The Android Lifecycle cheat sheet — part I: Single Activities

Jose Alcérreca Follow
Dec 5, 2017 · 3 min read

Android is designed to empower users and let them use apps in a intuitive way. For example, users of an app might rotate the screen, respond to a notification, or switch to another task, and they should be able to continue using the app seamlessly after such an event.

To provide this user experience, you should know how to manage component lifecycles. A component can be an Activity, a Fragment, a Service, the Application itself and even the underlying process. The component has a lifecycle, during which it transitions through states. Whenever a transition happens, the system notifies you via a lifecycle callback method.

To help us explain how lifecycles work, we've defined a series of scenarios which are grouped according to the components that are present:

Part I: Activities — single activity lifecycle (this post)

Part II: Multiple activities — navigation and back stack

Part III: Fragments — activity and fragment lifecycle

Part IV: ViewModels, Translucent Activities and Launch Modes

The diagrams are also available as a cheat sheet in PDF format for quick reference.

> *Note: these diagrams apply to **Android P / Jetpack 1.0** behavior.*

The following scenarios showcase the default behavior of the components, unless otherwise noted.

If you find errors or you think something important is missing, report it in the comments.

Part I: Activities

Single Activity — Scenario 1: App is finished and restarted

Triggered by:

- The user presses the **Back button**, or
- The `Activity.finish()` method is called

The simplest scenario shows what happens when a single-activity application is started, finished and restarted by the user:

Scenario 1: App is finished and restarted

Managing state

- `onSaveInstanceState` is not called (since the activity is finished, you don't need to save state)
- `onCreate` doesn't have a Bundle when the app is reopened, because the activity was finished and the state doesn't need to be restored.

Single Activity — Scenario 2: User navigates away

Triggered by:

- The user presses the **Home button**
- The user switches to another app (via Overview menu, from a notification, accepting a call, etc.)

Scenario 2: User navigates away

In this scenario the system will stop the activity, but won't immediately finish it.

Managing state

When your activity enters the Stopped state, the **system uses onSaveInstanceState to save the app state in case the system kills the app's process later on** (see below)**.**

Assuming the process isn't killed, the activity instance is kept resident in memory, retaining all state. When the activity comes back to the foreground, the activity recalls this information. You don't need to re-initialize components that were created earlier.

Single Activity — Scenario 3: Configuration changes

Triggered by:

- Configuration changes, like a **rotation**
- User resizes the window in multi-window mode

Scenario 3: Rotation and other configuration changes

Managing state

Configuration changes like rotation or a window resize should let users continue exactly where they left off.

- The activity is completely destroyed, but **the state is saved and restored for the new instance**.
- The Bundle in `onCreate` and `onRestoreInstanceState` is the same.

Single Activity — Scenario 4: App is paused by the system

Triggered by:

- Enabling Multi-window mode (API 24+) and losing the focus
- Another app partially covers the running app (a purchase dialog, a runtime permission dialog, a third-party login dialog…)
- An intent chooser appears, such as a share dialog

Scenario 4: App is paused by the system

This scenario *doesn't* apply to:

- Dialogs in the same app. Showing an `AlertDialog` or a `DialogFragment` won't pause the underlying activity.
- Notifications. User receiving a new notification or pulling down the notification bar won't pause the underlying activity.

Continue reading

- The Android Lifecycle cheat sheet part II — Multiple activities

Android App Development Android Lifecycle