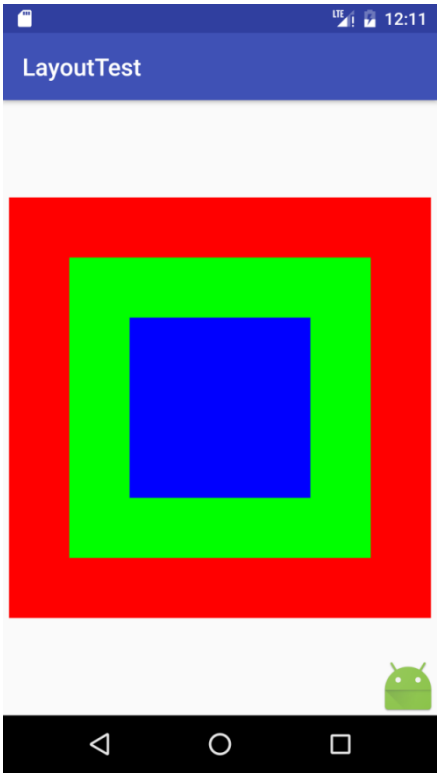
帧布局为每一个加入其中的组件创建一个空白的区域（称为一帧），这些帧都会根据 gravity 属性执行自动对齐。默认情况下，帧布局从屏幕的左上角（0, 0）坐标点开始布局，多个组件层叠排序，后面的组件覆盖前面的组件。

下面是一个简单的使用帧布局的实验：

修改新建项目 res\layout 目录下的布局文件 activity_main.xml，添加一个帧布局，并设置前景及其显示的位置，最后在该布局中添加三个居中显示的 TextView 组件，并且为其指定不同的颜色和大小。具体代码如下：

```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:foreground="@mipmap/ic_launcher"
    android:foregroundGravity="bottom|right"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="350dp"
        android:layout_height="350dp"
        android:layout_gravity="center"
        android:background="#f00"/>

    <TextView
        android:layout_width="250dp"
        android:layout_height="250dp"
        android:layout_gravity="center"
        android:background="#0f0"/>

    <TextView
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:layout_gravity="center"
        android:background="#00f"/>
</FrameLayout>
```

运行结果如下图所示：



1.4 相对布局（RelativeLayout）

相对布局通过组件之间的相对位置来布局，比如某个组件在另一个组件的左边或右边等。如果 A 组件的位置由 B 组件的位置来决定，Android 要求先定义 B 组件，再定义 A 组件。

下面通过一个实验来实现梅花布局：

修改新建项目 res\layout 目录下的布局文件 activity_main.xml，删除默认布局，添加相对布局，最后添加五个 ImageView 并设置它们的位置、填充和对齐方式。五个 ImageView 的图片如下:

分别命名 img1、img2 、img3 、img4 和 img5 并添加到 drawable 文件夹中，右键单击 drawable 并选择 Show in Explorer 即可找到该项目对应 drawable 文件夹。

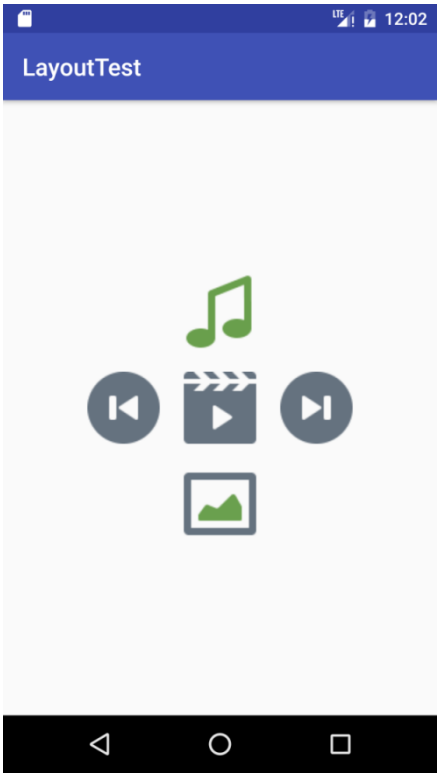
注意：梅花中心的 ImageView 必须设置 android:id 属性，因为我们需要通过这个属性来设置其他四个 ImageView 的位置。

具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
```

```
        android:id="@+id/img1"
        android:padding="10dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:src="@drawable/img1"/>

    <ImageView
        android:id="@+id/img2"
        android:padding="10dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/img1"
        android:layout_alignTop="@id/img1"
        android:src="@drawable/img2"/>

    <ImageView
        android:id="@+id/img3"
        android:padding="10dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/img1"
        android:layout_alignTop="@id/img1"
        android:src="@drawable/img3"/>

    <ImageView
        android:id="@+id/img4"
        android:padding="10dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@id/img1"
        android:layout_alignLeft="@id/img1"
        android:src="@drawable/img4"/>

    <ImageView
        android:id="@+id/img5"
        android:padding="10dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/img1"
        android:layout_alignLeft="@id/img1"
        android:src="@drawable/img5"/>

</RelativeLayout>
```

运行结果如下图所示：



1.5 网格布局（GridLayout）

GridLayout 的作用类似 HTML 中的 table 标签，它把整个容器划分为 rows*columns 个网格，每个网格可以放置一个组件。除此之外，也可以设置一个组件横跨多少列、一个组件纵跨多少行。

下面通过使用网格布局来实现一个简单的计算器界面：

修改布局文件添加 GridLayout，并在该 GridLayout 中依次定义文本框和两个按钮，其中文本框横跨 4 列，两个按钮各横跨 2 列。布局文件代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:columnCount="4"
    android:rowCount="6"
    android:orientation="horizontal"
    tools:context=".MainActivity">
    <TextView
        android:layout_columnSpan="4"
        android:layout_gravity="fill"
        android:layout_marginLeft="3dp"
        android:layout_marginRight="3dp"
```

```
        android:padding="2dp"
        android:background="#FFCCCC"
        android:text="0"
        android:textColor="#000"
        android:textSize="50sp" />

    <Button
        android:layout_columnSpan="2"
        android:layout_gravity="fill"
        android:text="回退"
        android:textSize="35sp"/>

    <Button
        android:layout_columnSpan="2"
        android:layout_gravity="fill"
        android:text="清空"
        android:textSize="35sp"/>

</GridLayout>
```

修改 MainActivity.java 文件，在代码中循环 16 次，依次添加 16 个按钮。具体代码如下：

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.GridLayout;

public class MainActivity extends AppCompatActivity {
    GridLayout gridLayout;
    //定义 16 个按钮的文本
    String[] chars = new String[]
            {
                    "+", "1", "2", "3",
                    "-", "4", "5", "6",
                    "*", "7", "8", "9",
                    "/", ".", "0", "="
            };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        gridLayout = (GridLayout)findViewById(R.id.activity_main);
        for(int i = 0;i < chars.length; i++)
        {
            Button bn = new Button(this);
```

```
            bn.setText(chars[i]);
            //设置按钮的字号大小
            bn.setTextSize(40);
            //设置按钮四周的空白区域
            bn.setPadding(5, 35, 5, 35);
            //指定组件所在的行
            GridLayout.Spec rowSpec = GridLayout.spec(i/4 + 2);
            //指定组件所在的列
            GridLayout.Spec columnSpec = GridLayout.spec(i%4);
            GridLayout.LayoutParams params = new GridLayout.LayoutParams(rowSpec, columnSpec);
            gridLayout.addView(bn, params);
        }
    }
}
```

运行结果如下图所示：

2. 文本框（TextView）

文本框用于在屏幕上显示文本，它可以显示单行文本，也可以显示多行文本，还可以显示带图像的文本。

下面通过一个简单的实验来熟悉文本框的使用：

修改布局文件，添加 4 个文本框，第一个是带图片的文本框，第二个是带阴影的文本框，第三个是矩形边框、渐变背景的文本框，第四个是圆角边框的文本框。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="我爱 Android"
        android:textSize="20pt"
        android:textColor="#EA5246"
        android:textStyle="bold|italic"
        android:background="#000000"
        android:drawableEnd="@mipmap/ic_launcher" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="带阴影的 TextView"
        android:textSize="18pt"
        android:textColor="#EA5246"
        android:shadowColor="#00f"
        android:shadowDx="10.0"
        android:shadowDy="8.0"
        android:shadowRadius="3.0"/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="矩形边框、渐变背景的 TextView"
        android:gravity="center"
        android:textSize="14pt"
        android:background="@drawable/bg_border1"/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="圆角边框的 TextView"
        android:textSize="14pt"
        android:background="@drawable/bg_border2"/>

</LinearLayout>
```

在默认情况下，TextView 是不带边框的，如果想添加边框，我们可以为 TextView 设置一个背景 Drawable，该 Drawable 只是一个边框。我们还可以在指定边框的同时指定渐变背景。这里我们通过 XML 文件来创建一个 Drawable 资源，下面是两个 XML 文件的代码，分别是 bg_border1.xml 和 bg_border2.xml，要将它们放在 drawable 文件夹内。

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

    <!-- 设置一个黑色边框 -->
    <stroke android:width="4px" android:color="#000000"/>
    <!-- 渐变 -->
    <gradient
        android:angle="270"
        android:endColor="#C0C0C0"
        android:startColor="#FCD209" />
    <!-- 设置一下边距,让空间大一点 -->
    <padding
        android:left="5dp"
        android:top="5dp"
        android:right="5dp"
        android:bottom="5dp"/>

</shape>
```

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

    <!-- 设置透明背景色 -->
    <solid android:color="#0000" />

    <!-- 设置一个红色边框 -->
    <stroke
        android:width="4px"
        android:color="#f00" />
    <!-- 设置四个圆角的半径 -->
    <corners
        android:bottomLeftRadius="20px"
        android:bottomRightRadius="20px"
        android:topLeftRadius="20px"
        android:topRightRadius="20px" />
    <!-- 设置一下边距,让空间大一点 -->
    <padding
        android:left="5dp"
        android:top="5dp"
```

```
        android:right="5dp"
        android:bottom="5dp"/>
</shape>
```

运行结果如下图所示：

3. 编辑框（EditText）、按钮（Button）、单选钮（RadioButton）和复选框（CheckBox）

编辑框接受用户输入，并且可以指定输入类型，继承自 TextView。按钮可供用户单击，当用户单击按钮时，按钮会触发一个 onClick 事件，继承自 TextView，TextView 大部分的 XML 属性也适用于 Button。单选钮和复选框都继承了 Button，都有可选中的功能，但是一组 RadioButton 只能选中其中一个，而一组 CheckBox 可以选中多个。

下面通过这些组件来实现一个简单的注册界面：

修改布局文件，使用表格布局并允许第二列拉伸，每个文本框和编辑框组合形成一行，并根据实际情况设置编辑框的输入类型，性别一栏使用单选钮，爱好一栏使用复选框，最后添加一个注册按钮。具体代码如下：

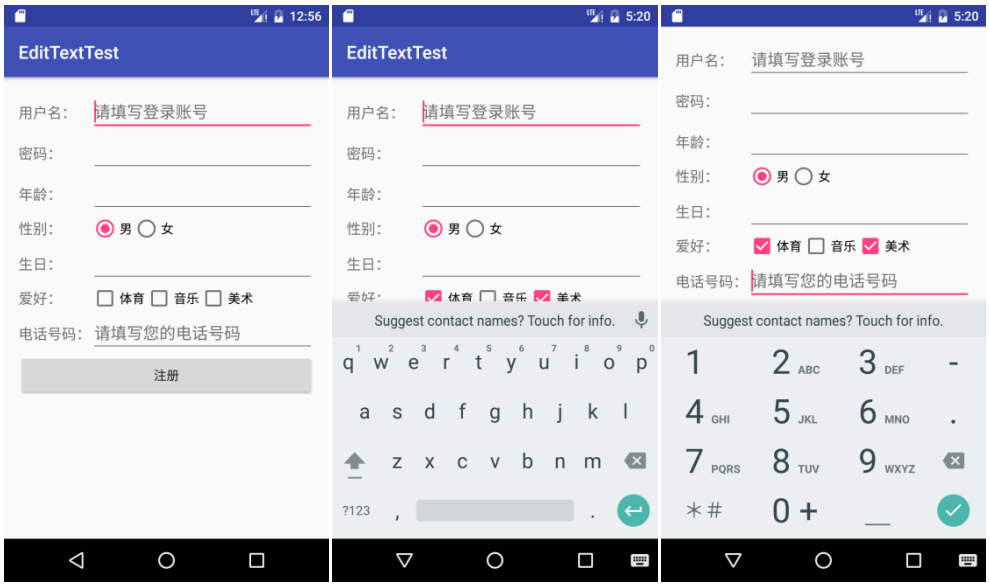
```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
```

```
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:stretchColumns="1"
    tools:context=".MainActivity">

    <TableRow>
        <TextView
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:text="用户名："
            android:textSize="16sp"/>

        <EditText
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:hint="请填写登录账号"
            android:selectAllOnFocus="true" />
    </TableRow>
    <TableRow>
        <TextView
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:text="密码："
            android:textSize="16sp" />
        <!--android:inputType="numberPassword"表明只能接受数字密码。-->
        <EditText
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:inputType="numberPassword"/>
    </TableRow>
    <TableRow>
        <TextView
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:text="年龄："
            android:textSize="16sp" />
        <!--android:inputType="number"表明是数值输入框。-->
        <EditText
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:inputType="number"/>
    </TableRow>
    <TableRow>
```

```
        <TextView
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:layout_gravity="center_vertical"
            android:text="性别："
            android:textSize="16sp" />
        <RadioGroup android:id="@+id/rg"
            android:orientation="horizontal">
            <RadioButton android:id="@+id/male"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="男"
            android:checked="true"/>
            <RadioButton android:id="@+id/female"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="女"/>
        </RadioGroup>
    </TableRow>
    <TableRow>
        <TextView
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:text="生日："
            android:textSize="16sp" />
        <!--android:inputType="date"表明是日期输入框。-->
        <EditText
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:inputType="date"/>
    </TableRow>
    <TableRow>
        <TextView
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:layout_gravity="center_vertical"
            android:text="爱好："
            android:textSize="16sp"/>
        <LinearLayout android:orientation="horizontal"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
            <CheckBox
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
```

```
                android:text="体育"
                android:id="@+id/like1"/>
            <CheckBox
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="音乐"
                android:id="@+id/like2"/>
            <CheckBox
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="美术"
                android:id="@+id/like3"/>
        </LinearLayout>
    </TableRow>
    <TableRow>
        <TextView
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:text="电话号码："
            android:textSize="16sp" />
        <!--android:inputType="phone"表明是电话号码的输入框。-->
        <EditText
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:hint="请填写您的电话号码"
            android:selectAllOnFocus="true"
            android:inputType="phone"/>
    </TableRow>
    <Button
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:text="注册"/>
</TableLayout>
```

运行结果如下图所示：



可以看到当把焦点定位到不同输入类型的编辑框时，系统自动显示对应类型的输入键盘，这就是设置输入类型的作用。

4. 图像视图（ImageView）

图像视图用于在屏幕中显示任何 Drawable 对象，通常用来显示图片。在使用 ImageView 组件显示图像时，通常可以将要显示的图片显示在 res/drawable 目录中。

下面通过一个简单的实验来熟悉 ImageView 的用法：

修改布局文件，添加 4 个 ImageView。其中，第一个用来原尺寸显示图像；第二个设置了高度和宽度，用来实现保持纵横比缩放图片并在右下角显示图片；第三个用来实现图像着色功能，设置的是半透明的红色；第四个限制了最大高度和宽度，允许调整自己的边界来保持图片的长宽比。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView android:id="@+id/img1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/flower"
        android:layout_margin="5dp"/>

    <ImageView android:id="@+id/img2"
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:src="@drawable/flower"
```

```
        android:scaleType="fitEnd"
        android:layout_below="@+id/img1"
        android:layout_margin="5dp" />

    <ImageView android:id="@+id/img4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/flower"
        android:layout_below="@id/img2"
        android:adjustViewBounds="true"
        android:maxHeight="150dp"
        android:maxWidth="150dp"
        android:layout_margin="5dp" />

    <ImageView android:id="@+id/img3"
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:src="@drawable/flower"
        android:tint="#77ff0000"
        android:layout_alignTop="@+id/img2"
        android:layout_toRightOf="@+id/img2"
        android:layout_toEndOf="@+id/img2" />

</RelativeLayout>
```

运行结果如下图所示：

5. 列表视图（ListView）

列表视图以垂直列表的形式显示所有列表项。有两种创建方式：1）直接使用 ListView 进行创建；2）让 Activity 继承 ListActivity。ListView 创建之后，就需要设置它要显示的列表项。列表项可以通过 android:entries 属性来指定，也可以通过 Adapter 来设置，其中，使用 Adapter 可以控制每个列表项的外观和行为。Adapter 本身是一个接口，其常用的实现类有 ArrayAdapter、SimpleAdapter、BaseAdapter 等。

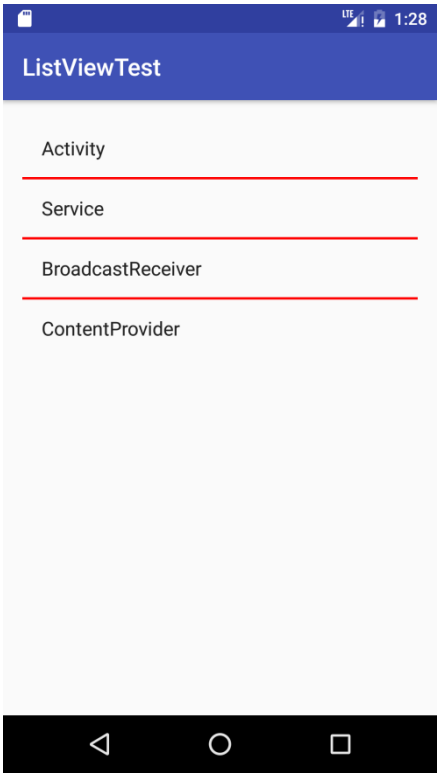
下面是一个基于数组实现的 ListView：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <ListView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:entries="@array/components"
    android:divider="#f00"
    android:dividerHeight="2dp"
    android:headerDividersEnabled="false"/>

</LinearLayout>
```

上面的 ListView 使用到了数组资源，因此还需要在应用中定义一个名为 component 的数组，定义数组的资源文件如下：

```
<?xml version="1.0" encoding="utf-8" ?>
<resources>
    <string-array name="components">
        <item>Activity</item>
        <item>Service</item>
        <item>BroadcastReceiver</item>
        <item>ContentProvider</item>
    </string-array>
</resources>
```

运行结果如下图所示：



下面是一个基于 ArrayAdapter 实现的 ListView：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <ListView
    android:id="@+id/list1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:divider="#f00"
    android:dividerHeight="2dp"
    android:headerDividersEnabled="false"/>

</LinearLayout>
```

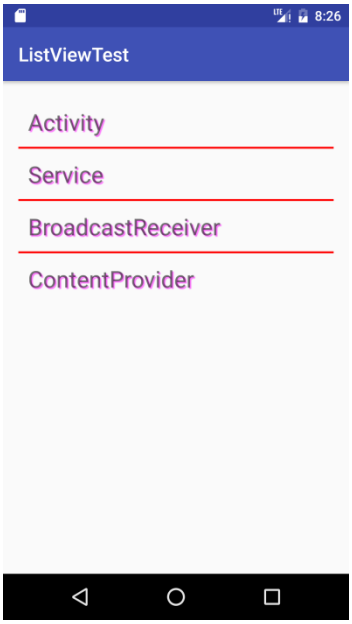
接下来 Activity 为 ListView 提供 Adapter，Adapter 决定 ListView 要显示的列表项。程序如下：

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ListView list1 = (ListView) findViewById(R.id.list1);
        //定义一个数组
        String[] arr = {"Activity", "Service", "BroadcastReceiver", "ContentProvider"};
        //将数组包装为ArrayAdapter
        ArrayAdapter<String> adapter1 = new ArrayAdapter<String>(this, R.layout.array_item,
arr);
        //为ListView设置Adapter
        list1.setAdapter(adapter1);
    }
}
```

上面程序中的 R.layout.array_item 布局文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="24dp"
    android:padding="10dp"
    android:shadowColor="#f0f"
    android:shadowDx="4"
    android:shadowDy="4"
    android:shadowRadius="2"/>
```

运行结果如下图所示：



6. 列表选择框（Spinner）

Android 中提供的列表选择框相当于在网页中常见的下拉列表框，通常提供一系列可选择的列表项供用户进行选择。其中，列表项的设置方法与 ListView 一样，具体情况可以参考 ListView。

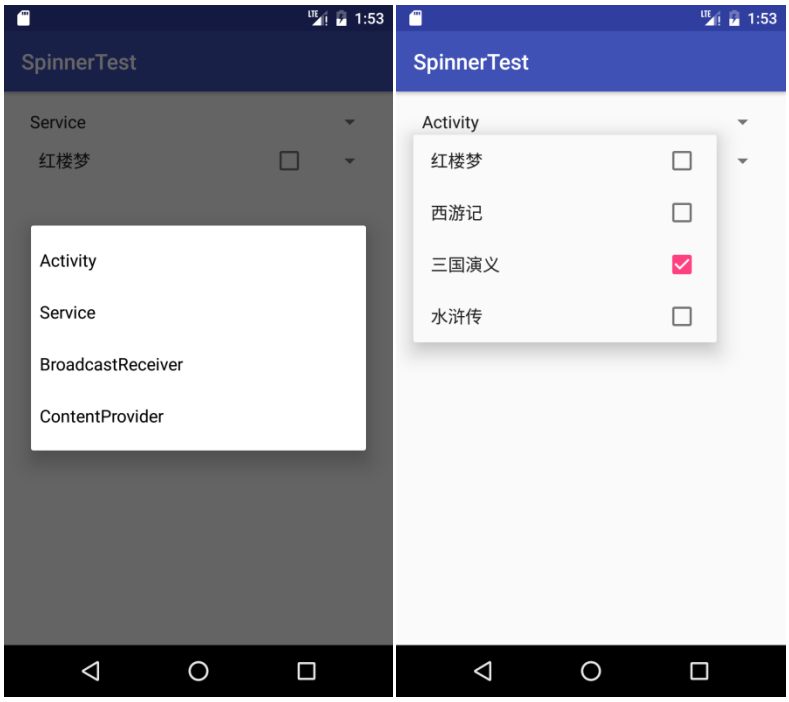
下面是一个简单的使用 Spinner 的实验，布局文件代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <Spinner
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:entries="@array/components"
        android:spinnerMode="dialog" />

    <Spinner
        android:id="@+id/spinner"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

MainActivity.java 文件内容如下：

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.Spinner;

public class MainActivity extends AppCompatActivity {

    Spinner spinner;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //获取布局文件中的 Spinner 组件
        spinner = (Spinner) findViewById(R.id.spinner);
        String[] arr = {"红楼梦","西游记","三国演义","水浒传"};
        //创建 ArrayAdapter 对象
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_multiple_choice, arr);
        //为 Spinner 设置 Adapter
        spinner.setAdapter(adapter);
    }
}
```

运行结果如下图所示：



7. 消息提示框（Toast）

Toast 是一种非常方便的提示消息框，它会在程序界面上显示一个简单的提示信息，这个提示信息不会获得焦点，过一段时间会自动消失。使用 Toast 生成提示消息非常简单，只要如下几个步骤：

1) 调用 Toast 的构造器或 MakeText()静态方法创建一个 Toast 对象；
2) 调用 Toast 的方法来设置提示消息的对齐方式、页边距等；
3) 调用 Toast 的 show()方法将它显示出来。

下面通过 Toast 来实现一个带图片的消息提示，布局文件代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/simple"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="简单提示"/>
    <Button
        android:id="@+id/bn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="带图片的提示" />

</LinearLayout>
```

Java 代码如下：

```
import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
```

```
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button simple = (Button) findViewById(R.id.simple);
        //为按钮的单击事件绑定事件监听器
        simple.setOnClickListener(new View.OnClickListener(){
            public void onClick(View v)
            {
                //创建一个 Toast 提示信息并设置其持续时间
                Toast.makeText(MainActivity.this, "简单的提示信息",
Toast.LENGTH_SHORT).show();
            }
        });

        Button bn = (Button) findViewById(R.id.bn);
        //为按钮的单击事件绑定事件监听器
        bn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //创建一个 Toast 提示消息
                Toast toast = new Toast(MainActivity.this);

                //设置 Toast 的显示位置
                toast.setGravity(Gravity.CENTER, 0, 0);
                //创建一个 ImageView
                ImageView image = new ImageView(MainActivity.this);
                image.setImageResource(R.drawable.tools);
                //创建一个线性布局容器
                LinearLayout ll = new LinearLayout(MainActivity.this);
                //向线性布局容器里添加图片、文本
                ll.addView(image);
                TextView textView = new TextView(MainActivity.this);
                textView.setText("带图片的提示消息");
                textView.setTextSize(24);
                textView.setTextColor(Color.MAGENTA);
                ll.addView(textView);
                //设置 Toast 显示自定义 View
                toast.setView(ll);
                //设置 Toast 显示时间
                toast.setDuration(Toast.LENGTH_LONG);
```

```
                toast.show();
            }
        });
    }
}
```

运行结果如下图所示:

8. 选项卡（TabHost）

选项卡主要由 TabHost、TabWidget 和 FrameLayout 3 个组件组成，用于实现一个多标签页的用户界面，便于分类显示和管理信息。其中，TabHost 是一个标签页容器，TabWidget 是选项卡的标题条，FrameLayout 用来“层叠”组合多个标签页。使用选项卡不仅可以使界面简洁大方，还可以有效地减少窗体的个数。

实现选项卡的一般步骤如下：

1） 在布局文件中添加 TabHost、TabWidget 和 FrameLayout 组件；

2） 编写各标签页要显示内容所对应的 XML 布局文件；

3） 在 Activity 中，获取并初始化 TabHost 组件；

4） 为 TabHost 对象添加标签页。

下面通过选项卡实现一个简单的通话记录界面，布局代码如下。其中，我们需要注意 TabHost、TabWidget 和 FrameLayout 的 ID 不是开发者自己定义的，而是引用了 Android 系统已有的 ID。

```
<?xml version="1.0" encoding="utf-8"?>
<TabHost xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@android:id/tabhost"
```

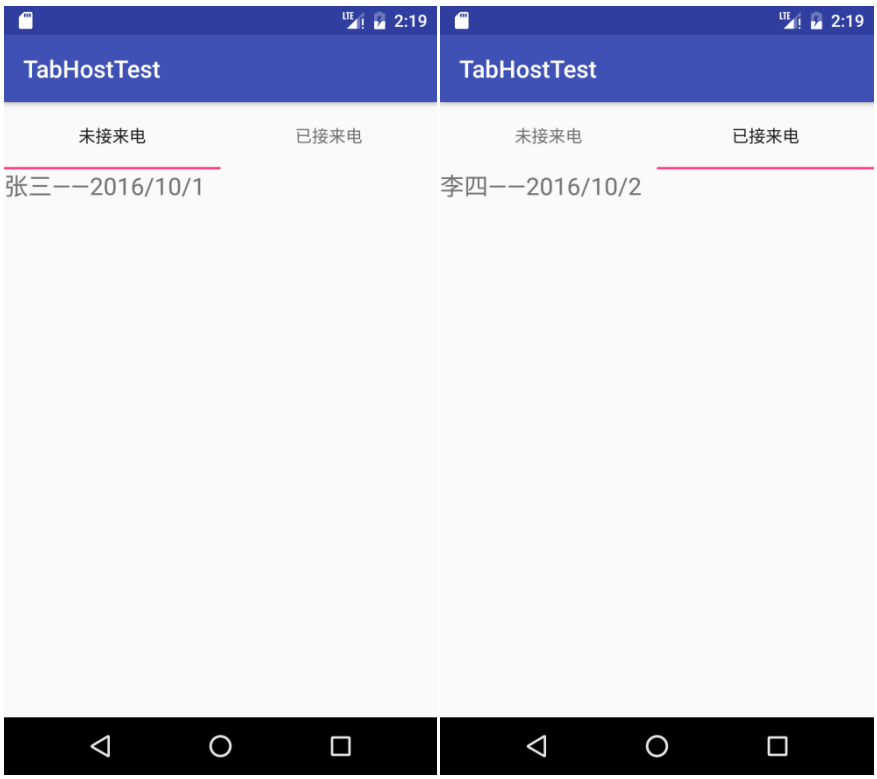
```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
        <TabWidget android:id="@android:id/tabs"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"/>
        <FrameLayout android:id="@android:id/tabcontent"
            android:layout_width="match_parent"
            android:layout_height="match_parent">
            <TextView android:id="@+id/tab01"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="张三——2016/10/1"
                android:textSize="20sp"/>
            <TextView android:id="@+id/tab02"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="李四——2016/10/2"
                android:textSize="20sp"/>
        </FrameLayout>
    </LinearLayout>
</TabHost>
```

MainActivity 的代码如下：

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TabHost;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //获取该 Activity 里面的 TabHost 组件
        TabHost tabHost = (TabHost) findViewById(android.R.id.tabhost);
        //初始化 TabHost 组件
        tabHost.setup();
        //添加第一个标签页
        TabHost.TabSpec tab1 = tabHost.newTabSpec("tab1")
                .setIndicator("未接来电")//设置标题
```

```
                .setContent(R.id.tab01); //设置内容
        tabHost.addTab(tab1);
        //添加第二个标签页
        TabHost.TabSpec tab2 = tabHost.newTabSpec("tab2")
                .setIndicator("已接来电")
                .setContent(R.id.tab02);
        tabHost.addTab(tab2);
    }
}
```

运行结果如下图所示：



9. 对话框（AlertDialog）

AlertDialog 可以生成各种内容的对话框，但实际上 AlertDialog 生成的对话框都可分为 4 个区域：图标区、标题区、内容区和按钮区。创建一个对话框需要经过如下几步：

1） 创建 AlertDialog.Builder 对象；
2） 调用 AlertDialog.Builder 的 setTitle()或 setCustomTitle()方法设置标题；
3） 调用 AlertDialog.Builder 的 setIcon()方法设置图标；
4） 调用 AlertDialog.Builder 的相关设置方法设置对话框内容；
5） 调用 AlertDialog.Builder 的 setPositiveButton()、setNegativeButton()或 setNeutralButton()方法添加多个按钮；
6） 调用 AlertDialog.Builder 的 create()方法创建 AlertDialog 对象，再调用 AlertDialog 对象的 show()方法将该对话框显示出来。

其中，第四步是最灵活的，AlertDialog 提供了如下 6 种方法来指定对话框的内容。

- setMessage():设置内容为简单文本。

- setItems():设置内容为简单列表项。
- setSingleChoiceItems():设置内容为单选列表项。
- setMultiChoiceItems():设置内容为多选列表项。
- setAdapter():设置内容为自定义列表项。
- setView():设置内容为自定义 View。

下面通过一个实验来介绍 AlertDialog 的用法，程序界面定义了 4 个按钮，每次用户单击一个按钮时，就会显示不同类型的对话框。

布局文件代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    android:gravity="center"
    tools:context=".MainActivity">

    <Button
        android:text="简单文本对话框"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button" />

    <Button
        android:text="简单列表对话框"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button2" />

    <Button
        android:text="单选列表对话框"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button3" />

    <Button
        android:text="多选列表对话框"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:id="@+id/button4" />
</LinearLayout>
```

MainActivity.java 代码如下：

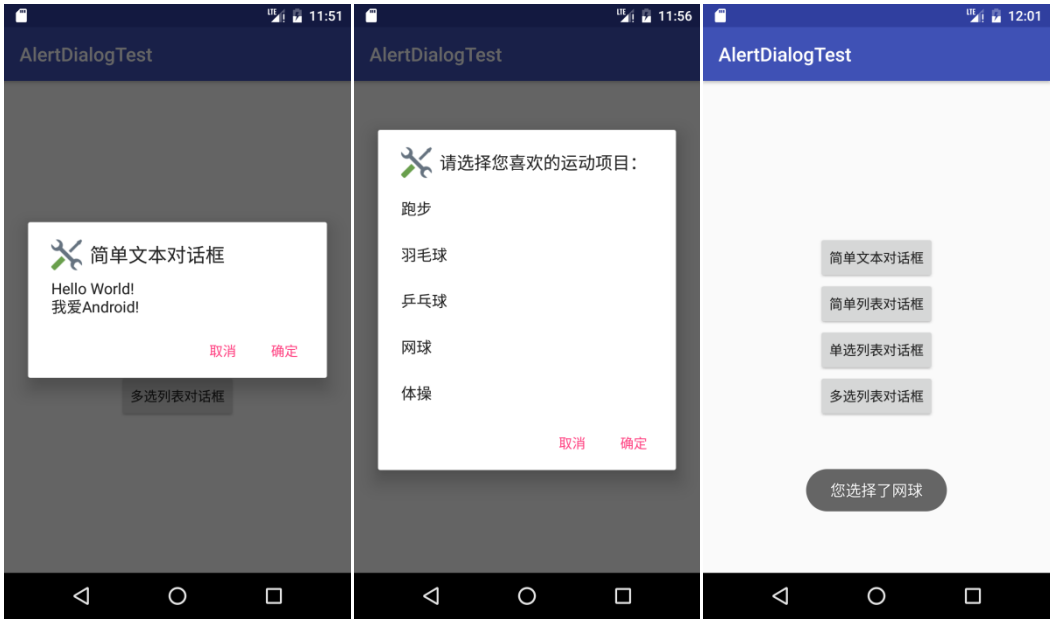
```
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private boolean[] checkedItems;//记录各列表项的状态
    private String[] items;//各列表项要显示的内容

    private AlertDialog.Builder setPositiveButton(AlertDialog.Builder builder)
    {
        return builder.setPositiveButton("确定", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(MainActivity.this, "您单击了【确定】按钮！",
                        Toast.LENGTH_SHORT).show();
            }
        });
    }

    private AlertDialog.Builder setNegativeButton(AlertDialog.Builder builder)
    {
        return builder.setNegativeButton("取消", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(MainActivity.this, "您单击了【取消】按钮！",
                        Toast.LENGTH_SHORT).show();
            }
        });
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

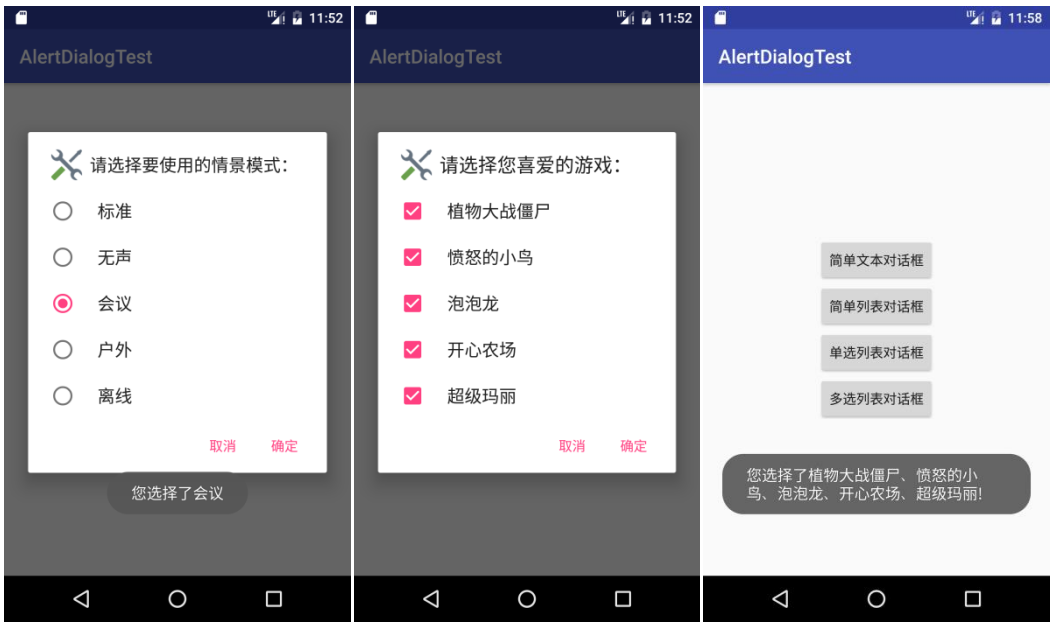
```
        // 简单文本对话框
        Button button = (Button)findViewById(R.id.button);//获取“简单文本对话框”按钮
        //添加单击事件监听器
        button.setOnClickListener(new View.OnClickListener(){
            public void onClick(View v)
            {
                AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
                builder.setTitle("简单文本对话框");
                builder.setIcon(R.drawable.tools);
                builder.setMessage("Hello World!\n我爱Android!");
                setPositiveButton(builder);//添加“确定”按钮
                setNegativeButton(builder);//添加“取消”按钮
                builder.create().show();
            }
        });

        // 简单列表对话框
        Button button2 = (Button) findViewById(R.id.button2); // 获取“简单列表对话框”按钮
        button2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                final String[] items = new String[] { "跑步", "羽毛球", "乒乓球", "网球",
                        "体操" };
                AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
                builder.setIcon(R.drawable.tools); //设置对话框的图标
                builder.setTitle("请选择您喜欢的运动项目："); //设置对话框的标题
                //添加列表项
                builder.setItems(items, new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        Toast.makeText(MainActivity.this,
                                "您选择了" + items[which], Toast.LENGTH_SHORT).show();
                    }
                });
                setPositiveButton(builder);//添加“确定”按钮
                setNegativeButton(builder);//添加“取消”按钮
                builder.create().show(); // 创建对话框并显示
            }
        });

        // 单选列表对话框
        Button button3 = (Button) findViewById(R.id.button3); // 获取“单选列表对话框”按钮
        button3.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {
                final String[] items = new String[] { "标准", "无声", "会议", "户外",
                        "离线" };
                AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
                builder.setIcon(R.drawable.tools); //设置对话框的图标
                builder.setTitle("请选择要使用的情景模式：");  //设置对话框的标题
                builder.setSingleChoiceItems(items, 0, new DialogInterface.OnClickListener() {

                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        Toast.makeText(MainActivity.this,
                                "您选择了" + items[which], Toast.LENGTH_SHORT).show(); //显示
选择结果

                    }
                });
                setPositiveButton(builder);//添加“确定”按钮
                setNegativeButton(builder);//添加“取消”按钮
                builder.create().show(); // 创建对话框并显示
            }
        });

        // 多选列表对话框
        Button button4 = (Button) findViewById(R.id.button4); // 获取“多选列表对话框”按钮
        button4.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                checkedItems= new boolean[] { false, true, false, true, false};    //记录各列
表项的状态
                items = new String[] { "植物大战僵尸", "愤怒的小鸟", "泡泡龙", "开心农场",
                        "超级玛丽" };  //各列表项要显示的内容
                AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
                builder.setIcon(R.drawable.tools); //设置对话框的图标
                builder.setTitle("请选择您喜爱的游戏：");    //设置对话框标题
                builder.setMultiChoiceItems(items, checkedItems,
                        new DialogInterface.OnMultiChoiceClickListener() {

                            @Override
                            public void onClick(DialogInterface dialog,
                                                int which, boolean isChecked) {
                                checkedItems[which]=isChecked; //改变被操作列表项的状态
                            }
                        });
```

```
                //为对话框添加“确定”按钮
                builder.setPositiveButton("确定", new DialogInterface.OnClickListener() {

                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        String result="";  //用于保存选择结果
                        for(int i=0;i<checkedItems.length;i++){
                            if(checkedItems[i]){   //当选项被选择时
                                result+=items[i]+"、";  //将选项的内容添加到 result 中
                            }
                        }
                        //当 result 不为空时，通过消息提示框显示选择的结果
                        if(!"".equals(result)){
                            result=result.substring(0, result.length()-1); //去掉最后面添加的
“、”号
                            Toast.makeText(MainActivity.this, "您选择了"+result+"!",
Toast.LENGTH_LONG).show();
                        }
                    }
                });
                setNegativeButton(builder);//添加“取消”按钮
                builder.create().show(); // 创建对话框并显示
            }
        });
    }
}
```

运行结果如下图所示：




11:52
AlertDialogTest
请选择要使用的情景模式：
标准
无声
会议
户外
离线
取消
确定
您选择了会议
11:52
AlertDialogTest
请选择您喜爱的游戏：
植物大战僵尸
愤怒的小鸟
泡泡龙
开心农场
超级玛丽
取消
确定
11:58
AlertDialogTest
简单文本对话框
简单列表对话框
单选列表对话框
多选列表对话框
您选择了植物大战僵尸、愤怒的小鸟、泡泡龙、开心农场、超级玛丽!