第5章 深入理解 Activity 与 Fragment

本章学习目标

- 掌握 Activity 的创建与使用。
- 熟悉 Activity 的生命周期。
- 掌握 Fragment 的创建与使用。
- 熟悉 Fragment 的生命周期。

Android 应用中，Activity、Service、BroadcastReceiver 和 ContentProvider 这四大基本组件是 Android 开发必学内容，而 Activity 是其中最重要的一项。它负责与用户交互，并向用户呈现应用的状态，通常一个 Android 应用由 *N* 个 Activity 组成。Fragment 代表了 Activity 的子模块，与 Activity 一样，Fragment 也有自己的生命周期。通过本章节的学习，熟悉 Activity 与 Fragment 的生命周期，以及掌握它们的建立与使用方法。

5.1 创建、配置和使用 Activity

5.1.1 Activity 介绍

学习一个新知识点时，总要追根溯源才能彻底掌握。学习 Activity 也不例外，Activity 直接或间接继承了 Context、ContextWrapper、ContextThemeWrapper 等基类，如图 5.1 所示。

在使用 Activity 时，需要用户继承 Activity 基类。在不同的应用场景下，可以选择继承 Activity 的子类。例如界面中只包括列表，则可以继承 ListActivity；若界面需要实现标签页效果，则要继承 TabActivity。

当一个 Activity 类被定义出来之后，这个 Activity 类何时被实例化、它所包含的方法何时被调用都是由 Android 系统决定的。用户只负责实现相应的方法创建出需要的 Activity 即可。

创建一个 Activity 需要实现一个或多个方法，其中最基本的方法是 onCreate(Bundle status)，它将会在 Activity 被创建时回调，然后通过 setContentView(View view)方法显示要展示的布局文件。这个知识点在第 1 章介绍 HelloWorld 项目时就提到过。

接下来看一个 LauncherActivity 的例子。从图 5.1 可以看到 LauncherActivity 继承自 ListActivity，所以它本质也是一个开发列表界面的 Activity，但不同的是，它的每个列表项都对应一个 Intent，因此当用户点击不同的列表项时，应用程序会自动启动对应的 Activity。

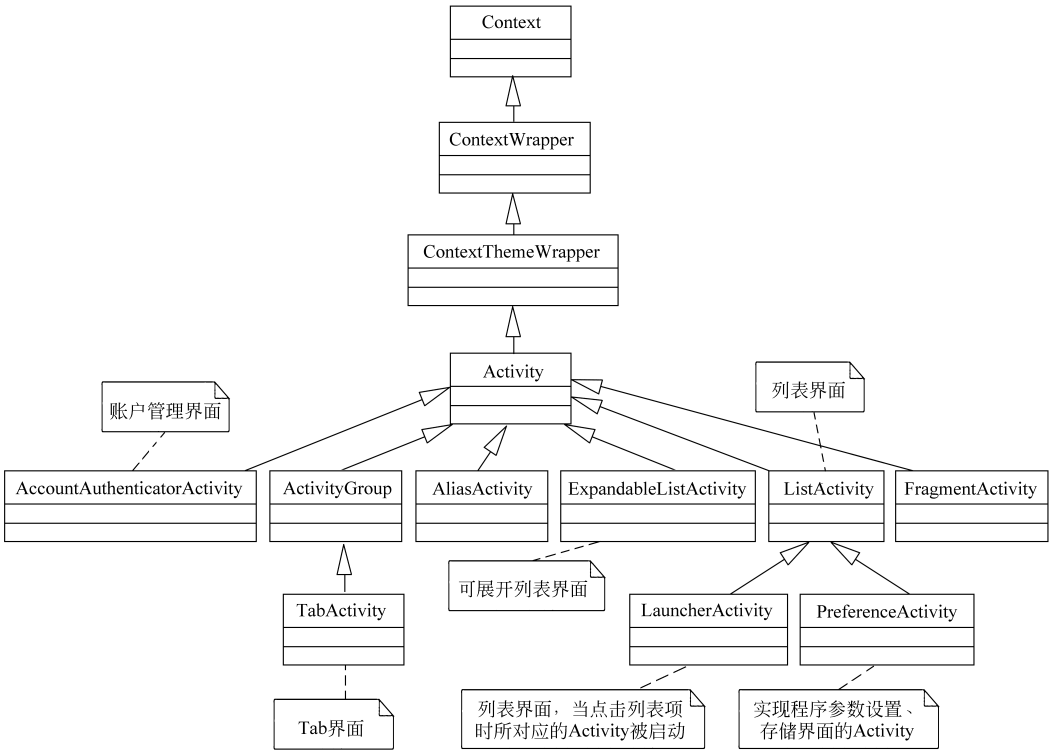

图 5.1　Activity 类

具体如例 5.1 所示。

例 5.1　LauncherActivity 用法示例

```
package com.example.chapater5_1;
import android.app.LauncherActivity;
import android.content.Intent;
import android.os.Bundle;
import android.widget.ArrayAdapter;
public class MainActivity extends LauncherActivity {
    //定义两个 Activity 的名称
    String[] names = {"FirstActivity", "SecondActivity"};
    //定义两个 Activity
    Class<?>[] clazzs = {FirstActivity.class, SecondActivity.class};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_1, names);
        //设置列表所需的 Adapter
        setListAdapter(adapter);
    }
```

```
19      //根据列表项返回指定 Activity 对应的 Intent
20      @Override
21      protected Intent intentForPosition(int position) {
22          return new Intent(MainActivity.this, clazzs[position]);
23      }
24  }
```

需要注意的是，上面代码中 onCreate(Bundle savedInstanceState)方法中没有使用 setContentView(View view)加载 view，而是使用 ArrayAdapter 加载了一个列表。这也是 ListActivity 的不同之处。intentForPosition(int position)方法根据用户点击的列表项启动相应的 Activity。布局文件 simple_layout_item.xml 是一个根标签为 TextView 的布局，用于加载 names 列表。

至此 LauncherActivity 的示例已经完成，但是这只是创建完成了 MainActivity，只有在 AndroidManifest.xml 文件中配置了 MainActivity 才可以使用。读者可能已经发现，之前的很多例子中也是在 MainActivity 中操作完成的，同样可以在模拟器上运行，这里为什么就不行呢？这是因为 Android Studio 自动在 AndroidMainfest.xml 文件中配置了 MainActivity，所以才能直接使用。

5.1.2 配置 Activity

5.1.1 节提到 Activity 必须在 AndroidManifest.xml 清单文件中配置才可以使用，而在 Android Studio 中是自动配置完成，但是有时自动配置完成的属性并不能满足需求。来看配置 Activity 时常用的属性，如表 5.1 所示。

表 5.1 配置 Activity 属性

属　性	说　明
name	指定 Activity 的类名
icon	指定 Activity 对应的图标
label	指定 Activity 的标签
exported	指定该 Activity 是否允许被其他应用调用
launchMode	指定 Activity 的启动模式

除了上述几个属性之外，Activity 中还可以设置一个或多个＜intent-filter…/＞元素，该元素用于指定该 Activity 相应的 Intent。

下面来看例 5.1 中清单文件配置的 3 个 Activity。

```
<activity android:name=".SecondActivity"></activity>
<activity android:name=".FirstActivity" />
<activity android:name=".MainActivity">
     <intent-filter>
       <action android:name="android.intent.action.MAIN" />
```

```
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
```


上述配置代码配置了 3 个 Activity,其中第一个 Activity 配置了<intent-filter…/>元素,指定了该 Activity 作为程序的入口。运行例 5.1 程序,将会看到如图 5.2 所示的界面。

点击第 1 个列表项 FirstActivity,会出现如图 5.3 所示的界面。

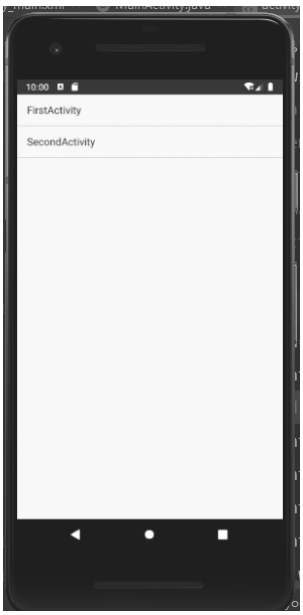

图 5.2 LauncherActivity 运行结果

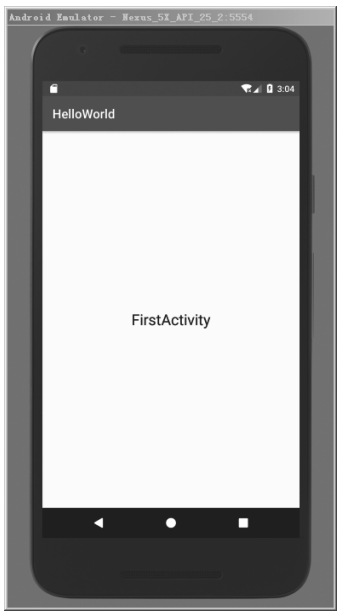

图 5.3 点击 LauncherActivity 第一个列表项后

点击第 2 个列表项出现的结果与第一个一样,故这里不做展示。

5.1.3 Activity 的启动与关闭

在一个 Android 应用程序中通常会有多个 Activity,每个 Activity 都是可以被其他 Activity 启动的,但程序只有一个 Activity 作为入口,即程序启动时只会启动作为入口的 Activity,其他 Activity 会被已经启动的其他 Activity 启动。

启动 Activity 的方式有以下两种。

- startActivity(Intent intent):启动其他 Activity。
- startActivityForResult(Intent intent, int requestCode):以指定的请求码(requestCode)启动新 Activity,并且原来的 Activity 会获取新启动的 Activity 返回的结果(需重写 onActivityResult()方法)。

启动 Activity 有两种方式,关闭 Activity 也有两种方式。

- finish():关闭当前 Activity。
- finishActivity(int requestCode):结束以 startActivityForResult(Intent intent, int requestCode)方法启动的 Activity。

接下来示范 Activity 的启动，以及实现两个 Activity 的切换。具体如例 5.2 所示。

例 5.2 Activity 的显式与隐式启动，两个 Activity 之间的切换

```
1  package com.example.chapater5_2;
2
3  import android.content.Intent;
4  import android.os.Bundle;
5  import android.view.View;
6  import android.widget.Button;
7
8  import androidx.appcompat.app.AppCompatActivity;
9
10 public class MainActivity extends AppCompatActivity {
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_main);
16         Button button1 = findViewById(R.id.btn1);
17         Button button2 = findViewById(R.id.btn2);
18         Button button3 = findViewById(R.id.btn3);
19
20         //显式启动 Activity
21         button1.setOnClickListener(new View.OnClickListener() {
22             @Override
23             public void onClick(View v) {
24                 Intent intent = new Intent(MainActivity.this,
25                         SecondActivity.class);
26                 startActivity(intent);
27             }
28         });
29
30         //调用 setClassName()方法显式启动 Activity
31         button2.setOnClickListener(new View.OnClickListener() {
32             @Override
33             public void onClick(View v) {
34                 Intent intent = new Intent();
35                 //第一个参数是包名，第二个参数是类的全路径名
36                 intent.setClassName("com.example.chapater5_2",
37                         "com.example.chapater5_2.SecondActivity");
38                 startActivity(intent);
39             }
40         });
41
```

```
        //隐式启动
        button3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent();
                intent.setAction("com.example.chapater5_2.SecondActivity");
                startActivity(intent);
            }
        });
    }
}

```

上述代码对应的布局文件中只有 3 个按钮,不做展示。这里示范了两种启动 Activity 的形式,要注意的是显式启动时用 setClassName(String packageName, String className)方法,第 1 个参数是包名,第 2 个参数是类的全路径名。隐式启动的方式之前没有例子涉及,它需要在清单文件中对应的 Activity 中设置 action 标签,并且与代码中的 setAction()中设置的内容一样。下面来看清单文件中 SecondAtivity 中的配置。

```
<activity android:name=".SecondActivity" >
    <intent-filter>
        <action android:name="com.example.helloworld.SecondActivity"/>
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

清单文件中的代码与隐式启动的 setAction()方法中一定要一样。其实隐式启动时 Android 系统会根据清单文件中设置的 action、category、uri 找到最合适的组件,只不过本例中设置 category 为“android.intent.category.DEFAULT”是一种默认的类别,在调用 startActivity()时会自动将这个 category 添加到 Intent。

接下来看 SecondActivity 中的代码。

```
package com.example.chapater5_2;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import androidx.appcompat.app.AppCompatActivity;

public class SecondActivity extends AppCompatActivity {

    @Override
```

```
12      protected void onCreate(Bundle savedInstanceState) {
13          super.onCreate(savedInstanceState);
14          setContentView(R.layout.second_layout);
15          Button button1 = findViewById(R.id.btn1);
16          Button button2 = findViewById(R.id.btn2);
17          button1.setOnClickListener(new View.OnClickListener() {
18              @Override
19              public void onClick(View v) {
20                  Intent intent = new Intent(SecondActivity.this,
21                          MainActivity.class);
22                  startActivity(intent);
23              }
24          });
25
26          button2.setOnClickListener(new View.OnClickListener() {
27              @Override
28              public void onClick(View v) {
29                  Intent intent = new Intent(SecondActivity.this,
30                          MainActivity.class);
31                  startActivity(intent);
32                  finish();
33              }
34          });
35      }
36  }
```

上述代码中有两个监听器，只是一个有 finish()方法而另一个没有。如果有 finish()则表示点击按钮后会关闭自己。

5.1.4 使用 Bundle 在 Activity 之间交换数据

在实际开发中，一个 Activity 启动另一个 Activity 时经常需要传输数据过去。在 Activity 之间交换数据很简单，使用 Intent 即可。在启动新的 Activity 时，利用 Intent 提供的多种方法将数据传递过去。常用的方法如表 5.2 所示。

表 5.2 传递数据时常用的方法

方　法	作　用
putExtras(Bundle data)	向 Intent 中放入需要携带的数据包
getExtras()	取出 Intent 所携带的数据包
putExtra(String name, Xxx value)	向 Intent 中放入 key-value 形式的数据
getXxxExtra(String name)	按 key 取出 Intent 中指定类型的数据
putXxx(String key, Xxx data)	向 Bundle 中放入各种类型的数据

续表

方　　法	作　　用
getXxx(String key)	从 Bundle 中取出各种类型的数据
putSerializable(String key, Serializable data)	向 Bundle 中放入一个可序列化的对象
getSerializable(String key, Serializable data)	从 Bundle 中取出一个可序列化的对象

Intent 主要通过 Bundle 对象来携带数据,使用到的方法都如表 5.2 所示。下面通过一个示例示范两个 Activity 通过 Bundle 交换数据,假设千锋需要学生的基本信息,学生在填写资料之后提交到下一个页面,具体代码如例 5.3 所示。

例 5.3　信息填写页面的布局文件

```
<?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">

    <LinearLayout
        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:text="昵称:"/>
        <EditText
            android:id="@+id/nickName"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"/>
    </LinearLayout>
    <LinearLayout
        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:text="年龄:"/>
        <EditText
            android:id="@+id/age"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
    </LinearLayout>

```

```
33      <LinearLayout
34          android:layout_width="match_parent"
35          android:layout_height="wrap_content"
36          android:orientation="horizontal">
37          <TextView
38              android:layout_width="wrap_content"
39              android:layout_height="wrap_content"
40              android:text="性别:"/>
41          <RadioGroup
42              android:layout_width="match_parent"
43              android:layout_height="wrap_content">
44              <RadioButton
45                  android:id="@+id/male"
46                  android:layout_width="wrap_content"
47                  android:layout_height="wrap_content"
48                  android:text="男"/>
49              <RadioButton
50                  android:id="@+id/female"
51                  android:layout_width="wrap_content"
52                  android:layout_height="wrap_content"
53                  android:text="女"/>
54          </RadioGroup>
55      </LinearLayout>
56      <LinearLayout
57          android:layout_width="match_parent"
58          android:layout_height="wrap_content"
59          android:orientation="horizontal">
60          <TextView
61              android:layout_width="wrap_content"
62              android:layout_height="wrap_content"
63              android:text="学历:"/>
64          <EditText
65              android:id="@+id/qualifications"
66              android:layout_width="match_parent"
67              android:layout_height="wrap_content"/>
68      </LinearLayout>
69      <LinearLayout
70          android:layout_width="match_parent"
71          android:layout_height="wrap_content"
72          android:orientation="horizontal">
73          <TextView
74              android:layout_width="wrap_content"
75              android:layout_height="wrap_content"
76              android:text="电话:"/>
```

```
        <EditText
            android:id="@+id/phone"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"/>
    </LinearLayout>
    <!--确认-->
    <Button
        android:id="@+id/submit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="提交"/>
</LinearLayout>
```

上述布局文件采用 LinearLayout 方式,对应的 Java 代码如下所示。

```
package com.example.chapater5_3;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    private EditText nickName, age, qualifications, phone;
    private RadioButton male, female;
    private Button submit;

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

        setTitle("学生信息调查");
        submit = findViewById(R.id.submit);
        submit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                nickName = findViewById(R.id.nickName);
                age = findViewById(R.id.age);
```