第3章 常用的UI组件介绍

本章学习目标

- 掌握本章中讲解的 Android 系统中常用的 UI 组件。
- 掌握对话框的具体使用方法。
- 理解 ProgressBar 及其子类的概念。

实际开发中会经常使用 UI 组件来组合项目的界面，特殊的组件可以通过第 2 章中自定义的 UI 组件来绘制。通过对本章节的学习，掌握常用 UI 组件的用法。

3.1 菜　　单

Android 中的菜单(menu)在桌面应用方面十分广泛，几乎所有的桌面应用都会使用到菜单。但在具体应用中却没有那么多菜单，即便如此，它也是很重要的。Android 应用中的菜单分为 3 种：选项菜单(OptionMenu)、上下文菜单(ContextMenu)、弹出式菜单(PopupMenu)。本节依次介绍这些内容。

3.1.1 选项菜单

从 Android 3.1 开始，Android 引入了全新的操作栏，扩展了很多功能，例如安置菜单选项、配置应用图标作为导航按钮等。

可显示在操作栏上的菜单称为选项菜单。选项菜单提供了一些选项，用户选择后可进行相应的操作。

一般为 Android 应用添加选项菜单的步骤如下：

(1) 重写 Activity 的 onCreateOptionsMenu(Menu menu)方法，在该方法里调用 Menu 对象的方法添加菜单项。

(2) 如果想要引用程序响应菜单项的点击事件，就要继续重写 Activity 的 onOptionsItemSelected(MenuItem mi)方法。

添加菜单项的方式与 UI 组件的方式一样，可以在代码中使用，也可以在 XML 布局文件中使用。Android 同样推荐在 XML 中使用菜单，具体为在 app\src\main\res 文件夹中创建名称为 menu 的文件夹，创建完成之后在 menu 文件夹中新建根标签为 menu 的布局文件，具体如例 3.1 所示。

例 3.1 XML 文件中的选项菜单 options_menu.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item android:id="@+id/menu_item1"
          android:title="第一个菜单项"/>
    <item android:id="@+id/menu_item2"
          android:title="第二个菜单项"/>
    <item android:id="@+id/menu_item3"
          android:title="第三个菜单项"/>
</menu>
```

菜单定义完成之后需要在代码中使用才可以看到效果,Java 代码如下所示。

```
1  public class MainActivity extends AppCompatActivity {
2      @Override
3      protected void onCreate(Bundle savedInstanceState) {
4          super.onCreate(savedInstanceState);
5          setContentView(R.layout.activity_main);
6      }
7      @Override
8      public boolean onCreateOptionsMenu(Menu menu) {
9          getMenuInflater().inflate(R.menu.option_menu, menu);
10         return true;
11     }
12     @Override
13     public boolean onOptionsItemSelected(MenuItem item) {
14         switch (item.getItemId()) {
15             case R.id.menu_item1:
16                 Toast.makeText(MainActivity.this,
17                     "第一个菜单项", Toast.LENGTH_LONG).show();
18                 break;
19             case R.id.menu_item2:
20                 Toast.makeText(MainActivity.this,
21                     "第二个菜单项", Toast.LENGTH_LONG).show();
22                 break;
23             case R.id.menu_item3:
24                 Toast.makeText(MainActivity.this,
25                     "第三个菜单项", Toast.LENGTH_LONG).show();
26                 break;
27         }
28         return true;
29     }
30 }
```

上述程序中第 8 行和第 13 行代码是显示菜单和响应菜单点击事件的两个方法。这样

就实现了简单的选项菜单，程序运行结果如图 3.1 所示。

图 3.1　选项菜单运行结果

一个简单的选项菜单示例就完成了。下面来分析 Menu 的组成结构。

Menu 接口是一个父接口，该接口下实现了两个子接口。

- SubMenu：代表一个子菜单，可包含 1～*N* 个 MenuItem(形成菜单项)。
- ContextMenu：代表一个上下文菜单，可包含 1～*N* 个 MenuItem。

Menu 接口定义了 add()方法用于添加菜单项，addSubMenu()方法用于添加子菜单项。有好几个重载方法可供选择，使用时可根据需求选择。SubMenu 继承自 Menu，它额外提供了 setHeaderIcon()、setHeaderTitle()和 setHeaderView()方法，分别用于设置菜单头的图标、标题以及菜单头。

这些方法的使用暂不举例讲解，希望读者自行练习。

3.1.2　上下文菜单

3.1.1 节讲到，ContextMenu 继承自 Menu，开发上下文菜单与开发选项菜单基本类似，区别在于：开发上下文菜单是重写 onCreateContextMenu(ContextMenu menu, View source, ContextMenu.ContextMenuInfo menuInfo)方法，其中 source 参数代表触发上下文菜单的组件。

开发上下文菜单的步骤如下：

(1) 重写 Activity 的 onCreateContextMenu()方法。

(2) 调用 Activity 的 registerForContextMenu(View view)方法为 view 注册上下文菜单。

(3) 如果想实现点击事件，需要重写 onContextItemSelected(MenuItem mi)方法。

与 3.1.1 节提到的 SubMenu 相似，ContextMenu 也提供了 setHeaderIcon()与 setHeaderTitle()方法为 ContextMenu 设置图标和标题。

接下来实现一个简单的 ContextMenu 示例，该示例的功能是长按文字，然后出现可供改变文字背景色的上下文菜单，如例 3.2 所示。

例 3.2　XML 文件中的上下文菜单 context_menu.xml

```
<menu
xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/red"
          android:title="红色"/>
    <item android:id="@+id/black"
          android:title="黑色"/>
    <item android:id="@+id/blue"
          android:title="蓝色"/>
</menu>
```

在 Java 代码 MainActivity.java 中添加上下文菜单。

```
1  package com.example.chapater3_2;
2  import androidx.appcompat.app.AppCompatActivity;
3  import android.graphics.Color;
4  import android.os.Bundle;
5  import android.view.ContextMenu;
6  import android.view.MenuItem;
7  import android.view.View;
8  import android.widget.TextView;
9
10 public class MainActivity extends AppCompatActivity {
11     private TextView textView;
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_main);
16         textView = (TextView) findViewById(R.id.my_text);
17         registerForContextMenu(textView);
18     }
19
20     @Override
21     public void onCreateContextMenu(ContextMenu menu, View v,
22                                     ContextMenu.ContextMenuInfo menuInfo) {
23         getMenuInflater().inflate(R.menu.context_menu, menu);
24         menu.setGroupCheckable(0, true, true);
25         menu.setHeaderTitle("选择背景颜色");
26     }
27
28     @Override
29     public boolean onContextItemSelected(MenuItem item) {
30         switch (item.getItemId()) {
31             case R.id.red:
32                 item.setChecked(true);
33                 textView.setBackgroundColor(Color.RED);
34                 break;
35             case R.id.black:
36                 item.setChecked(true);
37                 textView.setBackgroundColor(Color.BLACK);
38                 break;
39             case R.id.blue:
40                 item.setChecked(true);
41                 textView.setBackgroundColor(Color.BLUE);
42                 break;
43         }
```

```
44          return true;
45      }
46
47      @Override
48      protected void onDestroy() {
49          super.onDestroy();
50          unregisterForContextMenu(textView);
51      }
52  }
```

上述 Java 代码中重写了 onCreateContextMenu()与 onContextItemSelected()方法，分别用于实现加载上下文菜单、菜单的点击事件，代码中两处的粗体部分，分别是注册和解绑上下文菜单，可能读者会疑惑为什么要在 onDestroy()中解绑，在后文讲解 Activity 时一并讲解。程序运行结果如图 3.2 所示。

图 3.2　上下文菜单运行结果

上下文菜单需长按注册的组件才能出现，这一点和选项菜单不同。希望读者认真练习例 3.2 中的代码。

3.1.3　弹出式菜单

默认情况下，PopupMenu 会在指定组件的上方或下方弹出。PopupMenu 可增加多个菜单项，并可为菜单项增加子菜单。

使用 PopupMenu 的步骤与前两种 Menu 不同，步骤如下：

(1) 调用 new PopupMenu(Context context, View anchor)创建下拉菜单，anchor 代表要激发弹出菜单的组件。

(2) 调用 MenuInflater 的 inflate()方法将菜单资源填充到 PopupMenu 中。

(3) 调用 PopupMenu 的 show()方法显示弹出式菜单。

接下来，通过例 3.3 所示学习弹出式菜单。

例 3.3　XML 文件中的弹出式菜单 popup_menu.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <menu xmlns:android="http://schemas.android.com/apk/res/android">
3       <item
4           android:id="@+id/check"
5           android:title="查找" />
6       <item
7           android:id="@+id/add"
8           android:title="添加" />
```

```
    <item
        android:id="@+id/write"
        android:title="编辑" />
    <item
        android:id="@+id/hide"
        android:title="隐藏菜单" />
</menu>
```

界面布局文件中只有一个 Button,在 Button 标签下直接设置点击事件 popupMenuClick,代码如下所示。

```
<?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:gravity="center">

    <Button
        android:id="@+id/my_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="popupMenuClick"
        android:text="PopupMenu 菜单"
        android:textSize="20dp" />
</LinearLayout>
```

Java 代码如下所示。

```
package com.example.chapater3_3;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;
import android.widget.PopupMenu;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    private PopupMenu popupMenu;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

```

```
    public void popupMenuClick(View view) {
        popupMenu = new PopupMenu(this, view);
        getMenuInflater().inflate(R.menu.popup_menu, popupMenu.getMenu());
        popupMenu.setOnMenuItemClickListener(
                new PopupMenu.OnMenuItemClickListener() {
                    @Override
                    public boolean onMenuItemClick(MenuItem item) {
                        switch (item.getItemId()) {
                            case R.id.hide:
                                popupMenu.dismiss();
                                break;
                            default:
                                Toast.makeText(MainActivity.this, "点击了" +
item.getTitle(), Toast.LENGTH_LONG).show();
                        }
                        return true;
                    }
                });
        popupMenu.show();
    }
}
```

上述程序中创建了一个 PopupMenu 对象，通过 inflate 将 popup_menu 菜单资源填充入 PopupMenu 中，可实现当用户点击界面组件时弹出 PopupMenu。

运行程序，点击界面中的 Button 控件，可看到如图 3.3 所示的界面。

图 3.3　弹出式菜单运行结果

前两种菜单的创建非常相似，只有弹出式菜单创建比较特殊。在实际开发中这 3 种菜单会经常使用，希望读者能动手练习并掌握其用法。讲解完使用方式之后，下面再来看一个小知识点：在菜单项中启动另外一个 Activity(或 Service)。

3.1.4　设置与菜单项关联的 Activity

在实际开发中会经常碰到这样一种情况：点击某个菜单项后，跳转到另外一个 Activity(或者 Service)。对于这种需求，Menu 中也有直接的方法可以使用。用户只需要调用 MenuItem 的 setIntent(Intent intent)方法就可以把该菜单项与指定的 Intent 关联到一起，当用户点击该菜单项时，该 Intent 所包含的组件就会被启动。

接下来,将示范调用该方法启动一个 Activity 的例子,如例 3.4 所示。由于该程序几乎不包含任何界面组件,因此不展示界面布局文件。

例 3.4 使用 Menu 自带的 setIntent()方法启动 Activity

```
1  ppackage com.example.chapater3_4;
2  import androidx.appcompat.app.AppCompatActivity;
3  import android.content.Intent;
4  import android.os.Bundle;
5  import android.view.Menu;
6  import android.view.MenuItem;
7
8  public class MainActivity extends AppCompatActivity {
9      @Override
10     protected void onCreate(Bundle savedInstanceState) {
11         super.onCreate(savedInstanceState);
12         setContentView(R.layout.activity_main);
13     }
14
15     @Override
16     public boolean onCreateOptionsMenu(Menu menu) {
17         getMenuInflater().inflate(R.menu.option_menu, menu);
18         return super.onCreateOptionsMenu(menu);
19     }
20
21     @Override
22     public boolean onOptionsItemSelected(MenuItem item) {
23         switch (item.getItemId()) {
24             case R.id.menu_item1:
25                 item.setIntent(new Intent(MainActivity.this,
                   HelloWorldActivity.class));
26                 break;
27         }
28         return super.onOptionsItemSelected(item);
29     }
30 }
```

上述程序中的第 25 行代码就是启动 HelloWorldActivity,需要注意的是 HelloWorldActivity 是一个新的 Activity 文件,代表一个用户交互界面。在创建了 HelloWorldActivity 文件后,要在项目的核心清单配置文件 AndroidManifest.xml 中注册 HelloWorldActivity。程序运行结果如图 3.4 所示。

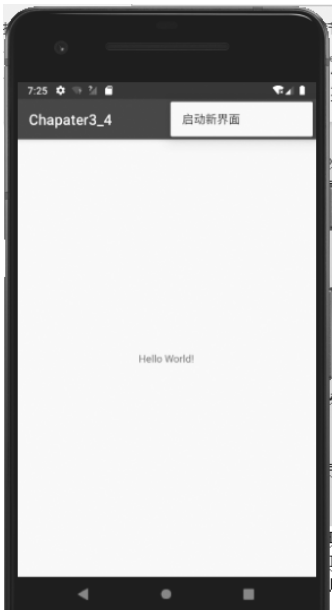

图 3.4　利用菜单选项启动 Activity 界面

3.2　对话框的使用

在日常的 App 使用中经常看到对话框，可以说对话框的出现使得 App 不再那么单调。Android 中提供了丰富的对话框支持，日常开发中会经常使用以下 4 种对话框，如表 3.1 所示。

表 3.1　4 种对话框

对　话　框	说　　明
AlertDialog	功能最丰富，实际应用最广的对话框
ProgressDialog	进度对话框，只用来显示进度条
DatePickerDialog	日期选择对话框，只用来选择日期
TimePlckerDialog	时间选择对话框，只用来选择时间

3.2.1　使用 AlertDialog 建立对话框

AlertDialog 是上述 4 种对话框中功能最强大、用法最灵活的一种，同时它也是其他 3 种对话框的父类。

使用 AlertDialog 生成的对话框样式多变，但是基本样式总包含 4 个区域：图标区、标题区、内容区和按钮区。

创建一个 AlertDialog 一般需要如下几个步骤：

(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。

AlertDialog 的样式多变,就是因为设置对话框内容时的样式多变,AlertDialog 提供了 6 种方法设置对话框的内容,如表 3.2 所示。

表 3.2 AlertDialog 中的方法

方 法	说 明
setMessage()	设置对话框内容为简单文本
setItems()	设置对话框内容为简单列表项
setSingleChoiceItems()	设置对话框内容为单选列表项
setMultiChoiceItems()	设置对话框内容为多选列表项
setAdapter()	设置对话框内容为自定义列表项
setView()	设置对话框内容为自定义 View

接下来,通过以下示例来深入理解 AlertDialog 中的方法,具体如例 3.5～例 3.9 所示。

例 3.5 简单对话框示例

```
1   public void simpleAlertDialog(View view) {
2           AlertDialog.Builder builder = new AlertDialog.Builder(this)
3                   .setTitle("简单对话框")
4                   .setIcon(R.drawable.icon_dialog)
5                   .setMessage("第一行内容\n第二行内容");
6                   setPositiveButton(builder);
7                   setNegativeButton(builder)
8                   .create()
9                   .show();
10      }
```

上述程序是在布局文件中设置 Button 的点击事件为 simpleAlertDialog,具体代码如下所示。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       xmlns:tools="http://schemas.android.com/tools"
```

```
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       tools:context=".MainActivity">
8
9       <Button
10          android:id="@+id/alert_dialog"
11          android:layout_width="wrap_content"
12          android:layout_height="wrap_content"
13          android:onClick="simpleAlertDialog"
14          android:text="@string/alert_dialog"
15          app:layout_constraintBottom_toBottomOf="parent"
16          app:layout_constraintLeft_toLeftOf="parent"
17          app:layout_constraintRight_toRightOf="parent"
18          app:layout_constraintTop_toTopOf="parent" />
19
20  </androidx.constraintlayout.widget.ConstraintLayout>
```

Java 代码中的 setPositiveButton(builder)和 setNegativeButton(builder)方法被抽出来作为单独的方法使用,由于 AlertDialog 的例子较多,把相同的代码抽出来作为工具使用很方便,这也是开发中经常用到的开发方式。这两个方法的具体代码如下所示。

```
1   private AlertDialog.Builder setPositiveButton(
2               AlertDialog.Builder builder) {
3
4           return builder.setPositiveButton("确定",
5                   new DialogInterface.OnClickListener() {
6                       @Override
7                       public void onClick(DialogInterface dialog, int which) {
8                           mBtnAlert.setText("点击了“确定”按钮");
9                       }
10                  });
11      }
12
13      private AlertDialog.Builder setNegativeButton(
14              AlertDialog.Builder builder) {
15          return builder.setNegativeButton("取消",
16                  new DialogInterface.OnClickListener() {
17                      @Override
18                      public void onClick(DialogInterface dialog, int which) {
19                          mBtnAlert.setText("点击了“取消”按钮");
20                      }
21                  });
22      }
```

在第一部分的第 2～5 行代码中,设置了对话框的标题、图标以及内容,运行程序,将看

到如图 3.5 所示的界面。

图 3.5　简单对话框

例 3.6　简单列表项对话框示例

```
1   public void simpleListAlertDialog(View view) {
2           AlertDialog.Builder builder = new AlertDialog.Builder(this)
3                   .setTitle("简单列表项对话框")
4                   .setIcon(R.drawable.warning)
5                   .setItems(items, new DialogInterface.OnClickListener() {
6                       @Override
7                       public void onClick(DialogInterface dialog, int which) {
8                           mBtnAlert.setText("您选中了《" + items[which] + "》");
9                       }
10                  });
11          setPositiveButton(builder);
12          setNegativeButton(builder)
13                  .create()
14                  .show();
15      }
```

与简单对话框一样,布局文件中同样使用了 Button 的点击事件 simpleListAlertDialog,这里就不显示具体的布局代码了,之后的几个 AlertDialog 例子与此相同。如上述代码所示,调用了 AlertDialog.Builder 中的 setItems()方法为对话框设置了多个列表项,首先定义了数组 items,这里具体的 items 数组如下所示。

```
1   Private String[] items = new String[]{"Java 语言程序设计",
2       "Android 基础", "Android 开发艺术探索", "FrameWork 学习"};
```

运行例 3.6 的程序,将看到如图 3.6 所示的界面。

例 3.7 单选列表项对话框示例

```
public void singleChoiceDialog(View view) {
        AlertDialog.Builder builder = new AlertDialog.Builder(this)
                .setTitle("单选列表项对话框")
                .setIcon(R.drawable.warning)
                //设置单选列表项,默认选中第一项(索引为 0)
                .setSingleChoiceItems(items, 0,
                        new DialogInterface.OnClickListener() {
                            @Override
                             public void onClick(DialogInterface dialog, int
                 which) {
                                mBtnAlert.setText("您选中了《" + items[which]
                    + "》");
                            }
                        });
        setPositiveButton(builder);
        setNegativeButton(builder)
                .create()
                .show();
    }
```

如上述代码所示,只要调用了 AlertDialog.Builder 的 setSingleChoiceItems()方法就可创建带单选列表项的对话框,运行程序,将看到如图 3.7 所示的界面。

图 3.6 简单列表项对话框

图 3.7 单选列表项对话框

例 3.8 多选列表项对话框示例

```
1   public void multiChoiceDialog(View view) {
2           AlertDialog.Builder builder = new AlertDialog.Builder(this)
3                   .setTitle("多选列表项对话框")
4                   .setIcon(R.drawable.icon_dialog)
5                   .setMultiChoiceItems(items,
6                       new boolean[]{true, false, false, false}, null);
7                   setPositiveButton(builder);
8                   setNegativeButton(builder)
9                           .create()
10                          .show();
11      }
```

调用 AlertDialog.Builder 的 setMultiChoiceItems()方法添加多选列表项时,需要传入一个 boolean 数组的参数,这个参数既可以在初始化时设置哪些选项可被选中,也可以动态获取列表项的选中状态。运行上面的程序,将看到如图 3.8 所示的界面。

图 3.8 多选列表项对话框

例 3.9 自定义 View 对话框

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:layout_width="match_parent"
4       android:layout_height="match_parent"
5       android:orientation="vertical">
6
```

```
7       <LinearLayout
8           android:layout_width="match_parent"
9           android:layout_height="wrap_content"
10          android:orientation="horizontal">
11
12          <TextView
13              android:layout_width="wrap_content"
14              android:layout_height="wrap_content"
15              android:paddingLeft="16dp"
16              android:text="手  机  号:"
17              android:textColor="#004"
18              android:textSize="17sp" />
19
20          <EditText
21              android:layout_width="wrap_content"
22              android:layout_height="wrap_content"
23              android:hint="请填写手机号码"
24              android:paddingLeft="8dp"
25              android:selectAllOnFocus="true" />
26      </LinearLayout>
27
28      <LinearLayout
29          android:layout_width="match_parent"
30          android:layout_height="wrap_content"
31          android:orientation="horizontal">
32
33          <TextView
34              android:layout_width="wrap_content"
35              android:layout_height="wrap_content"
36              android:paddingLeft="16dp"
37              android:text="密        码:"
38              android:textColor="#004"
39              android:textSize="16sp" />
40
41          <EditText
42              android:layout_width="wrap_content"
43              android:layout_height="wrap_content"
44              android:hint="请填写密码"
45              android:paddingLeft="8dp"
46              android:selectAllOnFocus="true" />
47      </LinearLayout>
48
49      <LinearLayout
50          android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingLeft="16dp"
            android:text="确认密码:"
            android:textColor="#004"
            android:textSize="16sp" />

        <EditText
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:hint="请填写手机号码"
            android:paddingLeft="8dp"
            android:selectAllOnFocus="true" />
    </LinearLayout>
</LinearLayout>
```

这里在 layout 文件夹下新建了名为 register_form.xml 的表单布局文件,具体内容为账号、密码以及确认密码等常规注册项,接下来在应用程序中调用 AlertDialog.Builder 的 setView()方法让对话框显示该注册界面,关键代码如下所示。

```
public void customListDialog(View view) {
    TableLayout registerForm = (TableLayout) getLayoutInflater().
    inflate(R.layout.register_form, null);
    new AlertDialog.Builder(this)
            .setTitle("自定义对话框")
            .setIcon(R.drawable.icon_dialog)
            .setView(registerForm)
            .setPositiveButton("注册",
                  new DialogInterface.OnClickListener() {
                  @Override
                  public void onClick(DialogInterface dialog, int which) {
                      //开始注册的逻辑编写
                    }
                }).setNegativeButton("取消",
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        //取消注册
                    }
                })
```

```
21                  .create()
22                  .show();
23      }
```

注意看上述代码中的粗体字代码部分，第 3 行是显式加载了 layout 文件夹中的 register_form.xml 文件，并返回该文件对应的 TableLayout 作为 View。第 7 行调用 AlertDialog.Builder 的 setView()方法显示 TableLayout。运行上面的程序，可以看到如图 3.9 所示的界面。

图 3.9　自定义 View 对话框

3.2.2　创建 DatePickerDialog 和 TimePickerDialog 对话框

DatePickerDialog 和 TimePickerDialog 的功能较为简单，用法也简单，使用步骤如下：

(1) 通过 new 关键字创建 DatePickerDialog 和 TimePickerDialog 对话框，然后调用它们自带的 show()方法即可将这两种对话框显示出来。

(2) 为 DatePickerDialog 和 TimePickerDialog 绑定监听器，这样可以保证用户通过 DatePickerDialog 和 TimePickerDialog 设置事件时触发监听器，从而通过监听器来获取用户设置的事件。

例 3.10　DatePickerDialog 对话框示例

```
1   public void dateDialog(View view) {
2           //创建 Calendar 实例
3           Calendar calendar = Calendar.getInstance();
4           //直接创建 DatePickerDialog 实例并显示
5           new DatePickerDialog(this,
6               new DatePickerDialog.OnDateSetListener() {
```

```
            @Override
            public void onDateSet(DatePicker view,
                int year, int month, int dayOfMonth) {
                TextView show = (TextView) findViewById(R.id.showDate);
                show.setText("日期选择:" + year +"-"
                    + (month + 1)+ "-" + dayOfMonth);
            }
        }, calendar.get(Calendar.YEAR),
                calendar.get(Calendar.MONTH),
                calendar.get(Calendar.DAY_OF_MONTH)).show();
    }
```

上述中粗体字代码直接创建了 DatePickerDialog 对话框。运行程序,将显示日期选择对话框,如图 3.10 所示。

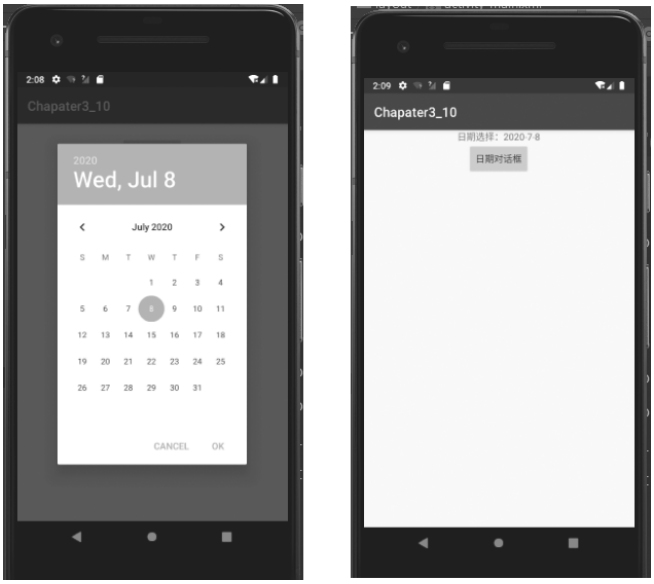

图 3.10　DatePickerDialog 对话框及选择的日期

3.2.3　创建 ProgressDialog 对话框

程序中只要创建了 ProgressDialog 实例并且调用 show( ) 方法将其显示出来,ProgressDialog 对话框就已经创建完成。在实际开发中,会经常对 ProgressDialog 对话框中的进度条进行设置,设置的方法如表 3.3 所示。

表 3.3　ProgressDialog 中的方法

方　　法	说　　明
setIndeterminater(Boolean indeterminater)	设置进度条不显示进度值
setMax(int max)	设置进度条的最大值

续表

方　　法	说　　明
setMessage(CharSequence messsge)	设置进度框里显示的消息
setProgress(int value)	设置进度条的进度值
setProgressStyle(int style)	设置进度条的风格

接下来看 ProgressDialog 示例，该程序中的界面部分和之前的一样，都是使用 Button 组件，并且在 Button 中设置点击事件。所以这里不给出界面部分的代码，直接看 Java 代码。具体如例 3.11 所示。

例 3.11　ProgressDialog 对话框示例

```
1  package com.example.chapater3_11;
2  import androidx.appcompat.app.AppCompatActivity;
3  import android.app.ProgressDialog;
4  import android.os.Bundle;
5  import android.os.Handler;
6  import android.os.Message;
7  import android.view.View;
8  import java.lang.ref.WeakReference;
9  import java.util.Timer;
10 import java.util.TimerTask;
11 public class MainActivity extends AppCompatActivity {
12     //设置 progress 的最大值
13     final static int MAX_VALUE = 100;
14     ProgressDialog progressDialog;
15     public MyHandler myHandler;
16     int status = 0;
17     //创建自定义 Handler,这种写法可避免内存泄漏
18     public class MyHandler extends Handler {
19         private WeakReference<MainActivity> myActivity;
20         public MyHandler(MainActivity activity){
21             this.myActivity = new WeakReference<>(activity);
22         }
23
24         @Override
25         public void handleMessage(Message msg) {
26             MainActivity activity = myActivity.get();
27             if (activity != null) {
28                 switch (msg.what) {
29                     case 0:
30                         //设置进度
31                         progressDialog.setProgress(status);
32                         break;
```

```
                    case 1:
                        //执行完毕之后隐藏 ProgressDialog
                        progressDialog.dismiss();
                        break;
                }
            }
            super.handleMessage(msg);
        }
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        myHandler = new MyHandler(this);
    }
    //设置的 Button 点击事件
    public void showProgress(View view) {
        status = 0;
        progressDialog = new ProgressDialog(MainActivity.this);
        //对 ProgressDialog 进行常规设置
        progressDialog.setMax(MAX_VALUE);
        progressDialog.setTitle("进度对话框");
        progressDialog.setMessage("已完成进度");
        progressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
        progressDialog.setIndeterminate(false);
        progressDialog.show();
        //从第一秒开始,每秒执行一次
        timer.schedule(task, 1000, 1000);
    }
    //使用 Timer 与 TimerTask 制造一个定时器,到固定时间向 Handler 发送消息
    Timer timer = new Timer();
    TimerTask task = new TimerTask() {
        @Override
        public void run() {
            //每次调用 TimerTask,status 就加 1
            status++;
            //任务执行中以及执行完成之后分别向 Handler 发送消息
            if (status < MAX_VALUE) {
                myHandler.sendEmptyMessage(0);
            } else {
                myHandler.sendEmptyMessage(1);
            }
        }
    };
}
```