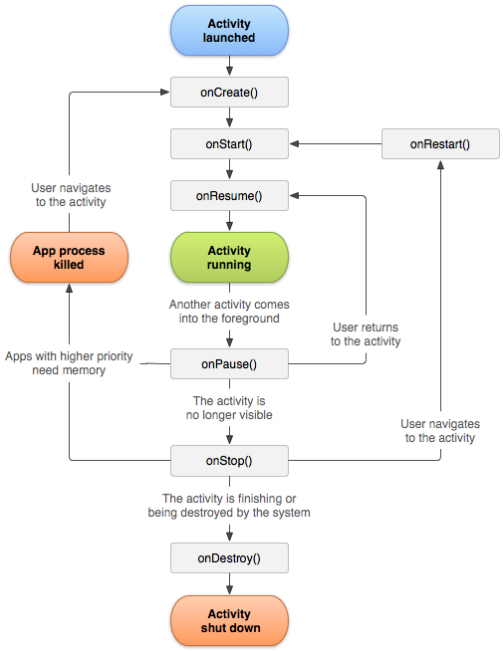
Written by Doğa Tuncer

What is Android Lifecycle?



Every Android application has various stages from start to end. Whenever a user opens, uses, and closes an application these various stages change and help the execution of application tasks smoothly. An app's current state allows developers to manage when a user can open an activity, pause, resume, stop and destroy execution of app. For every stage of these lifecycles, there are specialized **callback methods**. These methods are used and crucial for an easy and unproblematic application run. With the lifecycle callback methods, developers can shape the way that activities behave during the usage of app by assigning true callback methods to perform specific tasks. These method assignments must be matched and appropriate to a given change of state of the activity. Simply, with using right callback methods, developers can tell the app to do right tasks at the right time and prevent problems caused by complexity and messiness. Good implementation of lifecycle callbacks and efficient management of Android lifecycle is essential for a good performant app.

To navigate app and different transitions between different stages of activity lifecycle, activity class has six main callbacks: onCreate(), onStart(), onResume(), onPause(), onStop(), and onDestroy(). To understand further, some basic fundamental information about callback methods is given below.

1- onCreate()

This method fires first when the system creates the activity. On specific activity creation, the activity state is named as *Created* state. In this method one must perform basic application startup logic that will be occurred only once for the entire life of the activity.

2- ***onStart()***

When an activity enters the Started state, this callback method is called. Method call makes the activity visible to the user, while app prepares for the activity becoming interactive with user. As a clear example, with this method call, app can initialize the code that maintains the UI (user interface).

3- ***onResume()***

After starting, activity enters its' Resumed state,and it comes to the foreground. When this is achieved the system invokes the onResume callback. This is the state in which the app fully interacts with the user. The app stays in Resumed until something happens and shift focus away from the app. A shifting and taking focus away event can be such as 'receiving a phone call, the user's navigating to another activity, or the device screen's turning off.'

4- **onPause()**

This method is the first indication that the user is either leaving or changing focus from the focused activity (but this doesn't necessarily mean that the activity is being destroyed). onPause method is used for pausing or adjusting operations that should not continue (or should continue in moderation). Generally it is expected that the activity will resume shortly after onPause method.

5- **onStop()**

When previously focused activity is not visible to the user any longer, it is said that it has entered the stopped state, and the system invokes onStop method next. An onStop example can be a newly launched activity which covers the entire screen. But onStop also will be called when the activity has already finished running and is close to the ending.

6- **onDestroy()**

onDestroy callback method is called right before the activity is destroyed. Below there are two listed onDestroy triggers:

1. either the activity is finishing (due to the user completely dismissing the activity or due to finish() being called on the activity), or
2. the system is temporarily destroying the activity due to a configuration status change (such as device rotation or multi-window mode)

Some Common Problems that Occur Due to Inefficient Use of Callback Methods

- Crashing may occur if the user receives a phone call or switches to another app during using app.
- Valuable system resources can be consumed when the user is not actively using it.
- Losing the user's progress can occur if user leaves app and returns to it at a later time.
- Crashing or losing the user's progress may occur when the screen rotates between landscape and portrait orientation.
- Some event may interrupt app execution during Resumed state. This is the most common case.

References

The activity lifecycle : Android developers. Android Developers. (n.d.). Retrieved September 29, 2022, from https://developer.android.com/guide/components/activities/activity-lifecycle