第5章 对话框、菜单和状态栏通知

作为用户交互的重要工具和手段，对话框、菜单和状态栏通知在UI设计中发挥着重要的作用。对话框是一种浮于Activity之上的界面元素，一般用于给出提示信息或弹出一个与主进程直接相关的子程序；菜单能够在不占用界面空间的前提下为应用程序提供相应的功能和界面；状态栏通知是一种具有全局效果的提醒机制，不会打断用户当前的操作。本章将介绍这些界面元素的使用方法。

5.1 对 话 框

在Android中，对话框是提示用户做出决定或输入额外信息的小窗口。对话框不会填充屏幕，通常用于需要用户采取行动才能继续执行的情形。当显示对话框时，当前Activity失去焦点而由对话框负责所有的交互。常用的对话框有提示对话框(AlertDialog)、日期选择对话框(DatePickerDialog)、时间选择对话框(TimePickerDialog)等，其中AlertDialog是最常用的对话框。这些对话框都是android.app.Dialog的子类。

5.1.1 提示对话框

AlertDialog是一个消息提示对话框，能构造默认的3个按钮，分别用于肯定、否定和中立。创建AlertDialog的主要步骤如下。

步骤1：获得AlertDialog的静态内部类Builder对象，并由该对象来创建对话框。

步骤2：通过Builder对象设置对话框的标题、文字等属性。表5-1列出了Builder对象的部分常用方法。

表5-1 Builder对象的部分常用方法

方 法 名	说 明
setIcon()	设置显示在对话框标题左侧的图标
setTitle()	设置对话框的标题
setMessage()	设置对话框的提示文字
setItems()	设置对话框显示一个列表

续表

方 法 名	说 明
setSingleChoiceItems()	设置对话框显示一个单选的选项列表
setMultiChoiceItems()	设置对话框显示一系列的复选框
setPositiveButton()	使肯定按钮可见并为其设置 Click 事件监听器
setNegativeButton()	使否定按钮可见并为其设置 Click 事件监听器
setNeutralButton()	使中立按钮可见并为其设置 Click 事件监听器
setView()	设置一个自定义的 View 对象作为对话框的显示内容。调用该方法可以在对话框中显示一个布局或 UI 对象
setCancelable()	设置对话框是否能被取消
create()	创建 AlertDialog 对象,该方法的返回值为 AlertDialog 对象
show()	创建 AlertDialog 对象并显示对话框,该方法的返回值为 AlertDialog 对象

步骤 3:设置对话框的按钮以及点击按钮的事件响应处理程序。

对话框中可以有肯定、否定和中立 3 个按钮。一般用户使用肯定按钮来接受并继续执行操作,如确定操作;使用否定按钮来取消操作;中立按钮则用于用户不想继续执行操作,但也未必想要取消操作的情况,如“稍后提醒我”的操作。生成器 Builder 负责设置对话框上的按钮,并为按钮注册 OnClickListener 监听器。OnClickListener 在 android.content. DialogInterface 包中,事件处理时回调 onClick()方法。无论用户点击哪个按钮,对话框都会关闭,并导致接口中的 onClick()方法被调用。onClick()方法的定义如下:

```
abstract void onClick(DialogInterface dialog, int which)
```

其中,参数 dialog 就是当前要关闭的对话框,参数 which 是用户点击的按钮,其取值可以是 DialogInterface. BUTTON_NEGATIVE、DialogInterface. BUTTON_NEUTRAL 或 DialogInterface.BUTTON_POSITIVE。

步骤 4:调用 Builder 对象的 show()方法创建并显示对话框。或者调用 Builder 对象的 create()方法创建 AlertDialog 对象,再调用 AlertDialog 对象的 show()方法显示对话框。

如果不希望用户点击设备的“返回”按钮或在对话框外部点击使对话框消失,而是要求用户必须点击对话框中的按钮,可以通过调用 AlertDialog 对象的 setCancelable(false)方法来进行设置。

1. 简单提示对话框

简单提示对话框仅包括文字标题、标题左侧的图标、文字提示信息、按钮等基本元素。

【例 5-1】　示例工程 Demo_05_AlertDialog 演示了 AlertDialog 对话框的用法。

在主界面中有“弹出 AlertDialog 对话框”按钮，点击该按钮后弹出 AlertDialog，如图 5-1 所示，主要代码如代码段 5-1 所示。

AlertDialog

图 5-1　AlertDialog 提示

代码段 5-1　创建简单的 AlertDialog

```
btnStart.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        AlertDialog.Builder myDialog=new AlertDialog.Builder(MainActivity.
           this);                                    //创建 AlertDialog.Builder 对象
        myDialog.setIcon(R.mipmap.ic_launcher);          //设置图标
        myDialog.setTitle("提示");                        //设置标题
        myDialog.setMessage("这是一个 AlertDialog!");     //设置消息内容
        myDialog.setNegativeButton("取消", null);        //否定按钮
        myDialog.setNeutralButton("稍后提醒我", null);    //中立按钮
        myDialog.setPositiveButton("确定", new DialogInterface.
OnClickListener() {                                      //肯定按钮
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(getApplicationContext(), "您点击了“确定”按
                  钮!", Toast.LENGTH_LONG).show();
            }
        });
        myDialog.show();                                 //显示对话框
    }
});
```

2. 其他风格的提示对话框

除了按钮对话框，还可以创建列表对话框、单选对话框、多选对话框。通过调用 setItems()方法，可以在 AlertDialog 中添加列表项；通过调用 setSingleChoiceItems()/setMultiChoiceItems()方法，可以在 AlertDialog 中添加单选或多选列表。

【例 5-2】 示例工程 Demo_05_ListDialog 演示了列表对话框的用法。

在 Activity 中点击“弹出列表对话框”按钮后,弹出 AlertDialog,运行效果如图 5-2 所示。当在对话框中选择了一项之后,Activity 中 TextView 的显示内容随之更新。

图 5-2 列表对话框

首先在 values 下 arrays.xml 文件中定义字符串数组,内容如代码段 5-2 所示。创建列表对话框的主要代码如代码段 5-3 所示。

代码段 5-2 定义字符串数组

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="favor">
        <item>音乐</item>
        <item>体育</item>
        <item>美术</item>
    </string-array>
</resources>
```

代码段 5-3 创建列表对话框

```
btnStart.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        AlertDialog.Builder myDialog=new AlertDialog.Builder(MainActivity.
            this);
        myDialog.setTitle("列表对话框,请选择")     //设置对话框的标题
                .setItems(                          //用字符串数组设置列表中的各个属性
                    R.array.favor, new DialogInterface.OnClickListener() {
                                                    //设置监听器
                        public void onClick(DialogInterface dialog, int which) {
                            TextView message=(TextView) findViewById(R.id.text1);
                            message.setText("您选择了:" + getResources().
                                getStringArray(R.array.favor)[which]);
```

```
                        }
                    })
                .setNeutralButton("取消", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
                        Toast.makeText(getApplicationContext(), "您取消了选
                            择", Toast.LENGTH_SHORT).show();
                    }
                })
                .show();
        }
    });
```

【例 5-3】　示例工程 Demo_05_RadioButtonDialog 演示了单选对话框的用法。

本例使用与例 5-2 示例工程中相同的数组资源，当点击“弹出单选对话框”按钮后，弹出 AlertDialog，运行效果如图 5-3 所示。创建单选对话框的主要代码如代码段 5-4 所示。

图 5-3　单选对话框

代码段 5-4　创建单选对话框

```
btnStart.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        AlertDialog.Builder myDialog=new AlertDialog.Builder(MainActivity.
this);
        myDialog.setTitle("单选对话框,请选择")        //设置对话框的标题
                .setSingleChoiceItems(                //设置单选按钮选项
                        R.array.favor, selectedItem, new DialogInterface.
OnClickListener() {
                            public void onClick(DialogInterface dialog, int which) {
                                selectedItem=which;
                            }
```

```
                        });
            myDialog.setPositiveButton("确定", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    TextView message=(TextView) findViewById(R.id.text1);
                    message.setText("您选择了:"+getResources().
getStringArray(R.array.favor)[selectedItem]);
                }
            });
            myDialog.setNegativeButton("取消", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                  Toast.makeText(getApplicationContext(), "您取消了选择", Toast
.LENGTH_SHORT).show();
                }
            });
            myDialog.show();
        }
    });
```

3. 具有复杂界面的提示对话框

通过调用 AlertDialog.Builder 对象的 setView()方法,可以将自定义的 View 或布局添加至 AlertDialog。这样就可以实现在对话框中显示较复杂的内容。在默认情况下,自定义布局会填充整个对话框。

具有复杂界面的提示对话框

【例 5-4】 示例工程 Demo_05_ViewDialog 演示了如何将自定义的 View 作为其内容显示在对话框中,该程序通过点击“弹出登录对话框”按钮来弹出一个用来显示登录界面的对话框。

为了实现上述功能,需要为对话框设计相应的布局。在 res\layout 下创建一个 XML 布局文件 dialog_view.xml,布局的主要内容是提示文字及对应的文本输入框,分别用于输入用户名和密码,其主要代码如代码段 5-5 所示。对话框中的“登录”“注册”“退出”按钮不需要在布局文件中设定,而是在 Builder 被实例化后通过调用 setPositiveButton()、setNegativeButton()和 setNeutralButton()方法来添加,并在其中分别设定 3 个按钮对应的事件处理程序。

代码段 5-5 dialog_view.xml 布局文件

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

```
        android:padding="24dp" >
        <TextView
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:textSize="18sp"
            android:text="用户名:" />
        <EditText
            android:id="@+id/ed_username"
            android:layout_height="wrap_content"
            android:layout_width="match_parent"/>
        <TextView
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_marginTop="24dp"
            android:textSize="18sp"
            android:text="密码:"/>
        <EditText
            android:id="@+id/ed_password"
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:inputType="textPassword"/>
    </LinearLayout>
```

示例工程中，点击“弹出登录对话框”按钮后弹出对话框，如图 5-4 所示。

图 5-4　具有复杂界面的提示对话框

如果主界面中按钮被点击，则定义一个 LayoutInflater 类的实例。LayoutInflater 类的作用类似于 findViewById()，不同点是 LayoutInflater 是用来引入 Layout 下的 XML 布局文件并且实例化，而 findViewById()是引入 XML 文件中定义的具体 UI 对象（如

Button、TextView 等)。这里通过调用 LayoutInflater 实例的 inflate()方法引入 XML 布局文件 dialog_view.xml,然后通过调用 Builder 对象的 setView(View)方法,在对话框中加载这个布局文件。创建对话框的主要代码如代码段 5-6 所示。

代码段 5-6 定义对话框及其按钮功能

```
btnStart.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        LayoutInflater dialogInflater=LayoutInflater.from(MainActivity.this);
        final View myViewOnDialog=dialogInflater.inflate(R.layout.dialog_
            view, null);                                    //引入布局
        AlertDialog.Builder myDialogInstance=new AlertDialog.Builder
            (MainActivity.this)
              .setIcon(R.mipmap.ic_launcher)                //对话框的标题图标
              .setTitle("用户登录")
              .setView(myViewOnDialog)                      //参数是上面定义的 View 实例
                                                            //名,显示 dialog_view 布局
              .setPositiveButton("登录", new DialogInterface.OnClickListener() {
                                                            //肯定按钮
                  public void onClick(DialogInterface dialog, int whichButton) {
                                                            //监听点击事件
                      EditText editText=(EditText)myViewOnDialog.
                          findViewById(R.id.ed_username);
                      Toast.makeText(getApplicationContext(), "您的用户名是"+
                          editText.getText(), Toast.LENGTH_LONG).show();
                  }
              })
              .setNegativeButton("注册", new DialogInterface.OnClickListener() {
                                                            //否定按钮
                  public void onClick(DialogInterface dialog, int whichButton) {
                                                            //监听点击事件
                      Toast.makeText(getApplicationContext(), "欢迎您注册为新
                          用户", Toast.LENGTH_LONG).show();
                  }
              })
              .setNeutralButton("退出", new DialogInterface.OnClickListener() {
                                                            //中立按钮
                  @Override
                  public void onClick(DialogInterface dialogInterface, int i) {
                      MainActivity.this.finish();  //退出程序
                  }
              });
        myDialogInstance.show();                            //显示对话框
    }
});
```

5.1.2　日期和时间选择对话框

DatePickerDialog 和 TimePickerDialog 用来显示日期选择和时间选择对话框。可以在程序中直接通过 new 的方式实例化这两个类来得到对话框对象，二者的使用方法非常类似。

对于 DatePickerDialog，其常用的构造方法定义如下：

```
DatePickerDialog(Context con, OnDateSetListener call, int year, int month,
int day)
```

该方法有 5 个参数，其中第 2 个参数是一个 DatePickerDialog.OnDateSetListener 对象，当用户选择好日期点击“确定”按钮时，会调用其中的 onDateSet()方法。最后 3 个参数分别用于指定对话框弹出时初始选择的年、月、日。

TimePickerDialog 类常用的构造方法定义如下：

```
TimePickerDialog(Context con, OnTimeSetListener call, int h, int m, boolean
is24Hour)
```

该方法有 5 个参数，其中第 2 个参数是一个 TimePickerDialog.OnTimeSetListener 对象，当用户选择好时间点击对话框上的“确定”按钮时，会调用其中的 onTimeset()方法。第 3 个参数和第 4 个参数为弹出对话框时初始显示的小时和分钟，最后一个参数设置是否以 24 时制显示时间。

【例 5-5】 示例工程 Demo_05_DateAndTimePickerDialog 演示了 DatePickerDialog 和 TimePickerDialog 的用法。


日期和时间选择对话框

在主界面的布局中定义两个按钮，点击第 1 个按钮，则弹出日期选择对话框，如图 5-5(a)所示。点击“确定”按钮关闭对话框后，利用 Toast 显示所选择的日期。点击第 2 个按钮，

(a) 日期选择对话框　　(b) 时间选择对话框

图 5-5　示例工程的运行结果

弹出时间选择对话框,如图 5-5(b)所示。点击“确定”按钮关闭对话框后,利用 Toast 显示所选择的时间。如果想要使弹出对话框时初始选择某一个日期或时间,可以利用 java.util.Calendar 类获取日期或时间。

MainActivity 类的主要代码如代码段 5-7 所示。

代码段 5-7　MainActivity 类的主要代码

```
//package 和 import 语句略
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btn1=(Button)findViewById(R.id.btn_1);
        btn1.setOnClickListener(new View.OnClickListener() {
                                                //"DatePickerDialog 示例"按钮
            public void onClick(View v) {
                //创建一个 DatePickerDialog
                DatePickerDialog datePickerDialog=new DatePickerDialog
                    (MainActivity.this,
                   new DatePickerDialog.OnDateSetListener() {
                     public void onDateSet(DatePicker view, int year, int
                         monthOfYear, int dayOfMonth)  {
                       Toast.makeText(getApplicationContext(), "您选择的日
                           期:"+year+"-"+(monthOfYear+1)+"-"+dayOfMonth,
                       Toast.LENGTH_LONG).show();
                     }
                   }, 2020, 10, 6);
                   datePickerDialog.show();    //显示 DatePickerDialog
            }
        });
        Button btn2=(Button)findViewById(R.id.btn_2);
        btn2.setOnClickListener(new View.OnClickListener() {
                                                //"TimePickerDialog 示例"按钮
            public void onClick(View v) {
                //创建一个 TimePickerDialog
                TimePickerDialog timePickerDialog = new TimePickerDialog
                    (MainActivity.this,
                  new TimePickerDialog.OnTimeSetListener() {
                    public void onTimeSet(TimePicker view, int hourOfDay,
                        int minute)  {
                        Toast.makeText(getApplicationContext(), "您选择的时
                            间: "+hourOfDay+":"+minute, Toast.LENGTH_LONG).
                        show();
```