Mobile App Development - Android

Android Activity Lifecycle

Jigisha Patel

Agenda

- Android Activity Lifecycle
- Lifecycle State Transition
- Detecting orientation change
- Log
- Toast

Android Activity

- Presented to the user as full-screen windows (screens)
- Single, focused thing that the user can do
- Almost all activities interact with the user

- Takes care of creating a window for you in which you can place your UI with setContentView(View)
- Types of Activity : full screen window, floating window and multi-window mode

Four states of Activity

1. Active or Running

- foreground of the screen
- currently interacting with user

2. Visible

- activity has lost focus but still presented to the user
- possible if a new non-full-sized or transparent activity has focus on top of your activity

Four states of Activity

cont…

3. Stopped or Hidden

- completely obscured by another activity
- retains all state and member information
- no longer visible to the user
- often killed by the system when memory is needed elsewhere

4. Destroyed

- System can drop the activity from memory by either asking it to finish, or simply killing its process
- Must be completely restarted and restored to its previous state before displaying to the user

Activity Stack

- Activities in the system are managed as activity stacks.
- When a new activity is started, it is usually placed on the top of the current stack and becomes the running activity.

- The previous activity always remains below it in the stack and will not come to the foreground again until the new activity exits.
- There can be one or multiple activity stacks visible on screen.

Back Stack

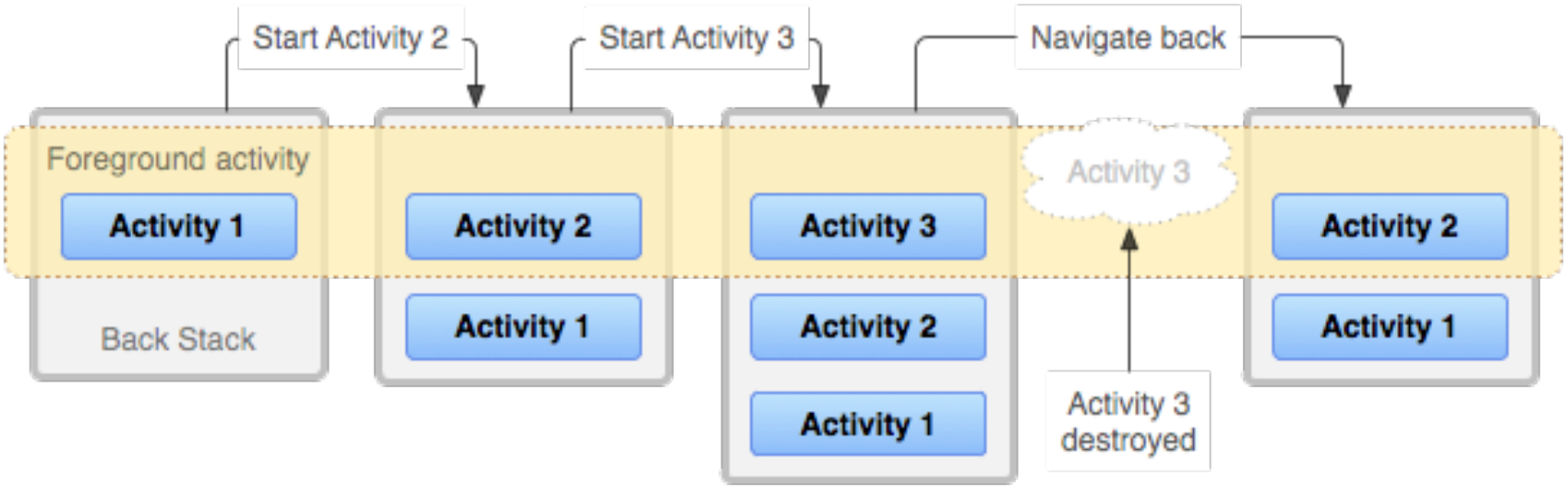
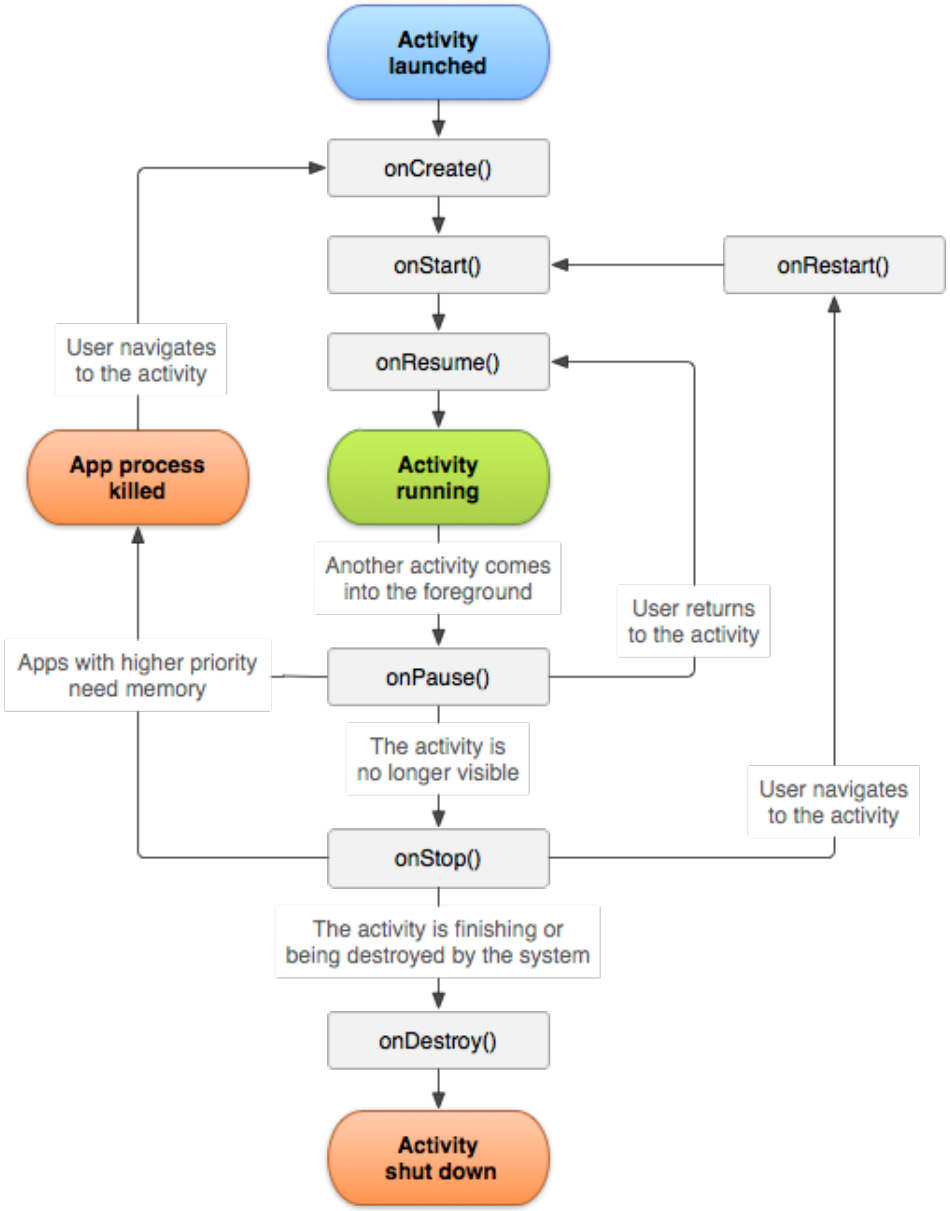


Fig (a) : Example of Activity Stack and Back Stack

Image source: developers.android.com

Activity Lifecycle

For details visit [Android Developer Documents]



Android App Lifecycle cont…

- To navigate transitions between stages of the activity lifecycle, the Activity class provides a core set of six callbacks:
  - onCreate(),
  - onStart(),
  - onResume(),
  - onPause(),
  - onStop(), and
  - onDestroy().
- The system invokes each of these callbacks as an activity enters a new state.

onCreate()

- Called when the activity is first created.
- This is where you should do all of your normal static set up: create views, bind data to lists, etc.

- This method also provides you with a Bundle containing the activity's previously frozen state, if there was one.
- Always followed by onStart().

- Invoked only once during the entire activity lifecycle.

onRestart()

- Called after your activity has been stopped, prior to it being started again.
- Always followed by onStart()

onStart()

- Called when the activity is becoming visible to the user.

onResume()

- Called when the activity will start interacting with the user.
- At this point your activity is at the top of its activity stack, with user input going to it.
- Always followed by onPause().

onPause()

- Called when the activity loses foreground state, is no longer focusable or before transition to stopped/hidden or destroyed state.

onStop()

- Called when the activity is no longer visible to the user.

onDestroy()

- The final call you receive before your activity is destroyed.
- This can happen either because the activity is finishing or because the system is temporarily destroying this instance of the activity to save space.
- Invoked only once during the entire activity lifecycle.

Log

android.util.log allows you to write log messages

Log.d – debug messages

Log.e – error or exception messages

Log.i – info messages

Log.v – verbose messages

Log.w – warn messages

Log cont…

- Syntax:

```
Log.d(Activity name, message)
```

- Example:

```
private static final String TAG = "MainActivity";

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Log.d(TAG, msg: "onCreate invoked");
}
```

Orientation Change

- By default, the activity will invoke onDestroy() and onCreate() callbacks when certain configurations of the app are changed.

- This could be avoided by indicating not to restart the activity for specific configuration changes.

- For orientation change, it could be done by adding following attribute to the Activity element in AndroidManifest.xml file

Orientation Change cont…

- To declare that your activity handles a configuration change, edit the appropriate <activity> element in your manifest file to include the android:configChanges attribute with a value that represents the configuration you want to handle.

Orientation Change cont…

- If you want to manually handle orientation changes in your app you must declare the "orientation", "screenSize", and "screenLayout" values in the android:configChanges attributes.

- You can declare multiple configuration values in the attribute by separating them with a pipe | character as shown below.

```
<activity android:name=".MainActivity"
    android:configChanges="screenSize|orientation|screenLayout|keyboardHidden">
```

Orientation Change cont...

orientation

- This value prevents restarts when the screen orientation changes.

screenSize

- It also prevents restarts when orientation changes.

screenLayout

- This value is necessary to detect changes that can be triggered by devices such as foldable phones and convertible Chromebooks.

keyboardHidden

- It prevents restarts when the keyboard availability changes.

Android Toast

- A toast provides simple feedback about an operation in a small popup.
- It only fills the amount of space required for the message and the current activity remains visible and interactive.
- Toasts automatically disappear after a timeout.

```
Toast.makeText( context: this,  text: "landscape", Toast.LENGTH_SHORT).show()
```

References

- https://developer.android.com/reference/android/util/Log
- https://developer.android.com/reference/android/widget/Toast
- https://developer.android.com/guide/topics/resources/runtime-changes
- https://developer.android.com/guide/components/activities/activity-lifecycle