

Introduction to Activities

The Activity class is a crucial component of an Android app, and the way activities are launched and put together is a fundamental part of the platform's application model. Unlike programming paradigms in which apps are launched with a `main()` method, the Android system initiates code in an Activity instance by invoking specific callback methods that correspond to specific stages of its lifecycle.

The concept of activities

- An activity provides the window in which the app draws its UI. This window typically fills the screen, but may be smaller than the screen and float on top of other windows. Generally, one activity implements one screen in an app.
- Most apps contain multiple screens, which means they comprise multiple activities. Typically, one activity in an app is specified as the main activity, which is the first screen to appear when the user launches the app. Each activity can then start another activity in order to perform different actions. For example, the main activity in a simple e-mail app may provide the screen that shows an e-mail inbox. From there, the main activity might launch other activities that provide screens for tasks like writing e-mails and opening individual e-mails.

Activity lifecycle

Over the course of its lifetime, an activity goes through a number of states. You use a series of callbacks to handle transitions between states. The following sections introduce these callbacks.

onCreate()

onStart()

onResume()

onPause()

onStop()

onRestart()

onDestroy()

onCreate method

You must implement this callback, which fires when the system creates your activity. Your implementation should initialize the essential components of your activity: For example, your app should create views and bind data to lists here. Most importantly, this is where you must call `setContentView()` to define the layout for the activity's user interface.

OnStart method

As `onCreate()` exits, the activity enters the Started state, and the activity becomes visible to the user. This callback contains what amounts to the activity's final preparations for coming to the foreground and becoming interactive.

onResume

The system invokes this callback just before the activity starts interacting with the user. At this point, the activity is at the top of the activity stack, and captures all user input. Most of an app's core functionality is implemented in the `onResume()` method.

OnPause method

The system calls `onPause()` when the activity loses focus and enters a Paused state. This state occurs when, for example, the user taps the Back or Recents button. When the system calls `onPause()` for your activity, it technically means your activity is still partially visible, but most often is an indication that the user is leaving the activity, and the activity will soon enter the Stopped or Resumed state.

OnStop method

The system calls `onStop()` when the activity is no longer visible to the user. This may happen because the activity is being destroyed, a new activity is starting, or an existing activity is entering a Resumed state and is covering the stopped activity. In all of these cases, the stopped activity is no longer visible at all.

OnRestart method

The system invokes this callback when an activity in the Stopped state is about to restart. `onRestart()` restores the state of the activity from the time that it was stopped.

OnDestroy method

The system invokes this callback before an activity is destroyed.

This callback is the final one that the activity receives. `onDestroy()` is usually implemented to ensure that all of an activity's resources are released when the activity, or the process containing it, is destroyed.

