Android移动应用开发基础教程

讲授：葛新

第2章 Android核心组件——活动

本章主要内容：

1. 活动是什么
2. 活动的基本操作
3. 在活动中使用Intent
4. 在活动之间传递数据
5. 活动的生命周期
6. 活动的启动模式

2.1 活动是什么

- 活动是Android的一个核心应用组件，它主要用于实现应用功能逻辑，并通过界面显示数据或接收用户输入。一个应用程序可以包含零个或多个活动。没有活动的应用程序，用户将无法看到程序界面，这种应用程序通常在后台运行，不涉及用户交互。
- 从用户的角度看，活动具有如下特点。
  - 可通过返回键退出活动。
  - 可通过Home键返回桌面。
  - 可在活动中启动另个一个界面，此时按返回键返回前一个活动。

2.2 活动的基本操作

本节主要内容：

1. 为活动绑定自定义视图
2. 启动另一个活动
3. 结束活动

2.2.1 为活动绑定自定义视图

- 在Android Studio中演示

2.2.1 为活动绑定自定义视图

- 在Android Studio中演示

2.2.3 结束活动

- 在Android Studio中演示

2.3 在活动中使用Intent

Intent是Android应用中的一种消息传递机制，通过Intent对象实现其他应用组件之间的通信。通常，Intent用于启动活动、启动服务以及发送广播。Intent可分为两种：显式Intent和隐式Intent

本节主要内容：

1. 显式Intent
2. 隐式Intent
3. Intent过滤器
4. 从网页中启动活动

2.3.1 显式Intent

- 显式Intent指在创建Intent对象时，指定了要启动的特定组件。
- 实例演示

2.3.2 隐式Intent

- 显式Intent指明了要启动的组件，隐式Intent则相反，它不指明要启动的组件，而是指明要执行的操作，让系统去选择可完成该操作的组件。

1. 启动同一个应用中的活动

   实例演示

2. 启动另一个应用中的活动

   实例演示

2.3.3 Intent过滤器

- Intent过滤器主要用于声明应用组件可接收的Intent操作、数据和其他设置。

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

在清单文件AndroidManifest.xml声明Intent过滤器时，可使用下面的3个元素：

- **<action>元素**：在其name属性中声明组件可接受的Intent操作，操作名称可以是自定义的文本字符串或者ndroid.intent.action类的常量。
- **<category>元素**：在其name属性中声明组件可接受的Intent类别，类别名称通常为android.intent.category类中的常量。如果要让活动响应隐式Intent，则必须将过滤器的类别设置为android.intent.category.DEFAULT。如果没有在Intent过滤器中声明DEFAULT类别，则隐式Intent不会解析该组件。
- **<data>元素**：声明数据URI的scheme、host、port、path等，或者是数据的MIME类型

在代码中创建Intent对象时，可调用下列方法为Intent对象添加操作、类别、数据或其他属性：

- **setAction()**：设置Intent对象操作。也可在Intent对象构造函数中指定操作。
- **addCategory()**：为Intent对象添加类别。
- **setData()**：设置数据URI。
- **setType()**：设置MIME类型。
- **setDataAndType()**：setData()和setType()会相互抵消彼此的设置，要同时设置URI和MIME类型，则需调用setDataAndType()。

2.3.4 从网页中启动活动

- Android允许在浏览器中启动活动。在Intent过滤器中包含BROWSABLE类别，即表示当前活动可从浏览器启动。

2.4 在活动之间传递数据

本节主要内容：

1. 传递简单数据
2. 传递Bundle对象
3. 传递对象
4. 获取活动返回的数据

2.4.1 传递简单数据

- 简单数据指字符串、整数、浮点数等各种简单数据类型的数据，或者是这些简单数据类型的数据。
- putExtra(name,value)方法可将指定的数据封装到Intent对象中。其中，name为表示数据名称的字符串，value为要传递的各种简单数据类型的值。
- 要获取Intent对象中封装的简单数据，可调用各种get*XXX*Extra()方法。

- getCharExtra(String name,char defaultValue)：从Intent对象中获取指定name的char类型数据。
- getFloatExtra(String name,float defaultValue)：从Intent对象中获取指定name的float类型数据。
- getFloatArrayExtra(String name)：从Intent对象中获取指定name的float类型数组。
- getIntArrayExtra(String name)：从Intent对象中获取指定name的int类型数组。
- getIntExtra(String name, int defaultValue)：从Intent对象中获取指定name的int类型数据。
- getStringArrayExtra(String name)：从Intent对象中获取指定name的String类型数组。
- getStringExtra(String name)：从Intent对象中获取指定name的String类型数据。

2.4.2 传递Bundle对象

- 将各种简单数据封装到一个Bundle对象中，再将Bundle对象封装到Intent对象中传递给启动的活动。
- Bundle对象的各种putXXX(String key,XXX value)方法，可将XXX类型的数据封装到其中，对应的用getXXX(String key)方法从其中获取数据。
- Bundle对象准备好之后，调用putExtras(bundle)或putExtra(name,bundle)方法将其封装到Intent对象中。要从Intent对象中获取Bundle对象时，调用对应的getExtras()或getBundleExtra()方法即可。

2.4.3 传递对象

- 自定义的类对象，不能像简单数据一样直接封装到Intent对象中。Android系统要求封装到Intent对象中支持序列化。
- 让类实现Java内置的Serializable接口，或者实现Android提供的Parcelable接口，即可使类对象支持序列化。

1. 使用实现Serializable接口的类对象

   实例演示

2. 使用实现Parcelable接口的类对象

   实例演示

2.4.4 获取活动返回的数据

- 要获得活动中返回的数据，则需要使用startActivityForResult(intent,requestCode)方法来启动活动。
- 其中，参数intent是一个Intent对象，用于封装需要传递给活动的数据。
- 参数requestCode为请求码，是一个整数，用来标识当前请求。一个活动可能会接收到其他不同活动的请求，从活动返回时，它会原样返回接收到的请求码。
- 在处理返回结果时，可通过请求码判断是不是从所请求的活动返回。

- 在当前活动中需重载onActivityResult()方法来处理返回结果，其代码基本结构如下：

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
……
}
```

- 参数requestCode为从所请求的活动返回的它所接收到的请求码。resultCode为结果代码，常量RESULT_CANCELED表示用户取消了操作，RESULT_OK表示用户正确完成了操作。data为请求活动返回的Intent对象，从中可获取返回的数据。
- 在请求的活动中，用setResult(resultCode,intent)方法设置返回结果，resultCode为结果代码，intent为封装了返回数据的Intent对象。

2.5 活动的生命周期

活动的生命周期指活动从第一次创建到被销毁的整个时间。在一个生命周期内，活动可能存在多种状态。深入了解活动的生命周期，有助于更合理管理应用程序资源，设计出效率更高的应用。

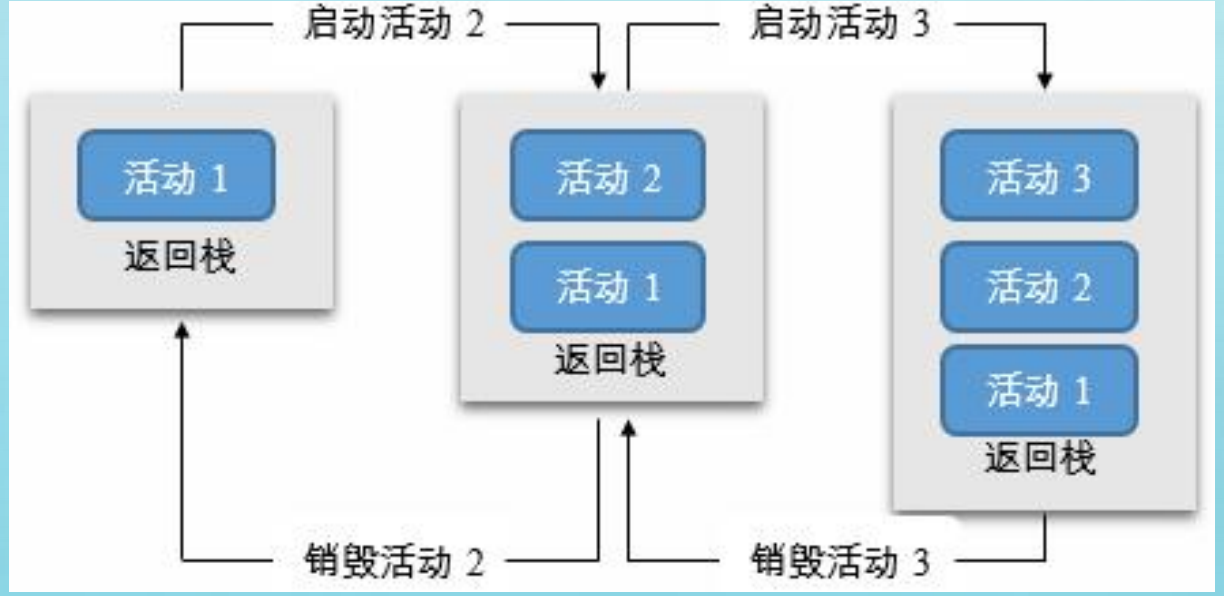
本节主要内容：

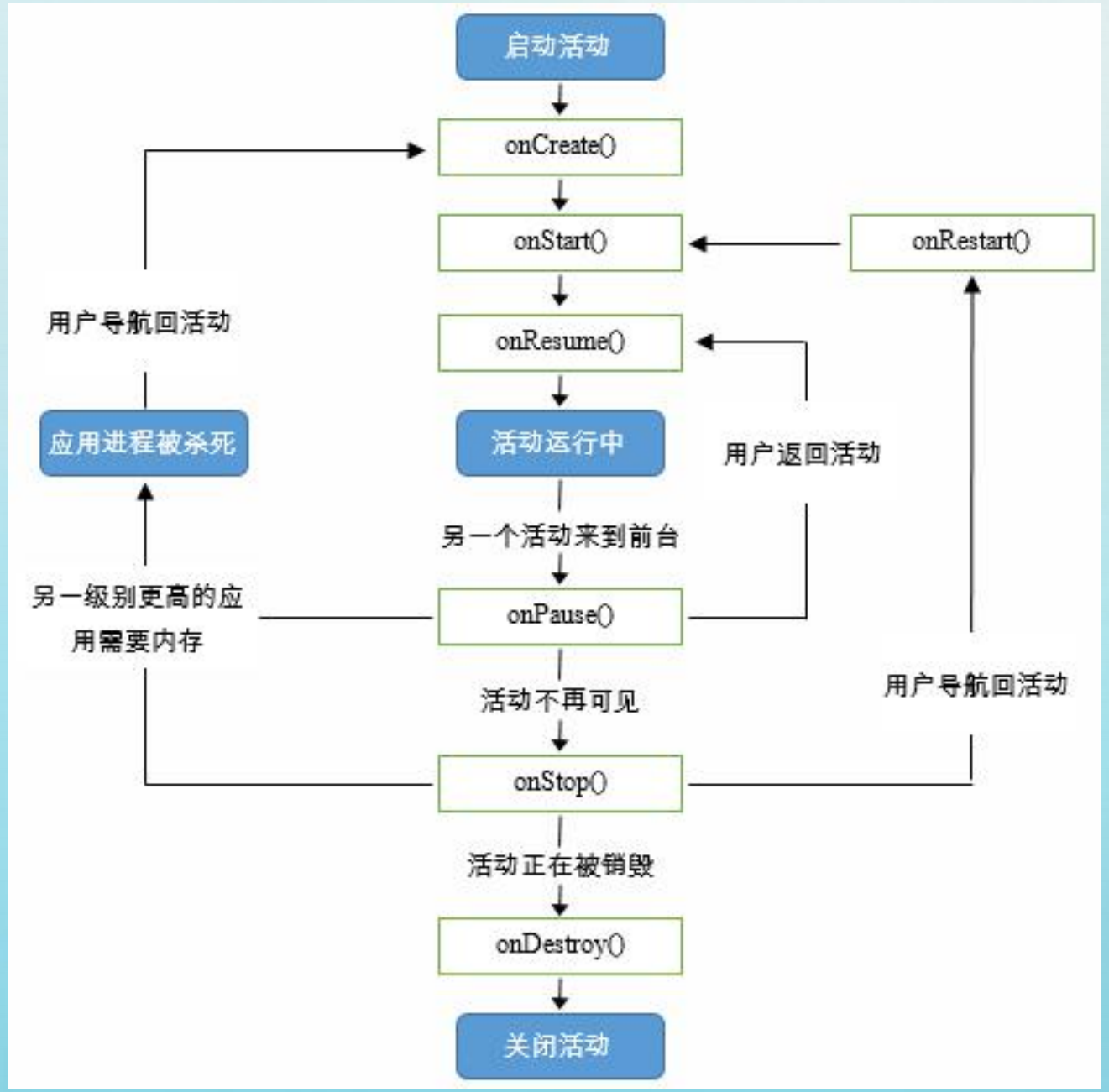
1. 返回栈、活动状态及生命周期回调
2. 检验活动的生命周期

2.5.1 返回栈、活动状态及生命周期回调

1、返回栈

- 一个应用可能包含多个活动。Android系统使用堆栈（也称返回栈）来管理活动。
- 返回栈遵循“先进后出”原则。




启动活动
onCreate()
onStart()
onRestart()
用户导航回活动
onResume()
应用进程被杀死
活动运行中
用户返回活动
另一个活动来到前台
另一级别更高的应
用需要内存
onPause()
活动不再可见
用户导航回活动
onStop()
活动正在被销毁
onDestroy()
关闭活动

2.5.2 检验活动的生命周期

- 实例演示

2.6 活动的启动模式

活动总是拥有特定的启动模式，启动模式决定了Android系统如何在任务的返回栈中管理活动的实例。活动的启动模式有4种：standard、singleTop、singleTask和singleInstance。可在清单文件使用<activity>元素的launchMode属性来指定活动的启动模式。

本节主要内容：

1. standard模式
2. singleTop模式
3. singleTask和singleInstance模式

2.6.1 standard模式

- standard是活动的默认启动模式。在前面的所有内容中用到的活动，其启动模式都是standard。系统在启动standard模式的活动时，不会检查任务返回栈顶部中是否已经有该活动，总是创建一个新的活动实例，将其放到返回栈顶部。

2.6.2 singleTop模式

- 如果活动是singleTop模式，在启动活动时，系统首先检查任务返回栈，若栈顶活动是相同活动的实例，则直接使用该活动，不会再创建新的实例。

2.6.3 singleTask和singleInstance模式

- singleTask启动模式表示一个任务中只能存在活动的一个实例。在启动singleTask模式的活动时，系统如果发现任务返回栈中有该活动实例，则将该实例之上的所有活动出栈，使该实例成为栈顶活动。如果入伍返回栈中没有活动实例，则创建一个新的活动实例，将其放到栈顶。
- 与singleTop模式有点类似，但singleInstance启动模式表示活动只允许“设备”中存在活动的一个实例，在启动singleInstance模式的活动时，系统会为活动实例创建一个新的任务返回栈。设备中的所有应用可共享该活动实例。