第3章　多个用户界面的程序设计


视频讲解

3.1 页面切换与传递参数值

■ 3.1.1　绑定机制组件

Intent 是 Android 系统的一种运行时的绑定机制，在应用程序运行时连接两个不同组件。在 Android 的应用程序中，不管是页面切换、传递数据，还是调用外部程序，都可能要用到 Intent。Intent 负责对应用中某次操作的动作、动作涉及的数据、附加数据进行描述。Android 则根据 Intent 的描述，负责找到对应的组件，将 Intent 传递给调用的组件，并完成组件的调用。因此，可以将 Intent 理解为不同组件之间通信的“媒介”，其专门提供组件互相调用的相关信息。

Intent 的属性有动作（Action）、数据（Data）、分类（Category）、类型（Type）、组件（Component）以及扩展（Extra），其中最常用的是 Action 属性。

例如：

Intent.ACTION_MAIN	表示标识 Activity 为一个程序的开始。
Intent.ACTION_GET_CONTENT	表示允许用户选择图片或录音等特殊种类的数据。
Intent.ACTION_SMS_SEND	表示发送邮件的动作。
Intent.ACTION_SMS_RECEIVED	表示接收邮件的动作。
Intent.ACTION_ANSWER	表示处理呼入的电话。
Intent.ACTION_CALL_BUTTON	表示按“拨号”键。
Intent.ACTION_CALL	表示呼叫指定的电话号码。

■ 3.1.2　Activity 页面切换

Activity 跳转与传递参数值主要通过 Intent 类协助实现。在一个 Activity 页面中启动另一个 Activity 页面的运行，这是最简单的 Activity 页面切换方式。

页面切换的步骤如下：

（1）首先创建一个 Intent 对象，其构造方法如下。

```
Intent intent = new Intent(当前 Activity.this, 另一 Activity.class);
```

（2）调用 Activity 的 startActivity(intent)方法，切换到另一个 Activity 页面。

【例 3-1】 从一个 Activity 页面启动另一个 Activity 页面示例。

创建名称为 ex3_1 的新项目，包名为 com.example.ex3_1。在本项目中要建立两个页面文件及两个控制文件，第 1 个页面的界面布局文件为 activity_main.xml、控制文件为 MainActivity.java，第 2 个页面的界面布局文件为 activity_second.xml、控制文件为 secondActivity.java。此外，还要修改配置文件 AndroidManifest.xml。

（1）设计第 1 个页面。

① 进入系统自动生成的应用程序框架，在页面的界面布局中，设置一个文本标签组件和一个按钮组件，建立组件约束如图 3.1 所示。

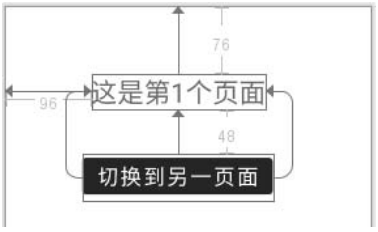

图 3.1　第 1 个页面的界面布局及组件约束

② 修改第 1 个页面的控制文件 MainActivity.java，源代码如下：

```
1   package com.example.ex3_1;
2   import androidx.appcompat.app.AppCompatActivity;
3   import android.os.Bundle;
4   import android.widget.Button;
5   import android.content.Intent;
6   import android.view.View;

7   public class MainActivity extends AppCompatActivity
8   {
9     private Button okBtn;
10    Intent intent;
11  @Override
12  public void onCreate(Bundle savedInstanceState)
13  {
14    super.onCreate(savedInstanceState);
15    setContentView(R.layout.activity_main);
16    btn = (Button)findViewById(R.id.button);
17    btn.setOnClickListener(new mClick());;
18  }
19  class btnclock implements OnClickListener ←— 定义实现监听接口的类
20  {
21    public void onClick(View v)
22    {
23       intent = new Intent(MainActivity.this, secondActivity.class);
```

```
24          startActivity(intent);   //启动新的 Activity 页面
25      }
26    }
27  }
```

（2）设计第 2 个页面。

① 右击项目管理器中的 app/java/com.example.ex3_1 选项，在弹出的快捷菜单中选择 New（新建）→Activity→Empty Activity 选项，如图 3.2 所示。

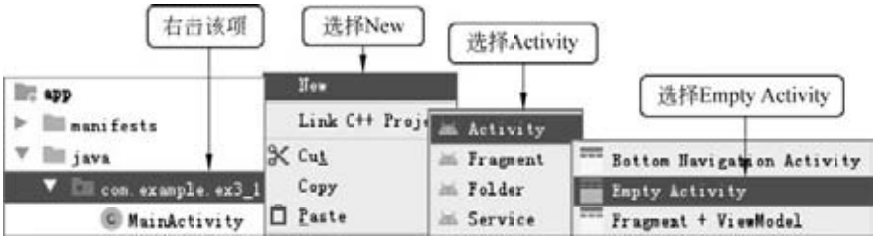

图 3.2　新建第 2 个页面

在弹出的对话框中，输入第 2 个页面的 Activity 名称 SecondActivity，其页面布局 Layout 名称为 activity_second，如图 3.3 所示。

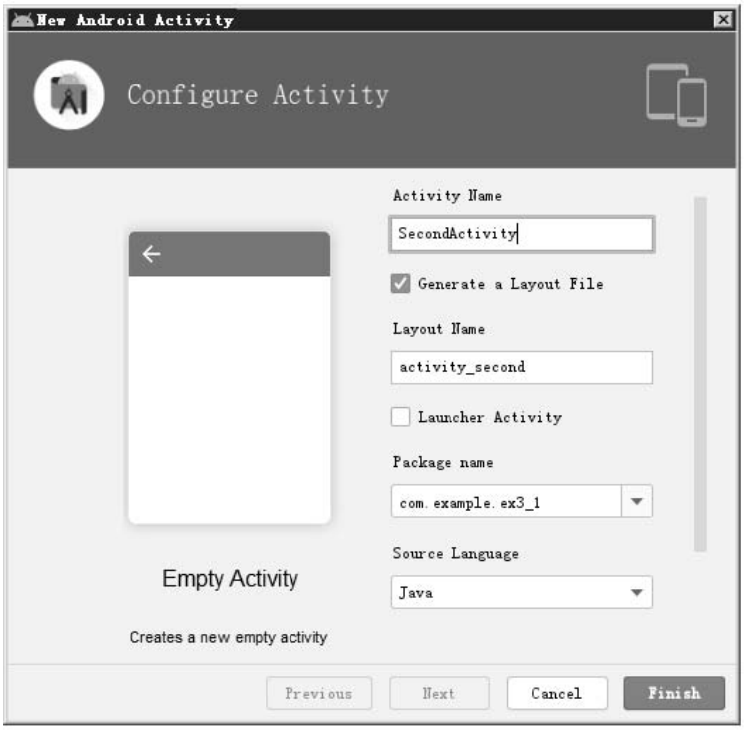

图 3.3　配置第 2 个页面

这时，系统自动生成第 2 个页面的界面布局文件 activity_second.xml 和控制文件 secondActivity.java。

② 在第 2 个页面的界面中，设置一个文本标签，将其 text 属性值设置为“这是第 2 个

页面”。自动生成的第 2 个页面的控制程序 secondActivity.java 代码不需要任何修改，其代码如下。

```
ackage com.example.ex3_1;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
ublic class SecondActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
    }
}
```

（3）配置文件 AndroidManifest.xml。

打开项目中的 AndroidManifest.xml 配置文件，可以看到增加了一行注册第 2 个 Activity 页面的代码，其程序代码如下。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.ex3_1">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Ex3_1">
        <activity android:name=".SecondActivity"></activity>
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

新增的第 2 个页面的注册语句

程序运行结果如图 3.4 所示。

图 3.4　从一个页面切换到另一页面


视频讲解

■ 3.1.3 在 Activity 页面之间传递数据

1. Bundle 类

Bundle 类是用于将字符串与某组件对象建立映射关系的组件。Bundle 组件与 Intent 配合使用，可以在不同的 Activity 之间传递数据。Bundle 类的常用方法如下。

- putString(String key, String value)：把字符串用“键-值”对形式存放到 Bundle 对象中。
- remove(String key)：移除指定 key 的值。
- getString(String key)：获取指定 key 的字符。

2. Intent 操作 Bundle 组件的方法

Intent 操作绑定组件 Bundle 的方法如下。

- getExtras()：获取 Intent 组件中绑定的 Bundle 对象。
- putExtras()：把 Bundle 对象绑定到 Intent 组件上。

3. 应用 Intent 在不同的 Activity 之间传递数据

下面说明应用 Intent 与 Bundle 配合从一个 Activity 页面传递数据到另一 Activity 页面的方法。

（1）在页面 Activity A 端。

① 创建 Intent 对象和 Bundle 对象：

```
Intent intent = new Intent();
Bundle bundle = new Bundle();
```

② 为 Intent 指定切换页面，用 Bundle 存放“键-值”对数据：

```
intent.setClass(MainActivity.this, SecondActivity.class);
bundle.putString("text", txt.getText().toString());
```

③ 将 Bundle 对象传递给 Intent：

```
intent.putExtras(bundle);
```

（2）在另一页面 Activity B 端。

① 从 Intent 中获取 Bundle 对象：

```
bunde = this.getIntent().getExtras();
```

② 从 Bundle 对象中按“键-值”对的键名获取对应数据值：

```
String str = bunde.getString("text");
```

在不同的 Activity 页面之间传递数据的过程如图 3.5 所示。

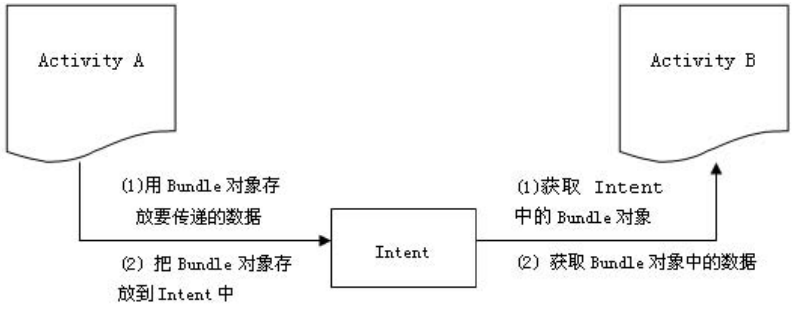

图 3.5　应用 Intent 在 Activity 页面之间传递数据

【例 3-2】 从第 1 个 Activity 页面传递数据到第 2 个 Activity 页面示例。

（1）设计第 1 个页面的界面布局 activity_main.xml。

在第 1 个页面的界面布局中，设置一个文本标签和一个按钮，建立组件约束如图 3.6 所示。

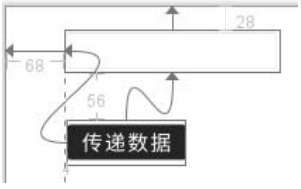

图 3.6　建立第 1 个页面的界面布局和组件约束

设置编辑框 EditText 的 background 属性值：

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

（2）设计第 1 个页面的控制程序 MainActivity.java。

```
1   package com.example.ex3_2;
2   import androidx.appcompat.app.AppCompatActivity;
3   import android.content.Intent;
4   import android.os.Bundle;
5   import android.view.View;
6   import android.widget.Button;
7   import android.widget.EditText;

8   public class MainActivity extends AppCompatActivity
9   {
10     EditText txt1;
11     Button   btn1;
12     @Override
13     protected void onCreate(Bundle savedInstanceState)
14     {
15        super.onCreate(savedInstanceState);
16        setContentView(R.layout.activity_main);
17        txt1 = (EditText)findViewById(R.id.editText);
```

```
18        btn1 = (Button)findViewById(R.id.button1);
19        btn1.setOnClickListener(new mClick());
20     }

21     class  mClick  implements View.OnClickListener{
22      @Override
23      public void onClick(View v) {
24       Intent intent = new Intent();
25       intent.setClass(MainActivity.this, SecondActivity.class);
26       Bundle bundle = new Bundle();
27       bundle.putString("text", txt1.getText().toString());
28       intent.putExtras(bundle);
29       startActivity(intent);
30      }
31    }
32  }
```

设置 Intent 对象切换的页面

Bundle 对象绑定数据

启动另一个 Activity 页面

（3）设计第 2 个页面的界面布局文件 activity_second.xml。

在项目中生成第 2 个页面，在页面中设置一个文本标签和一个按钮，其界面布局如图 3.7 所示。

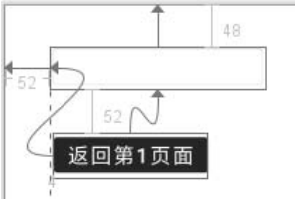

图 3.7　建立第 2 个页面的界面布局和组件约束

（4）设计第 2 个页面的控制文件 secondActivity.java，其代码如下。

```
1   package com.example.ex3_2;
2   import androidx.appcompat.app.AppCompatActivity;
3   import android.content.Intent;
4   import android.os.Bundle;
5   import android.view.View;
6   import android.widget.Button;
7   import android.widget.EditText;

8   public class MainActivity extends AppCompatActivity
9   {
10     TextView txt2;
11     Button  btn2;
12     Intent intent2;
13     Bundle bundle2;
14     @Override
15     protected void onCreate(Bundle savedInstanceState)
16     {
```

```
17          super.onCreate(savedInstanceState);
18          setContentView(R.layout.activity_main);
19          txt2 = (TextView)findViewById(R.id.textView);
20          btn2 = (Button)findViewById(R.id.button2);
21          intent2 = this.getIntent();
22          bundle2 = intent2.getExtras();
23          String str = bundle2.getString("text");   ← 获取键名为 text 的值
24          txt2.setText(str);
25          btn2.setOnClickListener(new mClick());
26      }

27      //定义返回到前一页面的监听接口事件
28      class btnclock2 implements OnClickListener
29      {
30        public void onClick(View v)
31        {
32          Intent intent3 = new Intent();
33          intent3.setClass(SecondActivity.this, MainActivity.class);
34          startActivity(intent3);   ← 返回前一页面
35        }
36      }
37  }
```

运行程序，在第 1 个页面的编辑框中输入数据，点击按钮后跳转到第 2 个页面，数据也随之传递到第 2 个页面，如图 3.8 所示。

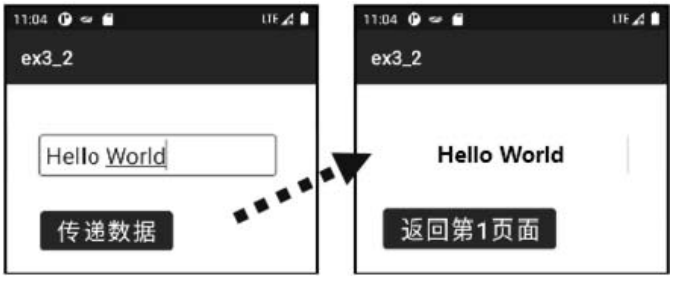

图 3.8　数据在不同 Activity 页面之间传递

3.2 菜单设计

一个菜单（Menu）由多个菜单选项组成，选择一个菜单项就可以引发一个动作事件。

在 Android 系统中，菜单可以分为 3 类：选项菜单（Option Menu）、上下文菜单（Context Menu）和子菜单（Sub Menu）。下面主要介绍选项菜单和上下文菜单的设计方法，由于子菜单的设计方法基本与选项菜单相同，这里就不赘述了。

■ 3.2.1 选项菜单

选项菜单需要通过按下设备的 Menu 键来显示。当按下设备上的 Menu 键后，在屏幕底部会弹出一个菜单，这个菜单称为选项菜单（Option Menu）。

1. Activity 中创建菜单的方法

设计选项菜单需要用到 Activity 中的 onCreateOptionMenu(Menu menu)方法，用于建立菜单并且在菜单中添加菜单项；还需要用到 Activity 中的 onOptionsItemSelected(MenuItem item)方法，用于响应菜单事件。Activity 实现选项菜单的方法如表 3-1 所示。

表 3-1 Activity 实现选项菜单的方法

方　法	说　明
onCreateOptionMenu(Menu menu)	用于初始化菜单，menu 为 Menu 对象实例
onPrepareOptionsMenu(Menu menu)	改变菜单状态，在菜单显示前调用
onOptionsMenuClosed(Menu menu)	菜单被关闭时调用
onOptionsItemSelected(MenuItem item)	菜单项被点击时调用，即菜单项的监听方法

2. 菜单 Menu

设计选项菜单需要用到 Menu、MenuItem 接口。一个 Menu 对象代表一个菜单，在 Menu 对象中可以添加菜单项 MenuItem 对象，也可以添加子菜单 Sub Menu。

菜单 Menu 使用 add(int groupId, int itemId, int order, CharSequence title) 方法添加一个菜单项，add()方法中的 4 个参数如下。

（1）groupId（组别）：如果不分组就写 Menu.NONE。

（2）itemId（id）：很重要，Android 根据这个 id 来确定不同的菜单。

（3）order（顺序）：哪个菜单项在前面由这个参数的大小决定。

（4）title（标题）：菜单项的显示文本。

3. 创建选项菜单的步骤

创建选项菜单的步骤如下：

（1）重写 Activity 的 onCreateOptionMenu(Menu menu)方法，当菜单第一次被打开时调用。

（2）调用 Menu 的 add( )方法添加菜单项（MenuItem）。

（3）重写 Activity 的 onOptionsItemSelected(MenuItem item)方法，当菜单项（MenuItem）被选择时来响应事件。

【例 3-3】 选项菜单应用示例。

设计一个选项菜单应用的示例程序，其运行结果如图 3.9 所示。

（1）设计界面布局文件 activity_main.xml。

在界面布局文件中设置一个文本标签，用于显示选择的菜单项。

图 3.9　菜单示例

（2）设计事件处理的控制程序 MainActivity.java。

控制程序 MainActivity.java 的源代码如下：

```
1   package com.example.ex3_3;
2   import androidx.appcompat.app.AppCompatActivity;
3   import android.os.Bundle;

4   import android.view.Menu;
5   import android.view.MenuItem;
6   import android.widget.TextView;

7   public class MainActivity extends AppCompatActivity
8   {
9      TextView txt;
10  @Override
11  public void onCreate(Bundle savedInstanceState)
12  {
13     super.onCreate(savedInstanceState);
14     setContentView(R.layout.activity_main);
15     txt = (TextView)findViewById(R.id.TextView1);
16  }

17  public boolean onCreateOptionsMenu(Menu menu)
18  {
19     // 调用父类方法来加入系统菜单
20     super.onCreateOptionsMenu(menu);
21     // 添加菜单项
22     menu.add(
23                 1,          //组号
24                 1,          //唯一的 id
25                 1,          //序号
26                 "菜单项 1"); //菜单项标题
27        menu.add( 1, 2, 2,  "菜单项 2");
28        menu.add( 1, 3, 3,  "菜单项 3");
29        menu.add( 1, 4, 4,  "菜单项 4");
```

添加菜单项的 4 个参数

```
30        return true;
31    }

32    public boolean onOptionsItemSelected(MenuItem item)
33    {
34      String title = "选择了" + item.getTitle().toString();
35      switch (item.getItemId())
36       { //响应每个菜单项(通过菜单项的 id)
37         case 1:
38             txt.setText(title); ←文本标签显示菜单项的标题
39             break;
40         case 2:
41             txt.setText(title); ←文本标签显示菜单项的标题
42             break;
43         case 3:
44             txt.setText(title); ←文本标签显示菜单项的标题
45             break;
46         case 4:
47             txt.setText(title); ←文本标签显示菜单项的标题
48             break;
49          default:
50              //没有处理的事件交给父类来处理
51             return super.onOptionsItemSelected(item);
52        }
53       return true;
54    }
55 }
```

■ 3.2.2 上下文菜单

Android 系统中的上下文菜单类似于计算机上的右键菜单。在为一个视图注册了上下文菜单之后，长按（两秒左右）这个视图对象就会弹出一个浮动菜单，即上下文菜单。任何视图都可以注册上下文菜单，最常见的是用于列表视图 ListView 的 item。

创建一个上下文菜单的步骤如下：

（1）重写 Activity 的 onCreateContenxtMenu()方法，调用 Menu 的 add 方法添加菜单项（MenuItem）。

（2）重写 Activity 的 onContextItemSelected()方法，响应上下文菜单菜单项的单击事件。

（3）调用 Activity 的 registerForContextMenu()方法，为视图注册上下文菜单。

【例 3-4】 上下文菜单应用示例。

设计一个上下文菜单应用的示例程序，其运行结果如图 3.10 所示。

（1）设计界面布局文件 activity_main.xml。

在界面布局文件中设置 3 个文本标签，用于显示选择项，建立其约束如图 3.11 所示。

图 3.10　上下文菜单应用示例

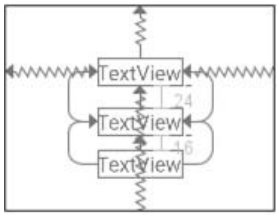

图 3.11　建立 3 个文本标签的约束

（2）设计事件处理的控制程序 MainActivity.java。

控制程序 MainActivity.java 的源代码如下：

```
1   package com.example.ex3_4;
2   import androidx.appcompat.app.AppCompatActivity;
3   import android.os.Bundle;
4   import android.view.Menu;
5   import android.view.MenuItem;
6   import android.view.ContextMenu;
7   import android.widget.TextView;
8   import android.view.View;
9   public class MainActivity extends AppCompatActivity
10  {
11      TextView txt1, txt2, txt3;
12      private static final int item1 = Menu.FIRST;
13      private static final int item2 = Menu.FIRST+1;
14      private static final int item3 = Menu.FIRST+2;
15      String str[] = {"黑暗天使", "战神觉醒", "渡劫修仙"};
16  @Override
17  public void onCreate(Bundle savedInstanceState)
18  {
19     super.onCreate(savedInstanceState);
20     setContentView(R.layout.activity_main);
21     txt1 = (TextView)findViewById(R.id.TextView1);
22     txt2 = (TextView)findViewById(R.id.textView2);
23     txt3 = (TextView)findViewById(R.id.textView3);
24     txt1.setText(str[0].toString());
25     txt2.setText(str[1].toString());
26     txt3.setText(str[2].toString());
27     registerForContextMenu(txt1);
28     registerForContextMenu(txt2);
29     registerForContextMenu(txt3);
```

初始化组件

```
30  }

31      //上下文菜单，本例会通过长按条目激活上下文菜单
32      @Override
33      public void onCreateContextMenu(ContextMenu menu, View view,
34      ContextMenuInfo menuInfo)
35      {
36           menu.setHeaderTitle("游戏人物简介");
37           //添加菜单项
38           menu.add(0, item1, 0, "武功");
39           menu.add(0, item2, 0, "战斗力");
40           menu.add(0, item3, 0, "经典语录");
41      }
42      //菜单单击响应
43      @Override
44      public boolean onContextItemSelected(MenuItem item)
45      {
46        //获取当前被选择的菜单项的信息
47        switch(item.getItemId())
48        {
49          case item1:
50             //在这里添加处理代码
51             break;
52         case item2:
53             //在这里添加处理代码
54             break;
55         case item3:
56             //在这里添加处理代码
57             break;
58        }
59        return true;
60      }
61  }
```

选项的具体功能没有实现

3.3 对话框

对话框是一个有边框、有标题栏的独立存在的容器，在应用程序中经常使用对话框组件来进行人机交互。Android 系统提供了 4 种常用对话框。

- AlertDialog：消息对话框。
- ProgressDialog：进度条对话框。
- DatePickerDialog：日期选择对话框。
- TimePickerDialog：时间选择对话框。

下面逐一介绍这些对话框的使用方法。

■ 3.3.1　消息对话框

AlertDialog 对话框是应用程序设计中最常用的对话框之一。AlertDialog 对话框的内容很丰富，使用 AlertDialog 类可以创建普通对话框、带列表的对话框以及带单选按钮和多选按钮的对话框。AlertDialog 类的常用方法如表 3-2 所示。

表 3-2　AlertDialog 类的常用方法

方　法	说　明
AlertDialog.Builder(Context)	对话框 Builder 对象的构造方法
create();	创建 AlertDialog 对象
setTitle();	设置对话框标题
setIcon();	设置对话框图标
setMessage();	设置对话框的提示信息
setItems();	设置对话框要显示的一个 list
setPositiveButton();	在对话框中添加 yes 按钮
setNegativeButton();	在对话框中添加 no 按钮
show();	显示对话框
dismiss();	关闭对话框

创建 AlertDialog 对象需要使用 AlertDialog 的内部类 Builder。设计 AlertDialog 对话框的步骤如下。

（1）用 AlertDialog.Builder 类创建对话框 Builder 对象。

```
Builder dialog=new AlertDialog.Builder(Context);
```

（2）设置对话框的标题、图标、提示信息内容、按钮等。

```
dialog.setTitle("普通对话框");
dialog.setIcon(R.drawable.icon1);
dialog.setMessage("一个简单的提示对话框");
dialog.setPositiveButton("确定", new okClick());
```

（3）创建并显示 AlertDialog 对话框对象。

```
dialog.create();
dialog.show();
```

如果在对话框内部设置了按钮，还需要为其设置事件监听 OnClickListener。

【例 3-5】 消息对话框应用示例。

在本例中设计了两种形式的对话框程序，一种是发出提示信息的普通对话框，另一种是用户登录对话框。

在用户登录对话框中，设计了用户登录的布局文件 long.xml，供用户输入相关验证信息。

在创建的应用程序框架中，将事先准备的图像文件 icon1.jpg、icon2.jpg 复制到 res\drawable 目录下，用作对话框的图标。

程序的运行结果如图 3.12 所示。

(a) 普通对话框　　（b）用户登录对话框

图 3.12　AlertDialog 对话框

（1）设计界面布局文件 activity_main.xml。

在界面中设置两个按钮，建立约束如图 3.13 所示。

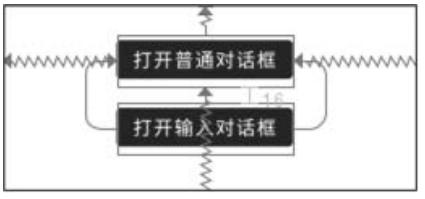

图 3.13　AlertDialog 对话框的约束

（2）设计登录对话框的界面布局文件 login.xml。

新建一个 xml 文件，命名为 login.xml，其代码如下：

```
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     <TextView
7         android:id="@+id/user"
8         android:layout_width="wrap_content"
9         android:layout_height="wrap_content"
10        android:text="用户名"
11        android:textSize="18sp"/>
```

```
12    <EditText
13        android:id="@+id/userEdit"
14        android:layout_width="match_parent"
15        android:layout_height="wrap_content"
16        android:textSize="18sp"/>
17    <TextView
18        android:id="@+id/textView"
19        android:layout_width="match_parent"
20        android:layout_height="wrap_content"
21        android:text="密码"
22        android:textSize="18sp"/>
23    <EditText
24        android:id="@+id/paswdEdit"
25        android:layout_width="match_parent"
26        android:layout_height="wrap_content"
27        android:textSize="18sp"/>
28 </LinearLayout>
```

（3）设计控制文件 MainActivity.java。

```
1   package com.example.ex3_5;
2   import androidx.appcompat.app.AppCompatActivity;
3   import android.os.Bundle;
4   import android.app.AlertDialog;
5   import android.content.DialogInterface;
6   import android.view.View;
7   import android.widget.Button;
8   import android. widget.EditText;
9   import android.widget.LinearLayout;
10  import android.widget.Toast;

11  public class MainActivity extends AppCompatActivity
12  {
13      Button btn1, btn2;
14      LinearLayout login;
15      @Override
16      public void onCreate(Bundle savedInstanceState)
17      {
18          super.onCreate(savedInstanceState);
19          setContentView(R.layout.activity_main);
20          btn1 = (Button)findViewById(R.id.button1);
21          btn2 = (Button)findViewById(R.id.button2);
22          btn1.setOnClickListener(new mClick());
23          btn2.setOnClickListener(new mClick());
24      }
25      class mClick implements OnClickListener
26      {
```

```
27       Builder dialog=new AlertDialog.Builder(MainActivity.this);
28       @Override
29        public void onClick(View v)
30       {
31          if(v == btn1)
32          {
33           //设置对话框的标题
34           dialog.setTitle("警告");
35           //设置对话框的图标
36           dialog.setIcon(R.drawable.icon1);
37           //设置对话框显示的内容
38           dialog.setMessage("本项操作可能导致信息泄露！");
39           //设置对话框的“确定”按钮
40           dialog.setPositiveButton("确定", new okClick());
41           //创建对象框
42           dialog.create();
43           //显示对象框
44           dialog.show();
45         }
46         else  if(v == btn2)
47         {
48           login = (LinearLayout)getLayoutInflater()
49                    .inflate(R.layout.login, null);
50           dialog.setTitle("用户登录").setMessage("请输入用户名和密码")
51                    .setView(login);
52           dialog.setPositiveButton("确定", new loginClick());
53           dialog.setNegativeButton("退出", new exitClick());
54           dialog.setIcon(R.drawable.icon2);
55           dialog.create();
56           dialog.show();
57          }
58       }
59    }
60    /*  普通对话框的“确定”按钮事件 */
61    class okClick implements DialogInterface.OnClickListener
62    {
63      @Override
64      public void onClick(DialogInterface dialog, int which)
65      {
66         dialog.cancel();
67       }
68     }
69    /*  输入对话框的“确定”按钮事件   */
70    class loginClick implements  DialogInterface.OnClickListener
71    {
72      EditText txt;
73      @Override
```

设置对话框

从新建布局中关联组件

关闭对话框

```
74      public void onClick(DialogInterface dialog, int which)
75      {
76       txt = (EditText)login.findViewById(R.id.paswdEdit);
77       //取出输入编辑框的值与密码“admin”比较
78       if((txt.getText().toString()).equals("admin"))
79          Toast.makeText(getApplicationContext(),
80              "登录成功", Toast.LENGTH_SHORT).show();
81       else
82          Toast.makeText(getApplicationContext(),
83              "密码错误", Toast.LENGTH_SHORT).show();
84       dialog.dismiss();
85      }
86    }
87    /*  输入对话框的“退出”按钮事件   */
88    class exitClick implements DialogInterface.OnClickListener
89    {
90      @Override
91      public void onClick(DialogInterface dialog, int which)
92      {
93         MainActivity.this.finish();
94      }
95    }
96  }
```

关联布局文件中的组件

密码为 admin 时，显示“登录成功”

关闭对话框

点击“退出”按钮，退出 MainActivity 程序

对于程序的第 48、49 行：

```
login = (LinearLayout)getLayoutInflater()
              .inflate(R.layout.login, null);
```

这里 inflate()是将组件从一个 XML 中定义的布局找出来。

在一个 Activity 中如果直接用 findViewById()，对应的是 setConentView()中的那个 layout 中的组件（程序第 19 行中的 R.layout.activity_main）。如果 Activity 中用到其他 layout 布局，比如对话框上的 layout，还要设置对话框上的 layout 中的组件（像图片 ImageView、文字 TextView）上的内容，这就必须用 inflate()先将对话框上的 layout 找出来，然后再用这个 layout 对象找到它上面的组件。

■ 3.3.2　其他几种常用对话框

1. 进度条对话框

Android 系统有一个 ProgressDialog 类，它继承于 AlertDialog 类，综合了进度条与对话框的特点，使用起来非常简单。ProgressDialog 类的继承关系如图 3.14 所示。

图 3.14　ProgressDialog 类继承于 AlertDialog 类

ProgressDialog 类的常用方法如表 3-3 所示。

表 3-3　ProgressDialog 类的常用方法

方　法	说　明
getMax()	获取对话框进度的最大值
getProgress()	获取对话框当前的进度值
onStart()	开始调用对话框
setMax(int max)	设置对话框进度的最大值
setMessage(CharSequence message)	设置对话框的文本内容
setProgress(int value)	设置对话框当前的进度
show(Context context, CharSequence title, CharSequence message)	设置对话框的显示内容和方式
ProgressDialog(Context context)	对话框的构造方法

2. 日期对话框和时间对话框

日期选择类 DatePickerDialog 和时间选择类 TimePickerDialog 都继承于 AlertDialog 类，一般用于日期和时间的设定，它们的常用方法如表 3-4 所示。

表 3-4　日期和时间选择对话框的常用方法

方　法	说　明
updateDate(int year, int monthOfYear, int dayOfMonth)	设置 DatePickerDialog 对象的当前日期
onDateChanged(DatePicker view, int year, int month, int day)	修改 DatePickerDialog 对象的日期
updateTime(int hourOfDay, int minutOfHour)	设置 TimePickerDialog 对象的时间
onTimeChanged(TimePicker view, int hourOfDay, int minute)	修改 TimePickerDialog 对象的时间

【例 3-6】 进度对话框、日期对话框和时间对话框示例。

（1）设计用户界面程序 activity_main.xml。

在界面设计中，设置 3 个按钮，分别用于打开进度对话框、日期对话框和时间对话框。

（2）设计控制程序 MainActivity.java。

```
1   package com.example.ex3_6;
2   import androidx.appcompat.app.AppCompatActivity;
3   import android.os.Bundle;

4   package com.example.ex3_6;
5   import android.app.Activity;
6   import android.app.DatePickerDialog;
7   import android.app.ProgressDialog;
8   import android.app.TimePickerDialog;
9   import android.app.DatePickerDialog.OnDateSetListener;
10  import android.app.TimePickerDialog.OnTimeSetListener;
11  import android.view.View;
12  import android.view.View.OnClickListener;
13  import android.widget.Button;
14  import android.widget.DatePicker;
15  import android.widget.TimePicker;
```

```
16  public class MainActivity extends Activity
17  {
18     Button btn1,btn2,btn3;
19     @Override
20     public void onCreate(Bundle savedInstanceState)
21     {
22        super.onCreate(savedInstanceState);
23        setContentView(R.layout.activity_main);
24        btn1=(Button)findViewById(R.id.button1);
25        btn2=(Button)findViewById(R.id.button2);
26        btn3=(Button)findViewById(R.id.button3);
27        btn1.setOnClickListener(new mClick());
28        btn2.setOnClickListener(new mClick());
29        btn3.setOnClickListener(new mClick());
30     }
31     class mClick implements OnClickListener
32     {
33        int m_year = 2012;
34         int m_month = 1;
35         int m_day = 1;
36         int m_hour = 12,   m_minute = 1;
37         @Override
38         public void onClick(View v)
39         {
40           if(v == btn1)
41            {
42              ProgressDialog  d=new ProgressDialog (MainActivity.this);
43              d.setTitle("进度对话框");
44              d.setIndeterminate(true);
45              d.setMessage("程序正在 Loading...");
46              d.setCancelable(true);
47              d.setMax(10);
48              d.show();
49            }
50           else  if(v == btn2)
51           {
52             //设置日期监听器
53             DatePickerDialog.OnDateSetListener dateListener =
54                     new DatePickerDialog.OnDateSetListener()
55             {
56               @Override
57               public void onDateSet(DatePicker view, int year,
58                     int month, int dayOfMonth)
59               {
60                  m_year = year;
61                  m_month = month;
```