



项目5

理解App的活动

5.1 项目目标：理解 App 的活动机制与状态

本项目的内容是创建和设计 App 必备的基础理论知识。任何一个 App 都像人的生命一样,有开始、运行、暂停和消亡,在这个过程中,可以用各种方法对程序进行控制。本项目设定的目标为掌握 Activity 的状态、生命周期及相关属性,熟练编写代码实现创建 Activity,设置 Activity 以及启动、关闭 Activity 等操作。

5.2 项目准备

Android 具有四大组件,分别为 Activity、Service(服务)、Content Provider(内容提供)、BroadcastReceiver(广播接收器)。Activity 是 Android 组件中最基本也是最常用的四大组件之一,其本质是一个应用程序组件,专门为人机交互提供可视化屏幕、交互接口,并完成交互任务。Activity 可以通过 setContentView(View)来显示指定控件。在一个 Android 应用中,一个 Activity 通常就是一个单独的屏幕,它上面可以显示一些控件,也可以监听并对

用户的事件做出响应。Activity 之间通过 Intent 进行通信。

5.2.1 介绍 Activity 的状态

在 Android 中,Activity 拥有以下 4 种基本状态。

(1) Active/Running(运行状态): 一个新 Activity 启动入栈后,它在屏幕最前端,处于栈的最顶端,此时它处于可见并可与用户交互的激活状态。

(2) Paused(暂停状态): 当 Activity 被另一个透明或者 Dialog 样式的 Activity 覆盖时,处于 Paused 状态。此时它依然与窗口管理器保持连接,系统继续维护其内部状态,所以它仍然可见,但它已经失去了焦点,故不可与用户交互。

(3) Stopped(停止状态): 当 Activity 被另外一个 Activity 覆盖、失去焦点并不可见时,处于 Stopped 状态。

(4) Killed(死亡状态): Activity 被系统杀死回收或者没有被启动时,处于 Killed 状态。

当一个 Activity 实例被创建、销毁或者启动另外一个 Activity 时,它在这 4 种状态之间进行转换,这种转换的发生依赖于用户程序的动作。图 5-1 说明了 Activity 在不同状态之间转换的时机和条件。

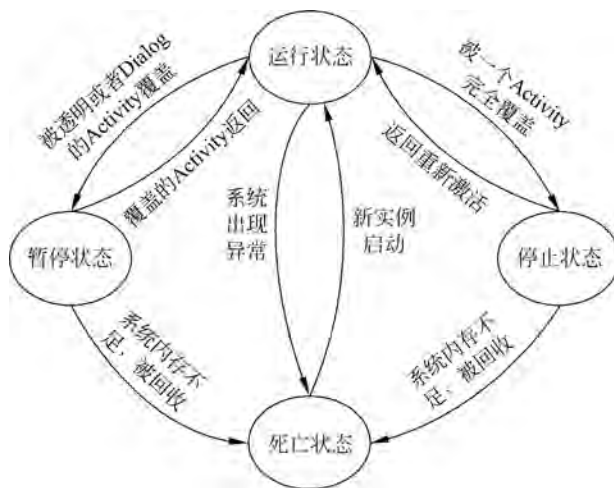


图 5-1 Activity 的 4 种状态的转换

值得注意的是,开发 Android 的程序员开启一个 Activity,却不能够通过 Activity.finish()方法结束它,仅能使其被回收,即从 Active/Running 状态转到 Paused 状态。

5.2.2 介绍 Activity 的生命周期

根据官方文档对 Activity 生命周期的说明(如图 5-2 所示),可以对任一款 App 的生命周期活动进行描述。首先,在第一次打开某款 App 时,当前的 FirstActivity 即被执行,随后 Android 系统依次调用 onCreate() 方法、onStart() 方法、onResume() 方法,使得 FirstActivity 完全被启动,并显示到前台。如果 Intent 同时启动了 SecondActivity,则会通过 onPause() 方法暂停之前的 FirstActivity。如果又选择了 Back 键,则会通过 onResume() 方法重新返回到 FirstActivity,而 SecondActivity 则被 onStop() 方法所销毁。如果想继续 SecondActivity,则需要通过 onRestart() 方法来实现。如果内存不够,或者结束了 Activity,那么就会通过 onDestroy() 方法结束整个 Activity,必须重新创建才能重新启动该 App。

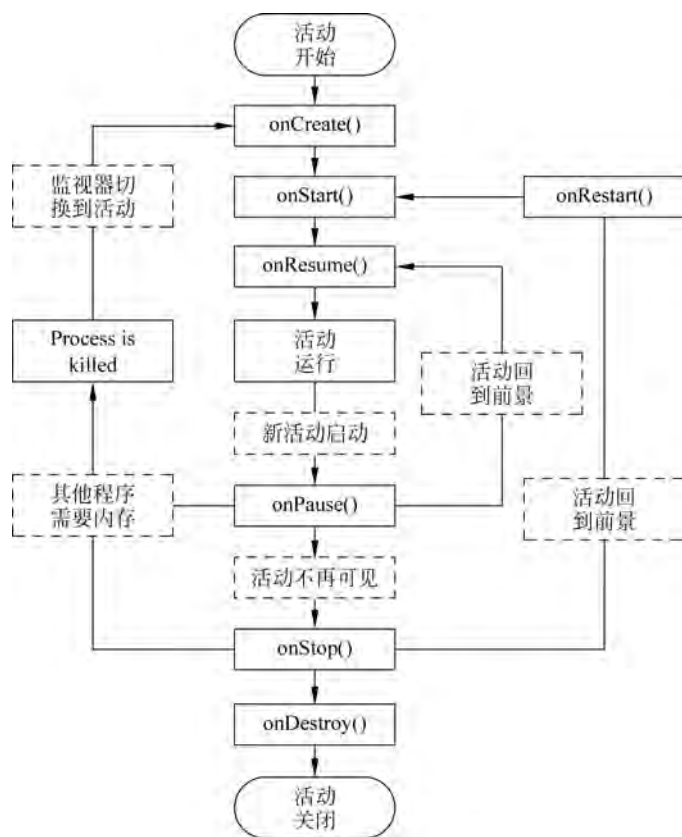


图 5-2 Activity 生命周期循环图

对于 Activity 生命周期而言,有 7 个方法能够帮助 Activity 进入循环流程。这 7 个方法分别为 onCreate()、onRestart()、onStart()、onResume()、onPause()、onStop() 和 onDestroy()。其具体的使用情境、详细说明以及方法之间的承接方法如表 5-1 所示。

表 5-1 Activity 生命周期方法及使用说明

生命周期所用方法	使用情境	详细说明	承接方法
onCreate()	首次创建 Activity 时调用	用户应该在此方法中执行所有正常的静态设置,例如创建视图、将数据绑定到列表等。系统向此方法传递一个 Bundle 对象,其中包含 Activity 的上一状态,不过前提是捕获了该状态	始终后接 onStart()
onRestart()	在 Activity 已停止并即将再次启动前调用	Activity 被重新激活时,就会调用 onRestart 方法	始终后接 onStart()
onStart()	在 Activity 即将对用户可见之前调用	onStart()是 Activity 界面被显示出来的时候执行的	如果 Activity 转入前台,则后接 onResume(); 如果 Activity 转入隐藏状态,则后接 onStop()
onResume()	在 Activity 即将开始与用户进行交互之前调用	此时, Activity 处于 Activity 堆栈的顶层,并具有用户输入焦点	始终后接 onPause()
onPause()	当系统即将开始继续另一个 Activity 时调用	此方法通常用于确认对持久性数据的未保存更改、停止动画以及其他可能消耗 CPU 的内容,诸如此类。它应该非常迅速地执行所需操作,因为它返回后,下一个 Activity 才能继续执行	如果 Activity 返回前台,则后接 onResume(); 如果 Activity 转入对用户不可见状态,则后接 onStop(); 如果它在后台仍然可见,则不会停止
onStop()	在 Activity 对用户不再可见时调用	如果 Activity 被销毁,或另一个 Activity(一个现有 Activity 或新 Activity)继续执行并将其覆盖,就可能发生这种情况	如果 Activity 恢复与用户的交互,则后接 onRestart(); 如果 Activity 被销毁,则后接 onDestroy()
onDestroy()	在 Activity 被销毁前调用	这是 Activity 将收到的最后调用。当 Activity 结束(有人对 Activity 调用了 finish()),或系统为节省空间而暂时销毁该 Activity 实例时,可能会调用它。用户可以通过 isFinishing()方法区分这两种情形	此方法为生命周期的结束方法

根据 Activity 官方文档说明,Activity 的生命周期有 3 个重要的循环对。

(1) 整个生命周期:从调用 onCreate()方法直到调用 onDestroy()方法的整个过程。Activity 需要通过 onCreate()方法准备启动全局状态,再通过 onDestroy()方法释放所有的资源。例如,当一个线程在后台通过网络下载数据时,则通过 onCreate()方法创建线程,再通过 onDestroy()方法停止线程。

(2) 可见生命周期:从调用 onStart()方法到调用 onStop()方法之间的过程。之所以称之为可见生命周期,是因为用户可以直观地在屏幕上看到这个过程,并且能够保持需要展示给用户的资源。例如,通过调用 onStart()方法注册一个 BroadcastReceiver,用于监视使 UI 产生变化的内容,再通过调用 onStop()方法取消监视。onStart()方法和 onStop()方法可以在被用户可见和隐藏两种方式切换的时候被多次调用。

(3) 前台生命周期:从调用 onResume()方法到调用 onPause()方法之间的过程。在此过程中,Activity 通过前台与用户产生交互。Activity 从 onResume()到 onPause()进行了十分频繁的切换。例如,当设备处于休眠状态时,将调用 onPause()方法;当 result 和一个新的 Intent 发送给 Activity 时,将调用 onResume()方法。

5.2.3 介绍 Activity 的属性

Activity 作为 Android 的对象,需要通过各种属性的设置来实现。Activity 的属性类型多样,表 5-2 列出了相关属性及其描述和调用方法。

表 5-2 Activity 的属性详表

属 性	描 述	调 用 方 法
android: allowTaskReparenting	是否允许 Activity 更换从属的任务,例如从短信息任务切换到浏览器任务	android: allowTaskReparenting = ["true" "false"]
android: alwaysRetainTaskState	是否保留状态不变,例如切换回 home,再重新打开,Activity 处于最后的状态	android: alwaysRetainTaskState = ["true" "false"]
android: clearTaskOnLaunch	例如 P 是 Activity,Q 是被 P 触发的 Activity,然后返回 Home,重新启动 P,是否显示 Q	android: clearTaskOnLaunch = ["true" "false"]

续表

属 性	描 述	调 用 方 法
android: configChanges	当配置 List 发生修改时,是否调用 onConfigurationChanged() 方法,例 如 " locale navigation orientation"	android: configChanges = [oneormoreof: "mcc" "mnc" "locale" "touchscreen" "keyboard" "keyboardHidden" "navigation" "orientation" "fontScale"]
android: enabled	Activity 是否可以被实例化	android: enabled = ["true" "false"]
android: excludeFromRecents	是否可被显示在最近打开的 Activity 列表里	android: excludeFromRecents = ["true" "false"]
android: exported	是否允许 Activity 被其他程序调用	android: exported = ["true" "false"]
android: finishOnTaskLaunch	是否关闭已打开的 Activity(当用户重新启动这个任务的时候)	android: finishOnTaskLaunch = ["true" "false"]
android: icon	调用图标	android: icon = "drawableresource"
android: label	调用标签	android: label = "stringresource"
android: launchMode	Activity 启动方式: "standard" "singleTop" "singleTask" "singleInstance" 其中前两个为一组,后两个为一组	android: launchMode = [" multiple" " singleTop" " singleTask" "singleInstance"]
android: multiprocess	可以多实例	android: multiprocess = [" true" "false"]
android: name	采用类名的简写方式,查看文档类名的简写格式	android: name = "string"
android: noHistory	是否需要移除这个 Activity(当用户切换到其他屏幕时)。这个属性是在 APIlevel3 中引入的	android: noHistory = ["true" "false"]
android: permission	权限与安全机制解析	android: permission = "string"
android: process	一个 Activity 运行时所在的进程名,所有程序组件运行在应用程序默认的进程中,这个进程名跟应用程序的包名一致	android: process = "string"
android: screenOrientation	Activity 显示的模式,"unspecified" 为默认值;"landscape"为风景画模式,宽度比高度大一些;"portrait" 为肖像模式,高度比宽度大;"user"为用户的设置;另外还有 "behind" "sensor" "nonsensor"	android: screenOrientation = ["unspecified" " user" " behind" "landscape" " portrait" " sensor" "nonsensor"]

续表

属 性	描 述	调 用 方 法
android: stateNotNeeded	是否 Activity 被销毁和成功重启并不保存状态	android: stateNotNeeded=["true" "false"]
android: taskAffinity	Activity 的亲属关系, 默认同一个应用程序下的 Activity 有相同的关系	android: taskAffinity="string"
android: theme	Activity 的样式主题, 如果没有设置, 则 Activity 的主题样式从属于应用程序, 请参见 < application > 元素的 theme 属性	android: theme="resourceortheme"
android: windowSoftInputMode	Activity 主窗口与软键盘的交互模式, 自 APIlevel3 被引入	android: windowSoftInputMode=[one-or-moreof: "stateUnspecified" "stateUnchanged" "stateHidden" "stateAlwaysHidden" "stateVisible" "stateAlwaysVisible" "adjustUnspecified" "adjustResize" "adjustPan"]>

5.3 项目运行

前面对 Activity 生命周期进行了讲解, 本节将通过一个实例让大家生动地体验项目的生命周期是如何进行的, 即如何在 App 项目中创建 Activity、设置 Activity, 以及启动 Activity、关闭 Activity。

在 Android 开发中, 在 src 文件夹下会自动生成一个 MainActivity.java 文件, 其中实现了对 Activity 的定义及 onCreate() 方法的调用, 具体代码如下:

```
-----
public class MainActivity extends ActionBarActivity {

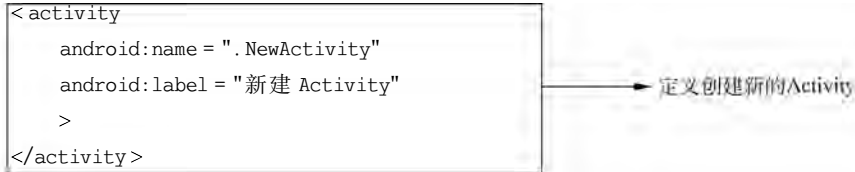
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

通过调用 `onCreate()` 方法实现 `savedInstanceState`, 即每次启动一个 Activity 后, 能够保持在一个界面内容不变, 直到启动另一个 Activity 才能修改界面内容视图。

5.3.1 创建新的 Activity

【例 5.1】 创建一个新的 Activity, 并通过代码实现新建 Activity 的设置、启动。除了新建项目时自带的 Activity 以外, 还需要另外新建其他 Activity, 以实现其他界面内容的布局、调用与交互。为了定义新创建的 Activity, 并在生命周期进行运转时告知 Android 系统, 还需在本项目根目录下的 `AndroidManifest.xml` 文件中进行标注, 具体代码如下:

```
-----  
<application  
    android:label="@string/app_name" >  
    <activity  
        android:label="@string/app_name"  
        android:name=".MainActivity" >  
        <intent-filter >  
            <action android:name="android.intent.action.MAIN" />  
  
            <category android:name="android.intent.category.LAUNCHER" />  
        </intent-filter >  
    </activity>  
    <activity  
        android:name=".NewActivity"  
        android:label="新建 Activity"  
    >  
    </activity>  
</application>
```



将新建的 Activity 定义为 `NewActivity` 的名字后, 继续通过 Android Studio 创建 `NewActivity` 活动。具体过程如下:

步骤 1: 右击 `layout` 文件夹, 选择 `New→Activity→Basic Activity` 命令, 如图 5-3 所示。

步骤 2: 在弹出的 `New Android Activity` 对话框中填写新建 Activity 的相关配置, 包括 Activity Name、Layout Name、Title、Package Name 等信息, 如图 5-4 所示。

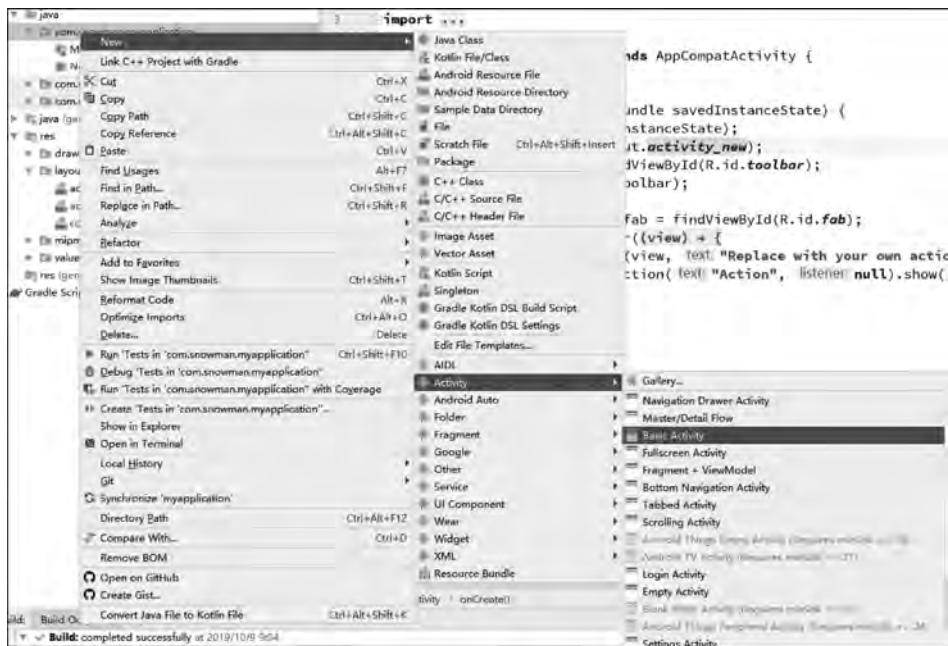


图 5-3 新建 Activity



图 5-4 填写新建 Activity 的相关配置

由 Android Studio 自动生成的代码如下：

```
-----  
package com.mingrisoft;  
  
import android.app.Activity;  
import android.os.Bundle;  
  
public class NewActivity extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        // TODO Auto-generated method stub  
        super.onCreate(savedInstanceState);  
    }  
  
}
```

步骤 3：为了使新建的 NewActivity 运行时有新的界面可以调用，需要为 NewActivity 新建一个布局文件，命名为 newactivity.xml，如图 5-5 所示。然后在 NewActivity.java 文件中添加一句脚本，内容为调用 setContentView() 方法为 NewActivity 指定布局文件 newactivity.xml，具体代码如下：

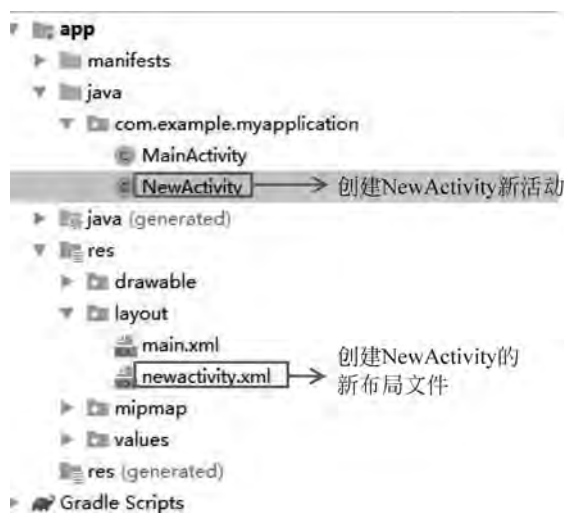


图 5-5 在项目中创建新活动与新布局的结构

```

-----
package com.mingrisoft;

import android.app.Activity;
import android.os.Bundle;

public class NewActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.newactivity);
    }

}

```

增加的脚本

5.3.2 为新建 Activity 设置属性

为了避免新建 Activity 启动时抛出异常,需要在 AndroidManifest.xml 文件中对 NewActivity 进行属性设置,具体代码如下:

```

-----
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.mingrisoft"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="15" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name=".MainActivity" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

```

```

<activity
    android:icon="@drawable/ic_launcher"
    android:name=".NewActivity"
    android:label="新建 Activity"
    android:launchMode="singleTask"
    android:screenOrientation="portrait"
    android:windowSoftInputMode="stateHidden"
>
</activity>
</application>

```

→ 为NewActivity设置属性

```
</manifest>
```

5.3.3 启动 Activity

在本项目中不止一个 Activity,因此在启动 Activity 时需要调用 startActivity() 方法,其语法格式为:

```
Public void startActivity(Intent newintent)
```

其中,Intent 用于 Activity 之间的数据传递,每个 Intent 都要与一个 Activity 相对应。该方法应编写于 layout 文件夹下新建的 newactivity.xml 文件中,具体代码如下:

```

<?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:orientation="vertical" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

    />

    Intent intent = new Intent(MainActivity.this, ToStartActivity.class);
    startActivity(intent);
</LinearLayout>

```

5.3.4 关闭 Activity

关闭 Activity 的方法较为简单,如果只有一个 Activity,只需调用 finish() 方法即可,其

语法格式如下：

```
Public void finish()
```

但如果有多多个 Activity,则需要调用 finishActivity()方法来指定关闭 Activity 对象, SDK 的官方说明文档如下：

```
+ public void finishActivity (int requestCode)
+ Since: API Level 1
+ Force finish another activity that you had previously started with startActivityForResult(Intent, int).
+ Parameters requestCode The request code of the activity that you had given to startActivityForResult(). If there are multiple activities started with this request code, they will all be finished.
```

```
package com.mingrisoft;
```

```
import android.app.Activity;
import android.os.Bundle;
```

```
public class NewActivity extends Activity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        // TODO Auto-generated method stub
```

```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.newactivity);
```

```
        Button btn1 = (Button)findViewById(R.id.btn1);
```

```
        btn1.setOnClickListener(new OnClickListener() {
```

```
            @Override
```

```
            public void onClick(View v) {
```

```
                ActivityA.this.finishActivity(1);
```

```
            }
```

```
        });
```

```
    }
```

5.4 项目结案

本项目通过实例的方式从 Activity(活动类)的 4 种基本状态谈起,再梳理 Activity 的生命周期,并在此基础上演示了如何创建新的 Activity、设置 Activity 以及控制 Activity 状态等操作。总结本项目内容,需要大家对以下 3 个部分有较为深刻的理解：

- (1) 创建新 Activity 的流程。
- (2) 启动 Activity 与关闭 Activity 的方法。
- (3) Activity 的生命周期运转方式。

5.5 项目练习

1. 开发一款 App,并在其中设置两个 Activity,调用 startActivity()方法实现两个 Activity 的启动。
2. 开发一款 App,并在其中创建 3 个 Activity,调用 finishActivity()方法实现依次结束 3 个 Activity。
3. 请用一款 App 的程序代码,对照 Activity 生命周期示意图,详细解析 App 的生命周期运转情况。