Because I'll forget it if I don't write it down...

search

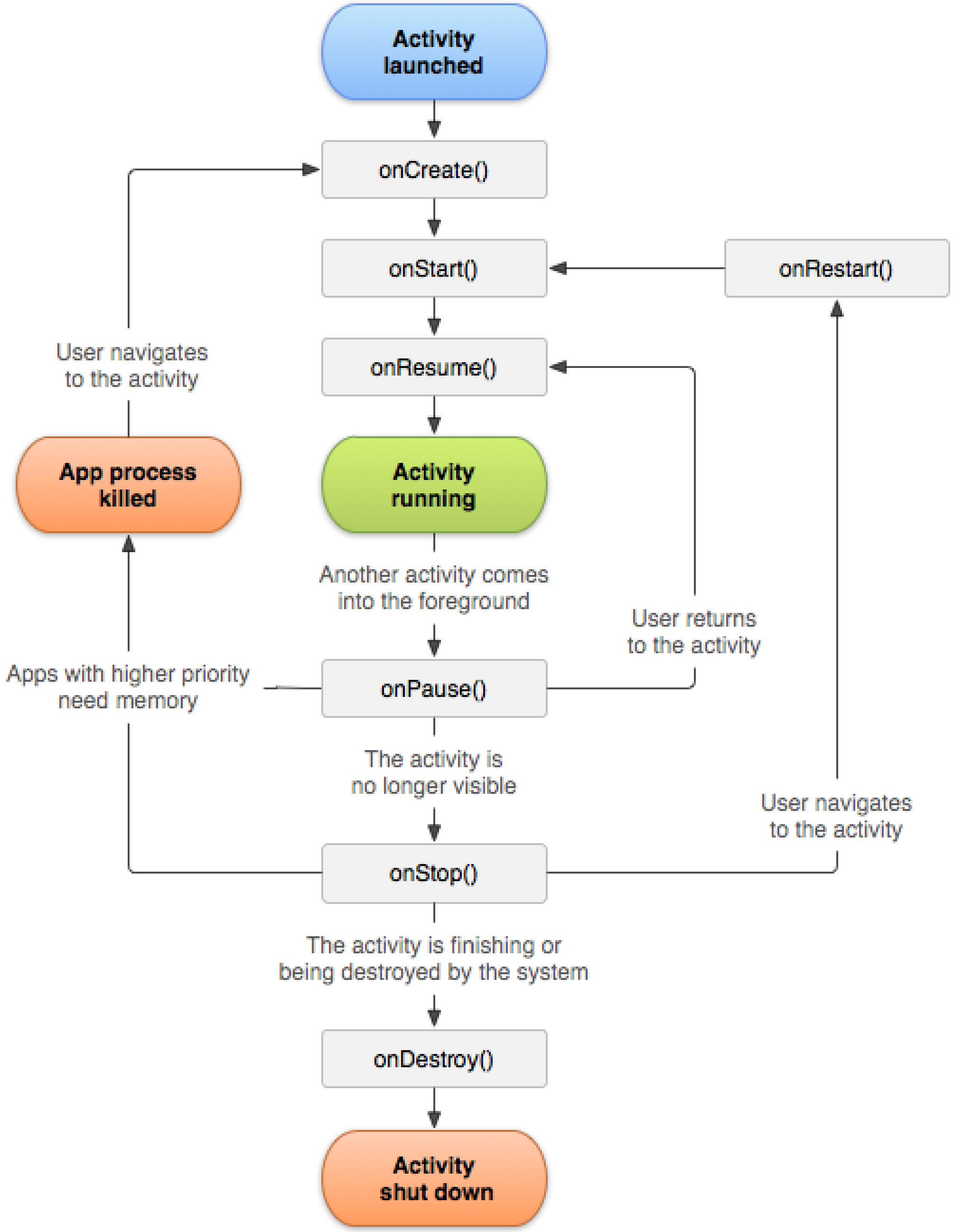
June 18, 2012

Android Activity Lifecycle Gotcha

My Android app has been live in Google Play for 6 months, but I'm still encountering strange bugs and behaviours, making big mistakes, and learning surprising things. The latest surprise has come with a recent flush of ICS users who's devices are putting more significant demands on my handling of the Activity lifecycle, specifically with relation to managing state.

tl:dr; - beware when invoking `startActivityForResult` that onActivityResult is invoked *before* onResume!

Before I get to the problems, lets have a quick look at the Activity lifecycle. I'm going to "borrow" google's lifecycle diagram:


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Some important things to remember here are:

- Apps typically consist of *multiple* Activity's, and *each* Activity follows the above lifecycle while your app is running.

- When your Activity starts a *child* Activity (with `startActivity` or `startActivityForResult`), both the `onPause` and `onStop` lifecycle methods of the parent Activity *should* be called, in that order.
- When an Activity is invoked *as* a child Activity, its lifecycle will be completed by the time the parent Activity is fully in control again (at least `onCreate`, `onStart`, `onResume`, and `onPause` will have been invoked).
- Your Activity can be killed off *at any time* after its `onPause` has completed, without necessarily passing through `onStop` or `onDestroy`. It is critically important to remember that *this includes situations where your Activity is on the back-stack waiting for a result from a child Activity, or even when it is still visible but mostly covered by a dialog!*

With regard to the last point its worth familiarising yourself with the way Android manages Processes.

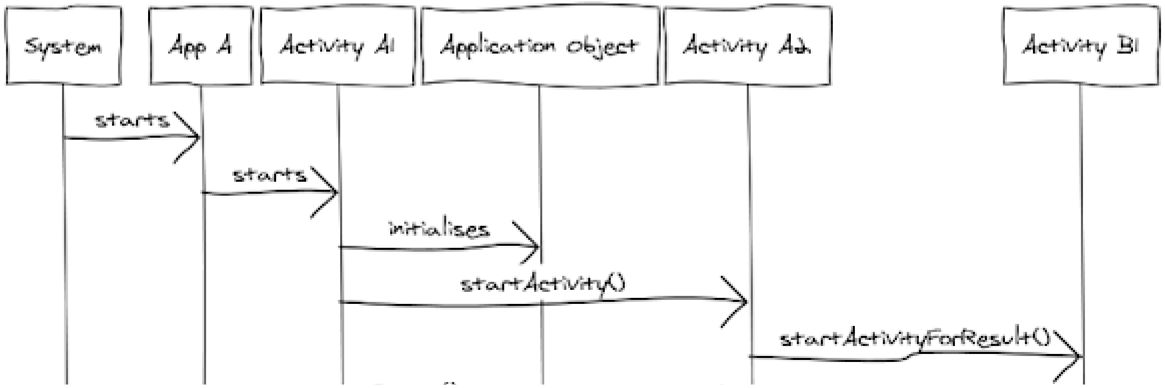
State, and the Application object

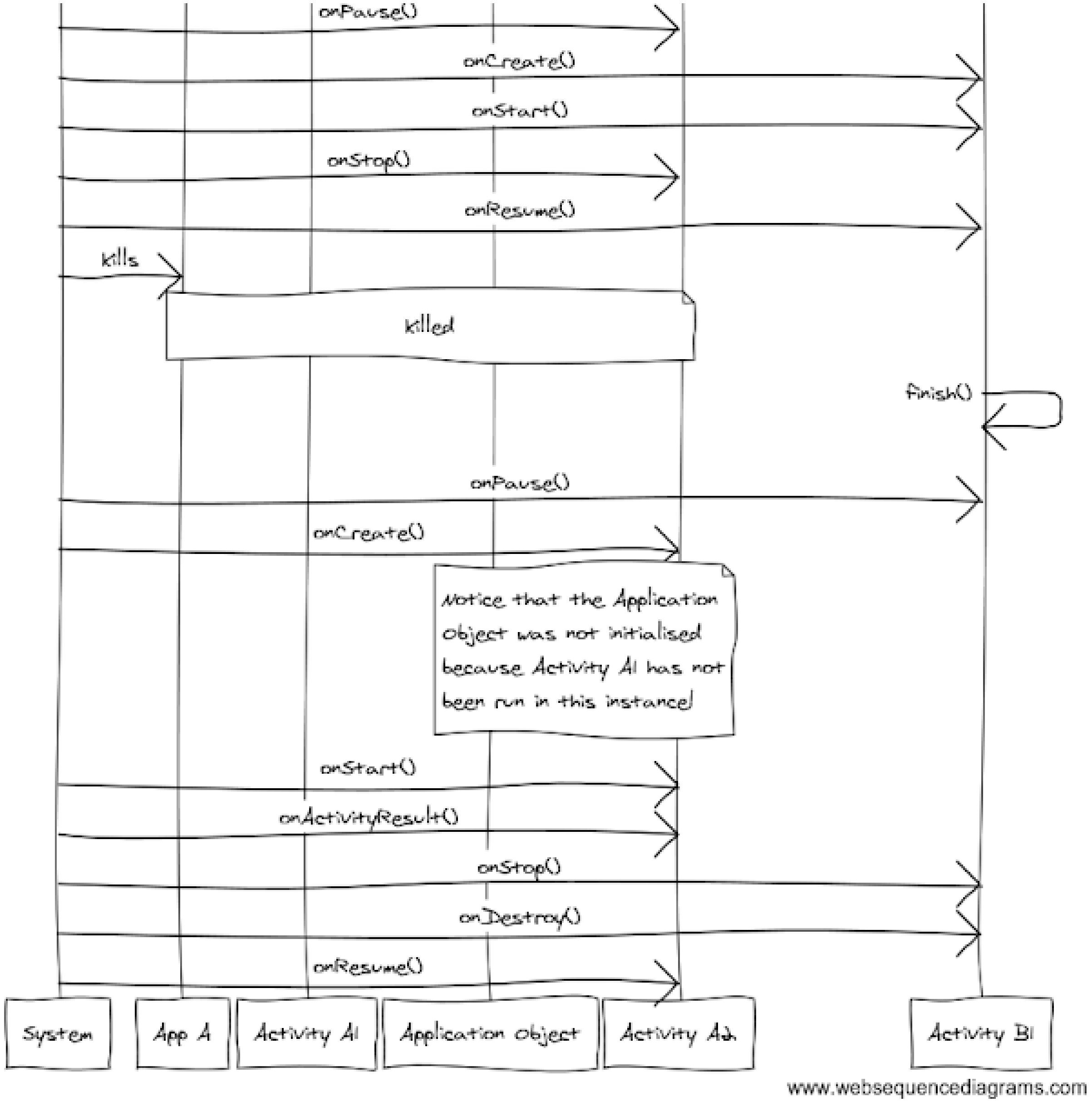
One simple way, you might think, to manage state without worrying too much about the `Activity` lifecycle is to use the Application object. Android allows you to specify your own class (`extends Application`), which your Activity's can access through `getApplication`.

That's nice. It needs care though, since the process that your Application object lives in can be killed and restarted at (perhaps) unexpected junctures. Take this scenario:

1. App A starts with Activity A1, which sets up some state in the Application object.
2. Activity A1 starts Activity A2, which uses the state in the Application object.
3. Activity A2 fires an `Intent` for Activity B1 of App B and expects some result (lets say we fired an `Intent` asking for an image to be captured by the camera app).
4. App B starts, and launches Activity B1.
5. Activity B1 is memory-heavy, so *the system shuts down App A (completely kills its process), even though it is on the back-stack waiting for a result*.
6. Activity B1 returns, app A's Application object is created, Activity A2 is started again but Activity A1 never launched in the lifetime of this Application object so *does not get the opportunity to set up the state of the Application object*.

The sequence diagram might look something like this:





Clearly, if Activity A2 relies on A1 having run first to set up the application state, there's going to be trouble as soon as A2 starts trying to access that state after resuming from B2. *If you're going to use the Application object to manage state, make sure that it is set up as part of the Application's own lifecycle methods*.

Now, the gotcha that's been hurting me is this: I assumed that onActivityResult would be invoked *after* onResume. Turns out this is not the case, and in fact onActivityResult was getting called long before my state was re-initialised in onResume.

On my devices I never suffered from this because my process was not being killed and the state was still present in memory at the point when onActivityResult was invoked!

- Android
- Activity Lifecycle
- Large Heap
- Fragmentation

10 Comments **Steve Liles' Blog** 1 **Login**

♡ **Recommend** Tweet Share **Sort by Best**

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

heila • 6 years ago

I see your problem. I found this post https://groups.google.com/f.... The basic idea is to use a Singleton object in the application class.

16 ^ | ⌄ • Reply • Share ›

Steve Liles Mod → heila • 6 years ago

Hi Heila... Yes, a lazy singleton is an alternative to using the Application object, although I have a slight allergy to Singletons in general :).

There's no actual problem with using the Application object _provided_ you initialize it correctly using its own lifecycle methods (as opposed to expecting that the "first" Activity will perform initialization of the Application object, which won't always be the case).

My only real problem was the order of invocation of onActivityResult vs onResume - having sorted that out I don't have any problems.

^ | ⌄ • Reply • Share ›

Diego Dagum • 4 years ago

You can always force the worst-case scenario (activity in the back stack being killed) if you go to Settings / Developer options / Apps section, and there you make sure that the option "Don't keep activities" is checked.

1 ^ | ⌄ • Reply • Share ›

Brijesh Thakur • 3 years ago

I am facing the similar issue while using Facebook SDK. SDK returns data using onActivityResult() which I need to use in my Fragment. But the issue is, Since It gets called before onResume(), I need to persist the data somewhere, so that I can use it in onResume() to avoid crash. Since there is no guarantee that, My Fragment will be still valid during onActivityResult().

What are the ways, I can pass activity result to the fragment which is not yet ready ??

⌃ | ⌄ • Reply • Share ›

Steve Liles Mod ➦ Brijesh Thakur • 3 years ago

Maybe call startActivityForResult from your Fragment, rather than your Activity? (in which case the result will be delivered to the Fragment's onActivityResult callback).

If, for some reason, you _need_ to receive the result in the Activity - say, because more than one Fragment needs access to the result - you probably need to look up your fragments by id and only pass the result to the ones that are available, and maybe hold the result as a member variable in Activity so that the Fragments can later ask for it ... or ... broadcast the result to all interested parties via a publish/subscribe channel of some sort (eventbus or BroadcastReceiver or ...)

⌃ | ⌄ • Reply • Share ›

Brijesh Thakur ➦ Steve Liles • 3 years ago

Thanks Steve !!
In my case, I am having onActivityResult() in Fragment. And I need to pass the data from onActivityResult() to onResume() of same fragment.

I guess, holding the result as a member variable will work right ? And what's wrong in using getArguments()/setArguments() ?

⌃ | ⌄ • Reply • Share ›

Steve Liles Mod ➦ Brijesh Thakur • 3 years ago

You wouldn't be able to use setArguments/getArguments unless you were handling the activity result in the activity and calling setArguments _before_ creating the fragment.

A member variable is the way to go.

⌃ | ⌄ • Reply • Share ›

Brijesh Thakur ➦ Steve Liles • 3 years ago

Thanks !!! Really appreciate it.

Thanks !!! Really appreciate it.

⌃ | ⌄ • Reply • Share ›

Philip Jeffrey Trowe • 3 years ago

If you would like to learn how to use the following objects to write an Android application that displays a vertically upward scrolling Rainbow of colours in a FREE video, then click the link at the end of this comment:

. LinearLayout
. Activity
. View
. Canvas
. ArrayList
. Paint

http://androidprogrammering...

⌃ | ⌄ • Reply • Share ›

Philip Jeffrey Trowe • 3 years ago

Why not take a look at my blog about how to create an Android app that displays an Image in an ImageView control of the main Activity at the full width of the screen.

The app uses the following Android SDK objects:

. LinearLayout
. ImageView
. Bitmap
. Activity
. XML layout
. LayoutParams
. Display

Also:
. layout_width
. layout_height
. orientation
. id
. vertical

. vertical
. match_parent

XML attributes and values are covered.

Click the link BELOW! to see

http://androidprogrammering...

⌃ | ⌄ • Reply • Share ›

ALSO ON **STEVE LILES' BLOG**

Creating comic-book style icons with Gimp

3 comments • 7 years ago

AvatarSaweki — Is there a way to change the white to transparency? So I can use only the dots as a png. on any background i …

Dynamic Proxies in Java

1 comment • 5 years ago

AvatarMaze — Simple, useful, beautiful. Helped me to finally use dynamic proxies. Thank you, sir!

Porting isChangingConfigurations to API-levels below 11

1 comment • 5 years ago

AvatarEdeiwmurk Nivek — #onSaveInstanceState(...) may be called when an activity goes into the backstack, …

Subversion 1.7 Eclipse integration in Ubuntu 12

5 comments • 6 years ago

AvatarSteve Liles — do you have the javahl client selected in the eclipse svn prefs?

Subscribe **Add Disqus to your siteAdd DisqusAdd** **Disqus' Privacy PolicyPrivacy PolicyPrivacy**

- Tweet
- G+

Recent Tweets

Follow @steveliles

Recent Posts

- Building OpenCV as an .aar for Android - Jan 25, 2016
- Is my Android app currently foreground or background? - Apr 21, 2014
- Asynchronous Android - Dec 18, 2013
- Porting isChangingConfigurations to API-levels below 11 - Oct 21, 2013
- Android SSL - Certificate not trusted - May 01, 2013

Older posts...
Setting up embedded Jetty 8...
Testing Android apps