

3.1 应用程序基本组成

- Android系统没有使用常见的应用程序入口点的方法（例如main()方法），应用程序是由组件组成的，组件可以调用相互独立的基本功能模块，根据完成的功能不同，Android划分了四类核心组件，即Activity、Service、BroadcastReceiver和ContentProvider，各组件之间的消息传递通过Intent完成。
- **3.1.1 Activity**
- Activity是Android应用程序核心组件中最基本的一种，是用户和应用程序交互的窗口。在Android应用程序中，一个activity通常对应一个单独的视图。一个Android应用程序是由一个或多个Activity组成的，这些Activity相当于Web应用程序中的网页，用于显示信息，并且相互之间可以进行跳转。和网页跳转不同的是，Activity之间的跳转可以有返回值。
- 当新打开一个视图时，之前的那个视图会被置为暂停状态，并且压入历史堆栈中，用户可以通过回退操作返回到以前打开过的视图。Activity是由Android系统进行维护的，它有自己的生命周期，即“产生、运行、销毁”，但是这过程中会调用许多方法，如创建onCreate()、激活onStart()、恢复onResume()、暂停onPause()、停止onStop()、销毁onDestroy()和重启onRestart()等。

- ### 3.1.2 Service

- Service是一种类似于Activity但是没有视图的程序，它没有用户界面，可以在后台运行很长的时间，相当于操作系统中的一个服务。Android定义了两种类型的Service，即本地Service和远程Service。本地Service是只能由承载该Service的应用程序访问的组件，而远程Service是供在设备上运行的其他应用程序远程访问的Service。
- 通过Context.startService(Intent service)可以启动一个Service，通过Context.bindService()可以绑定一个Service。

- ### 3.1.3 BroadcastReceiver

- BroadcastReceiver也就是“广播接收者”的意思，顾名思义，它就是用来接收来自系统和其他应用程序的广播，并作出回应。在Android系统中，当有特定事件发生时就会产生相应的广播。广播体现在方方面面，例如，当开机过程完成后系统会产生一条广播，接收到这条广播就能实现开机启动服务的功能；当网络状态改变时系统会产生一条广播，接收到这条广播就能及时地做出提示和保存数据等操作；当电池电量改变时，系统会产生一条广播，接收到这条广播就能在电量低时告知用户及时保存进度等。
- BroadcastReceiver不能生成UI，通过NotificationManager来通知用户有事件发生，对于用户来说是隐式的。BroadcastReceiver的注册方式有两种，一种是在AndroidManifest.xml中进行静态注册，另一种可以在运行时的代码中使用Context.registerReceiver()进行动态注册。只要注册了BroadcastReceiver，即使对应的事件广播来临时应用程序并未启动，系统也会自动启动该应用程序对事件进行处理。另外，用户还可以通过Context.sendBroadcast()将自己的Intent对象广播给其他的应用程序。

- ### 3.1.4 ContentProvider

- 文件、数据库等数据在Android系统内是私有的，仅允许被特定应用程序直接使用。在两个程序之间数据的交换或共享由ContentProvider实现。
- ContentProvider类实现了一组标准方法的接口，从而能够让其他的应用保存或读取ContentProvider提供的各种数据类型。

- ### 3.1.5 Intent

- Intent并不是Android应用程序四大核心组件之一，但是其重要性无可替代，因此在这里我们做一下简单介绍。
- Android应用程序核心组件中的三大核心组件——Activity、Service、BroadcastReceiver，通过消息机制被启动激活，而所使用的消息就是Intent。Intent是对即将要进行的操作的抽象描述，承担了Android应用程序三大核心组件相互之间的通信功能。

3.2 Activity

- Activity是Android组件中最基本也是最为常见的组件。Activity 是用户接口程序，原则上它会提供给用户一个交互式的接口功能，几乎所有的Activity都要和用户打交道，也有人把它比喻成Android的管理员。需要在屏幕上显示什么，用户在屏幕上做什么，处理用户不同操作等都由Activity来管理和调度。
- Activity提供用户与Android系统交互的接口，用户通过Activity来完成自己的目的，例如打电话、拍照、发送E-mail、查看地图等。每个Activity都提供一个用户界面窗口，一般情况下，该界面窗口会填满整个屏幕，但是也可以比屏幕小，或者浮在其他的窗口之上。
- 一个Android应用程序通常由多个Activity组成，但是其中只有一个为主Activity，其作用相当于Java应用程序中的main函数，当应用程序启动时，作为应用程序的入口首先呈现给用户。Android应用程序中的多个Activity直接可以相互调用以完成不同工作。当新的Activity被启动的时候，之前的Activity会被停止，但是不会被销毁，而是被压入到“后退栈（Back Stack）”的栈顶，而新启动的Activity获得焦点，显示给用户。“后退栈”遵循“后入先出”的原则。当新启动的Activity被使用完毕，用户单击“Back”按钮时，当前的Activity会被销毁，而原先的Activity会被从“后退栈”的栈顶弹出并且激活。
- 当Activity状态发生改变时，都会通过状态回调函数通知Android系统。而程序编写人员可以通过这些回调函数对Activity进行进一步的控制。
- 下面对Activity生命周期及其涉及的回调函数进行简单介绍。

• 3.2.1 Activity的生命周期

- 从本质上讲，Activity在生命周期中共存在三个状态，分别为：
- 运行态：运行态指Activity运行于屏幕的最上层并且获得了用户焦点。
- 暂停态：暂停态是指当前Activity依然存在，但是没有获得用户焦点。在其之上有其他Activity处于运行态，但是由于处于运行态的Activity没有遮挡住整个屏幕，当前Activity有一部分视图可以被用户看见。处于暂停态的Activity保留了自己所使用的内存和用户信息，但是在当系统极度缺乏资源的情况下，有可能会被杀死以释放资源。
- 停止态：停止态是指当前Activity完全被处于运行态的Activity遮挡住，其用户界面完全不能被用户看见。处于停止态的Activity依然存活，也保留了自己所使用的内存和用户信息，但是一旦系统缺乏资源，停止态的Activity就会被杀死以释放资源。
- Activity在声明周期中从一种状态到另一种状态时会激发相应的回调方法，这些回调方法包括：
- onCreate (Bundle savedInstanceState)。创建activity时调用。设置在该方法中，还以Bundle的形式提供对以前储存的任何状态的访问。其中参数savedInstanceState对象是用于保存Activity的对象的状态。
- onStart()。activity变为在屏幕上对用户可见时调用。

- **onResume()**。Activity开始与用户交互时调用（无论是启动还是重启一个活动，该方法总是被调用的）。
- **onPause()**。当Android系统要激活其他Activity时，该方法被调用，暂停或收回CPU和其他资源时调用。
- **onStop()**。Activity被停止并转为不可见阶段时调用。
- **onRestart()**。重新启动已经停止的Activity时调用。
- **onDestroy()**。Activity被完全从系统内存中移除时调用，该方法被调用可能是有人直接调用finish()方法或者系统决定停止该活动以释放资源。
- 上面7个生命周期方法分别在4个阶段按着一定的顺序进行调用，这4个阶段如下：
- **启动Activity**。在这个阶段依次执行3个生命周期方法：onCreate、onStart和onResume。
- **Activity失去焦点**。如果在Activity获得焦点的情况下进入其他的Activity或应用程序，这时当前的Activity会失去焦点。在这一阶段，会依次执行onPause和onStop方法。
- **Activity重获焦点**。如果Activity重新获得焦点，会依次执行3个生命周期方法：onRestart、onStart和onResume。

- 关闭Activity。当Activity被关闭时系统会依次执行3个生命周期方法：onPause、onStop和onDestroy。
- Activity生命周期中方法调用过程如图3.1所示。

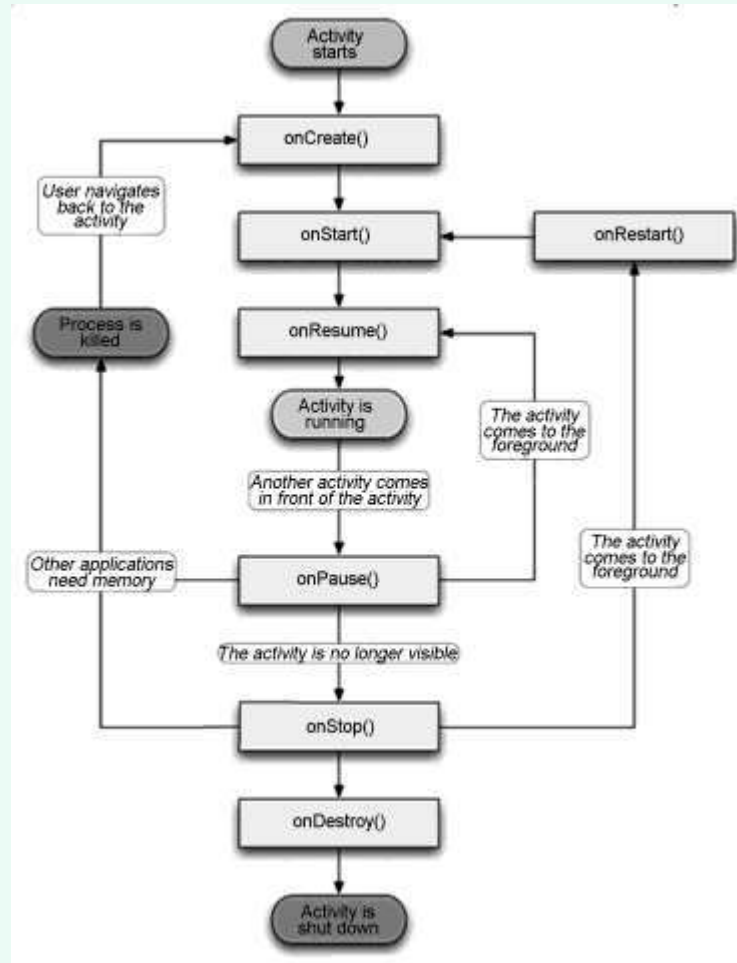


图3.1 Activity生命周期

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：<https://d.book118.com/055203224240011214>