Ganpat University
॥ विद्यया समाजोत्कर्षः ॥
U.V. Patel College of Engineering

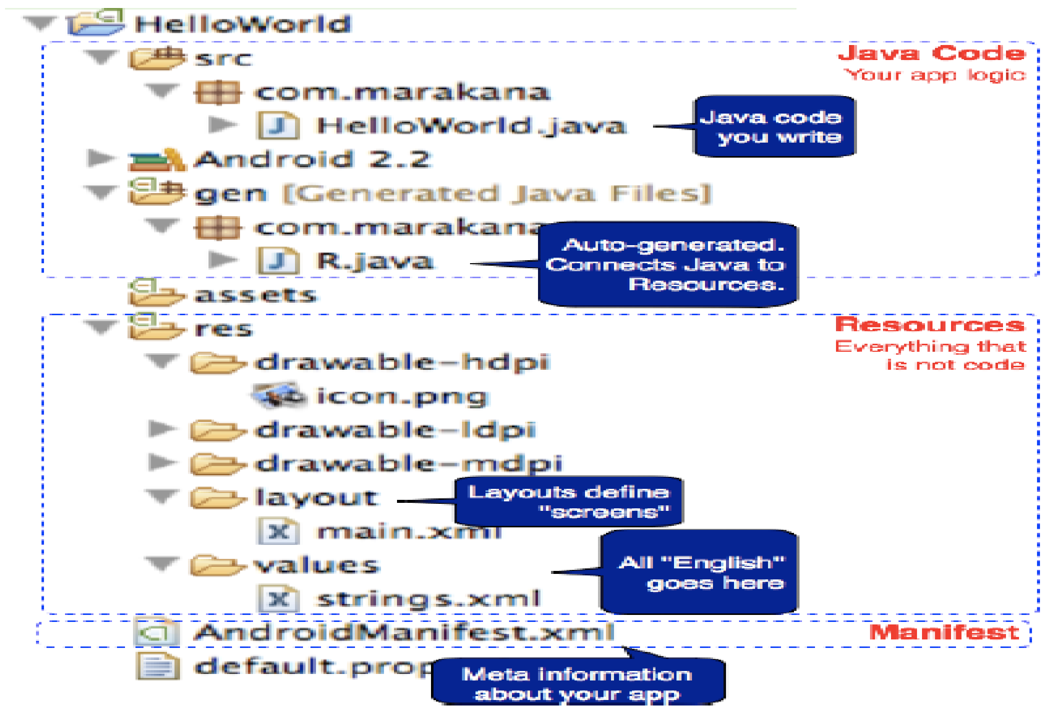
Mobile Application Development

By,
Prof. Himanshu H Patel,
Prof. Hiten M Sadani

U. V. Patel College of Engineering, Ganpat University

Ganpat University || विद्यया समाजोत्कर्षः || U.V. Patel College of Engineering

Anatomy of Android Application

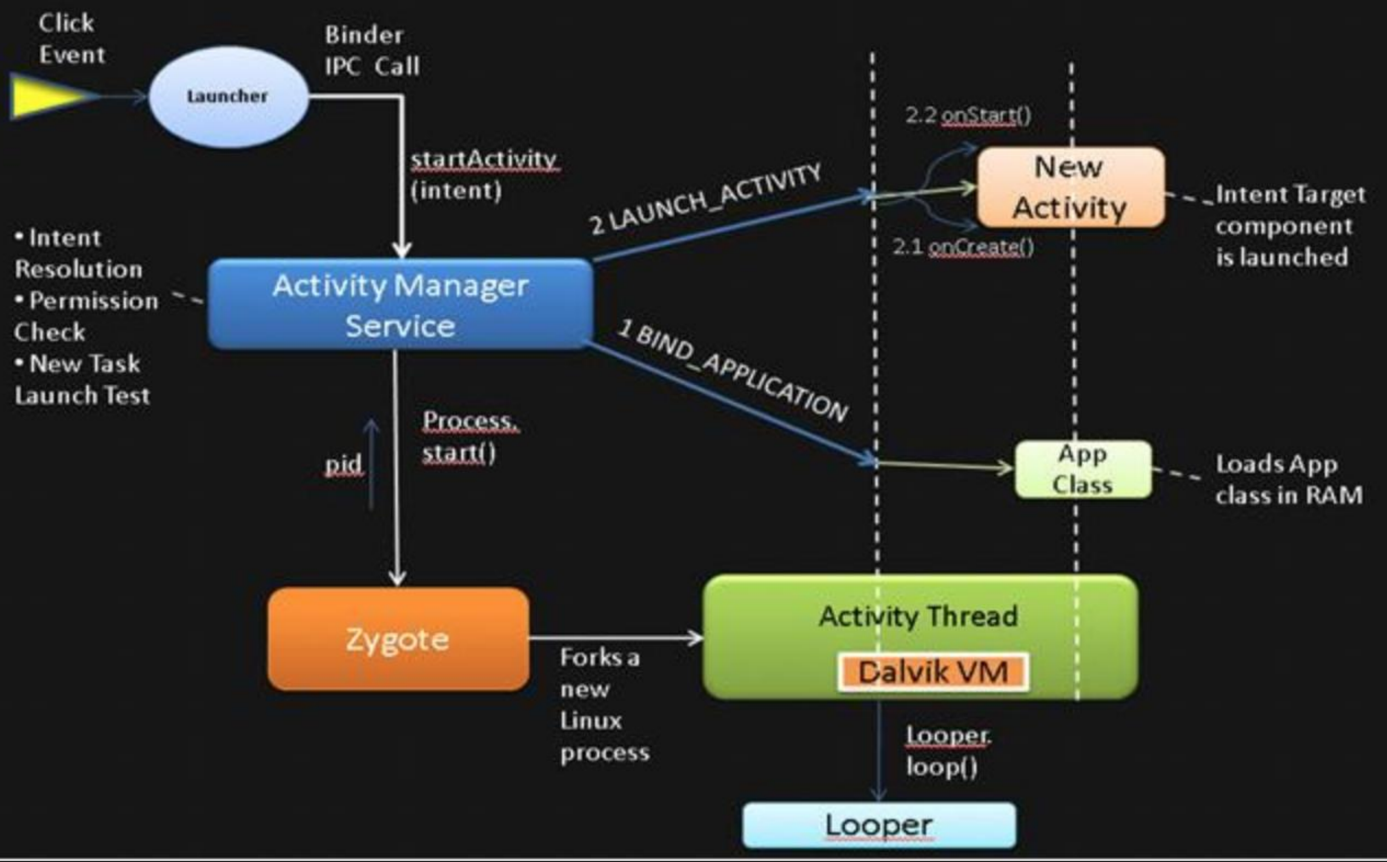


Anatomy of An App

Java Code
+
XML and Other Resources
+
Manifest File
=
Android App

of Dalvik virtual machine and pre-loads all the common classed used by the application framework

Application Launch



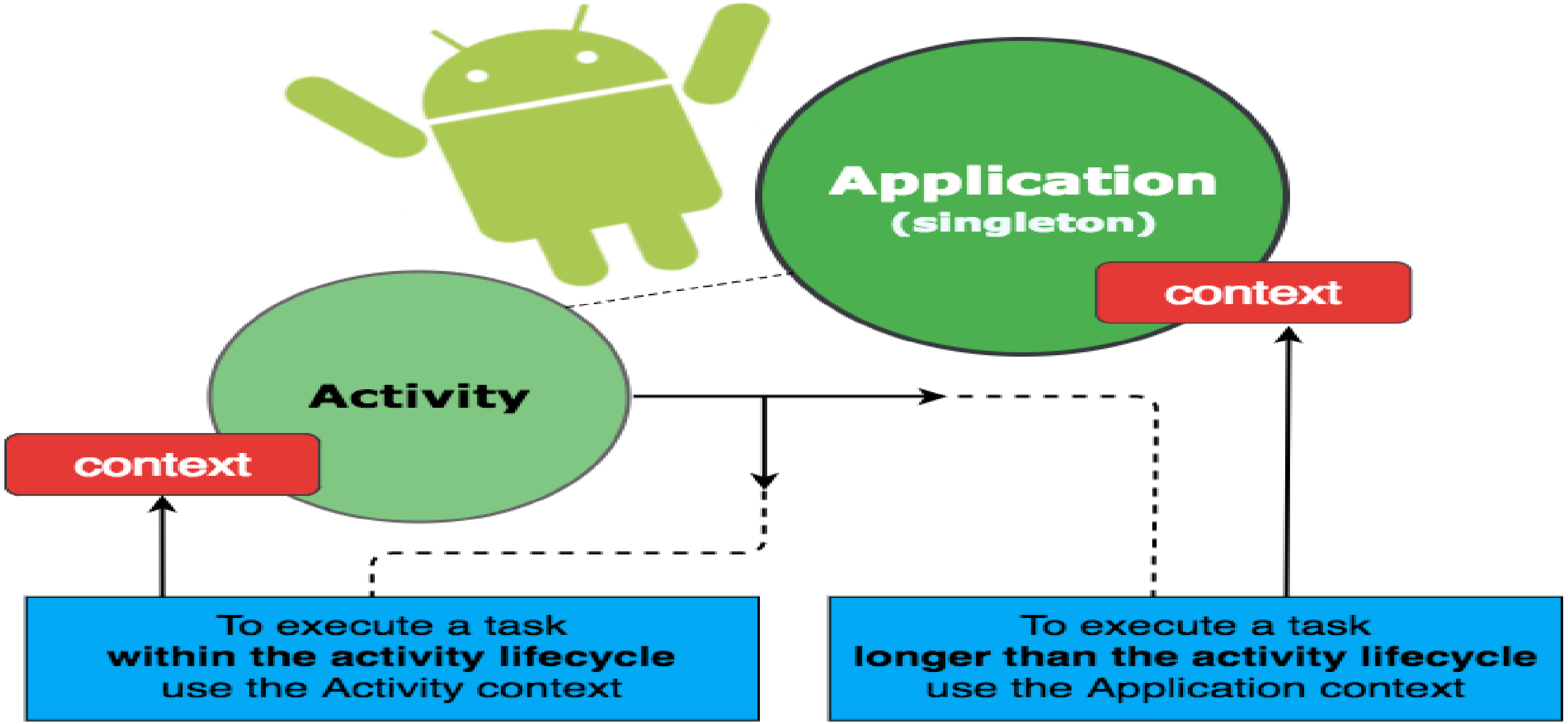
Anatomy of Android Application

- An application consists of one or more *components that are defined in the application's manifest file.*
- *A component can be one of the following:*
- ***Context***
- ***Activity***
- ***Service***
- ***Broadcast receiver***
- ***Content provider***

Context:

- **The context is the central command center for an Android application.**
- All application-specific functionality can be accessed through the context
- You can retrieve the Context for the current process using the getApplicationContext()
- Context context = getApplicationContext();

Context:



To execute a task **within the activity lifecycle** use the Activity context

To execute a task **longer than the activity lifecycle** use the Application context

Context:

Context is an abstract class, implemented in the Android system by some classes like e.g. Activity, Service, Application.

Why I need it?

Context:

- *We can use it to retrieve resources, start a new Activity, show a dialog, start a system service, create a view, and more.*
- *Context type depends on the Android component that lives within it.*

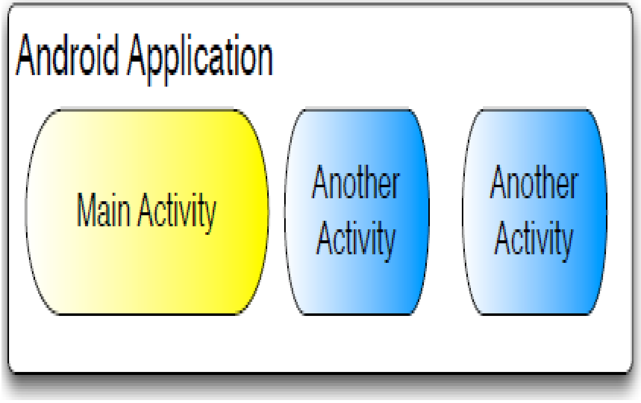
Context:

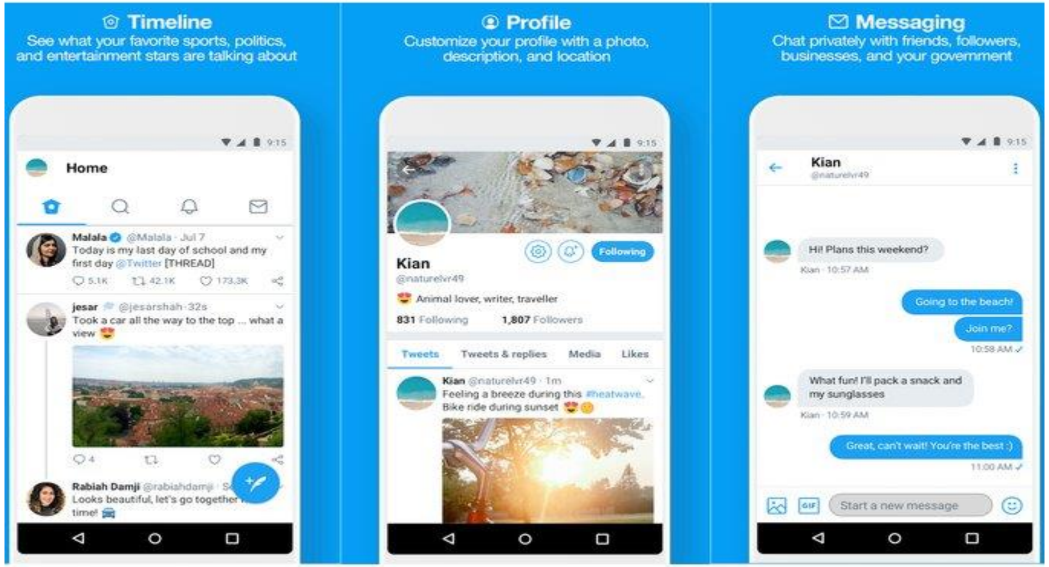
- *getContext() — returns the Context which is linked to the Activity from which is called,*
- *getApplicationContext() — returns the Context which is linked to Application which holds all activities running inside it,*

Activity:

- An Activity represents a screen or a window. Sort of.
- *each activity is independent of the others.*
- *one of the activities is marked as the first one that should be presented to the user when the application is launched.*

Activity:




Timeline
See what your favorite sports, politics, and entertainment stars are talking about

Home

Malala @Malala · Jul 7
Today is my last day of school and my first day @Twitter [THREAD]
5.1K 42.1K 173.3K

jesar @jesarshah · 32s
Took a car all the way to the top ... what a view 😍

Rabiah Damji @rabiahdamji
Looks beautiful, let's go together time! 🚘

Profile
Customize your profile with a photo, description, and location

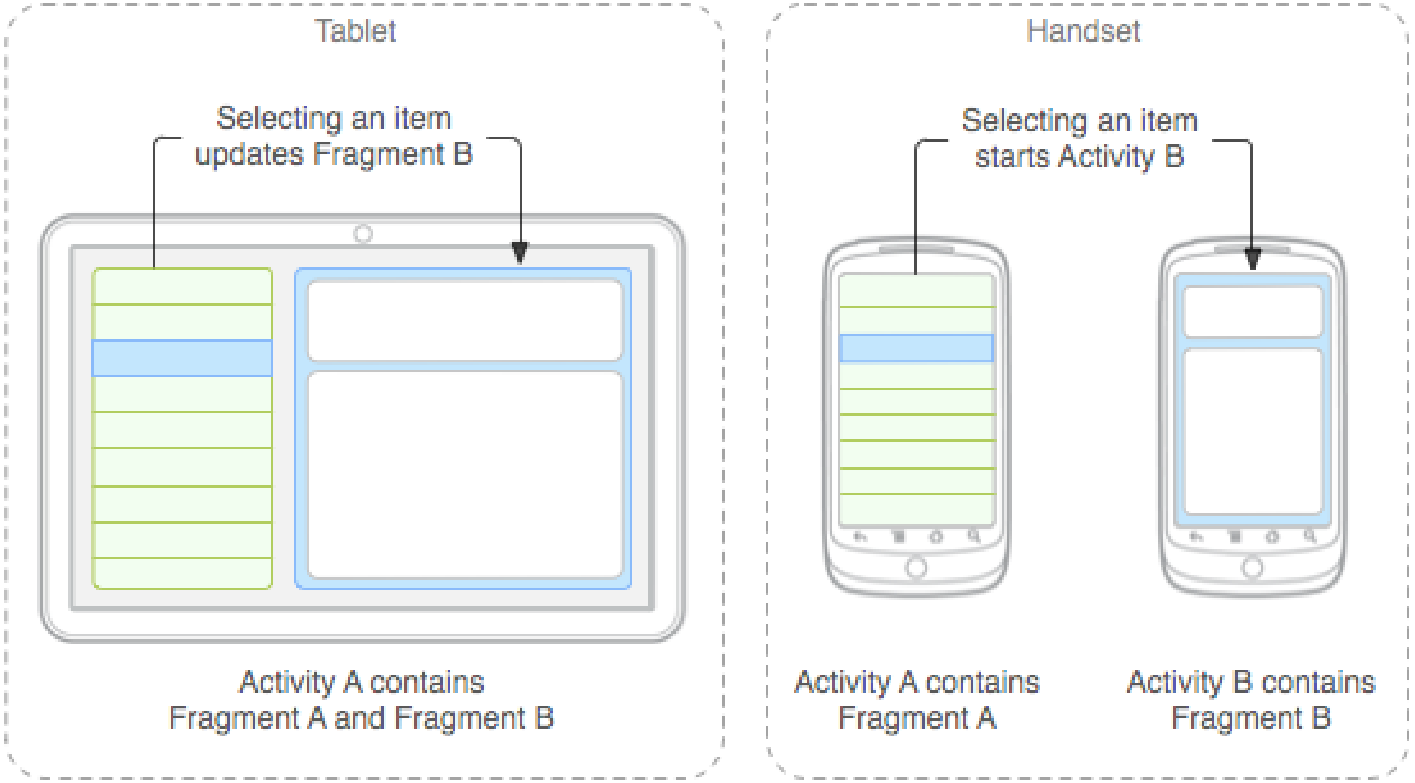
Kian
@naturelvr49
😍 Animal lover, writer, traveller
831 Following 1,807 Followers
Following
Tweets Tweets & replies Media Likes

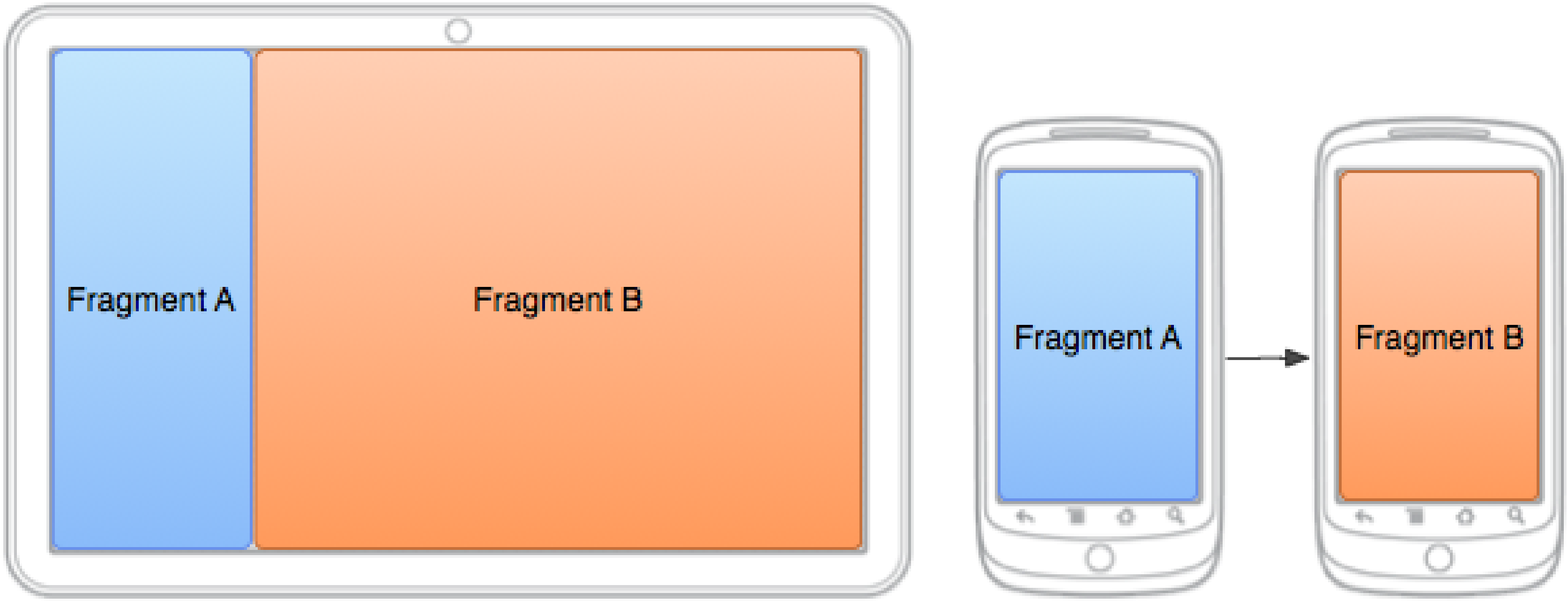
Kian @naturelvr49 · 1m
Feeling a breeze during this #heatwave. Bike ride during sunset 😍😊

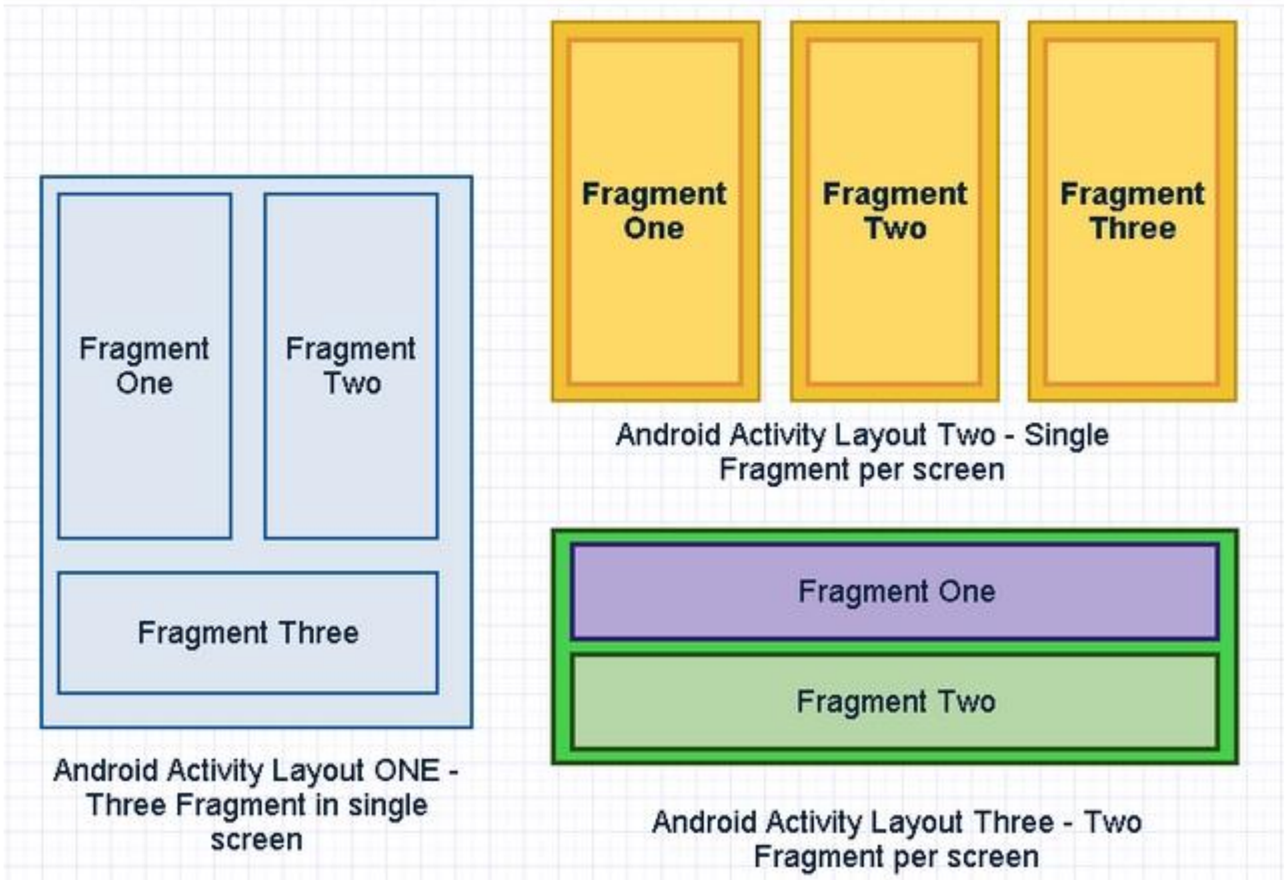
Messaging
Chat privately with friends, followers, businesses, and your government

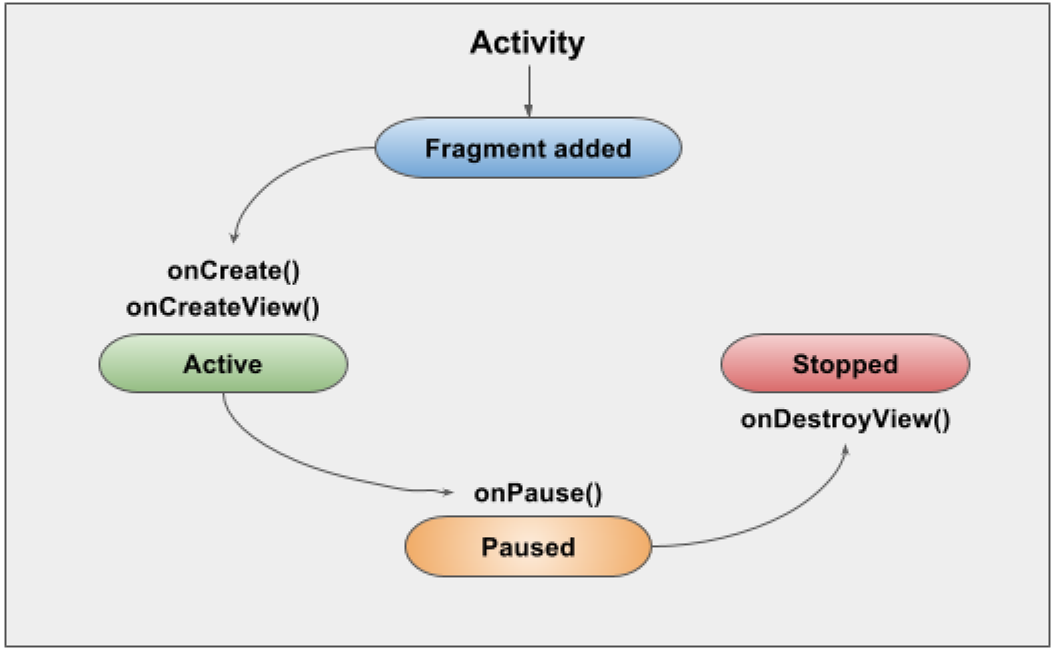
Kian
@naturelvr49

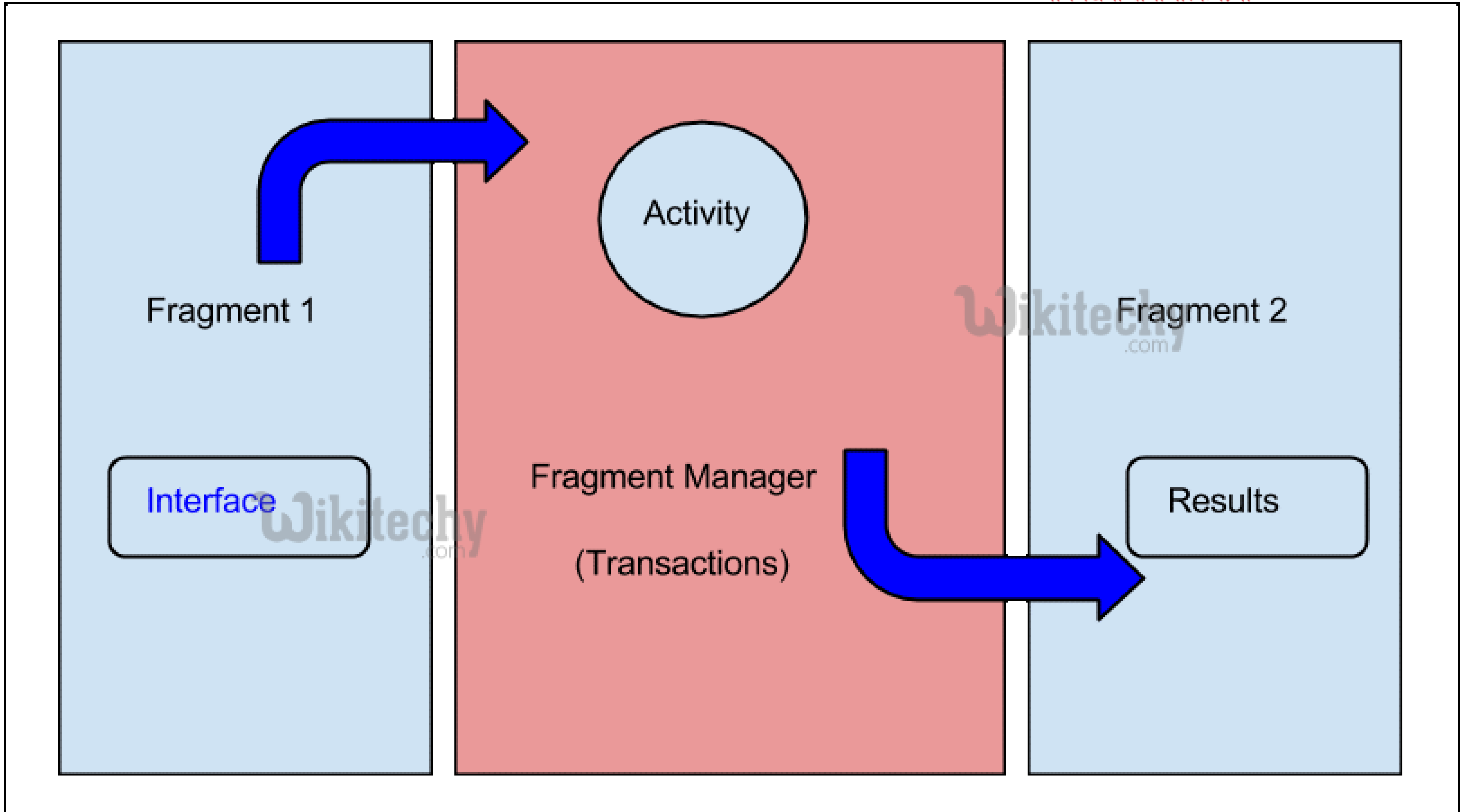
Hi! Plans this weekend?
Kian · 10:57 AM
Going to the beach!
Join me?
10:58 AM
What fun! I'll pack a snack and my sunglasses
Kian · 10:59 AM
Great, can't wait! You're the best :)
11:00 AM
Start a new message

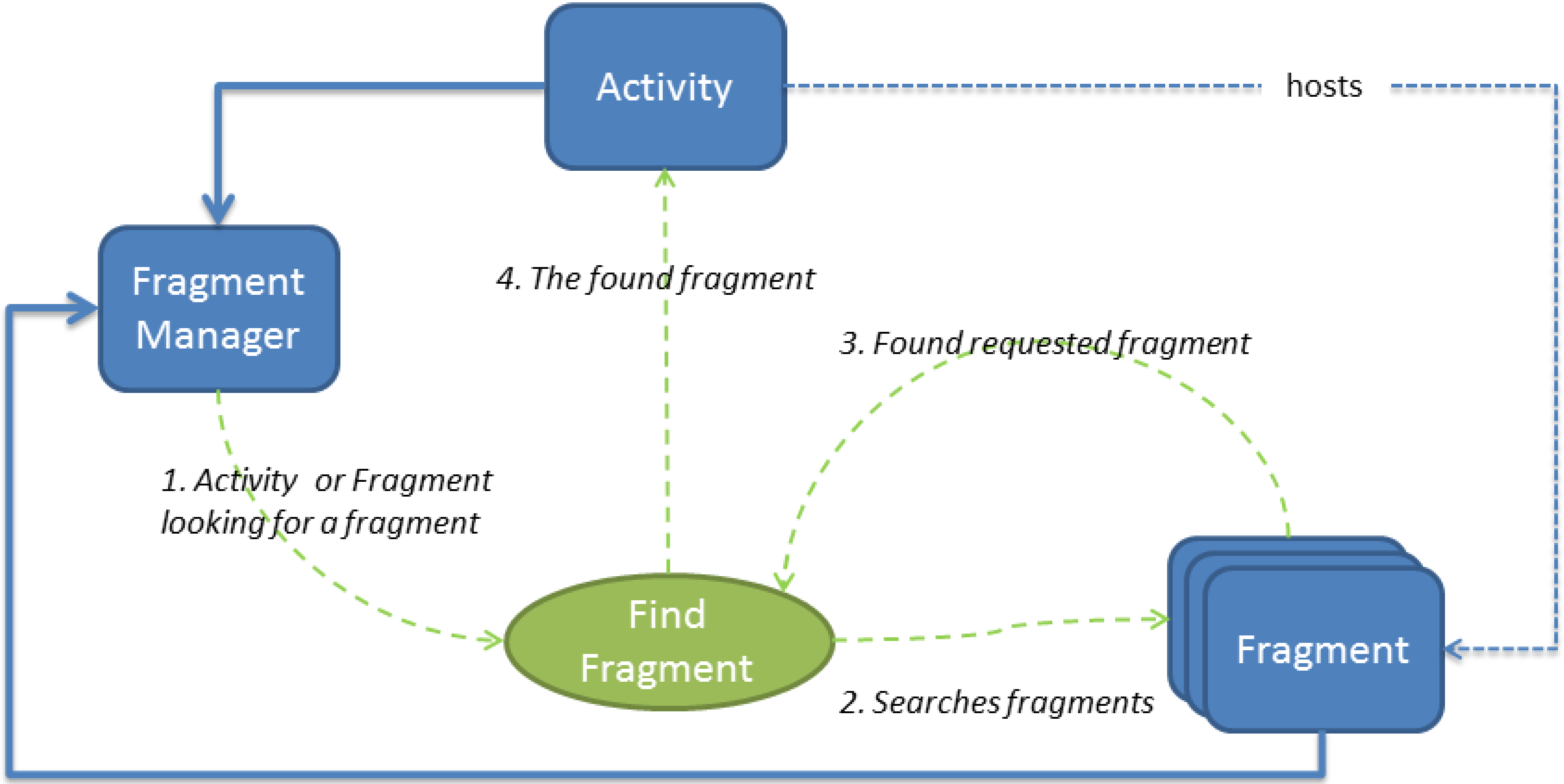

Tablet
Selecting an item updates Fragment B
Activity A contains Fragment A and Fragment B
Handset
Selecting an item starts Activity B
Activity A contains Fragment A
Activity B contains Fragment B

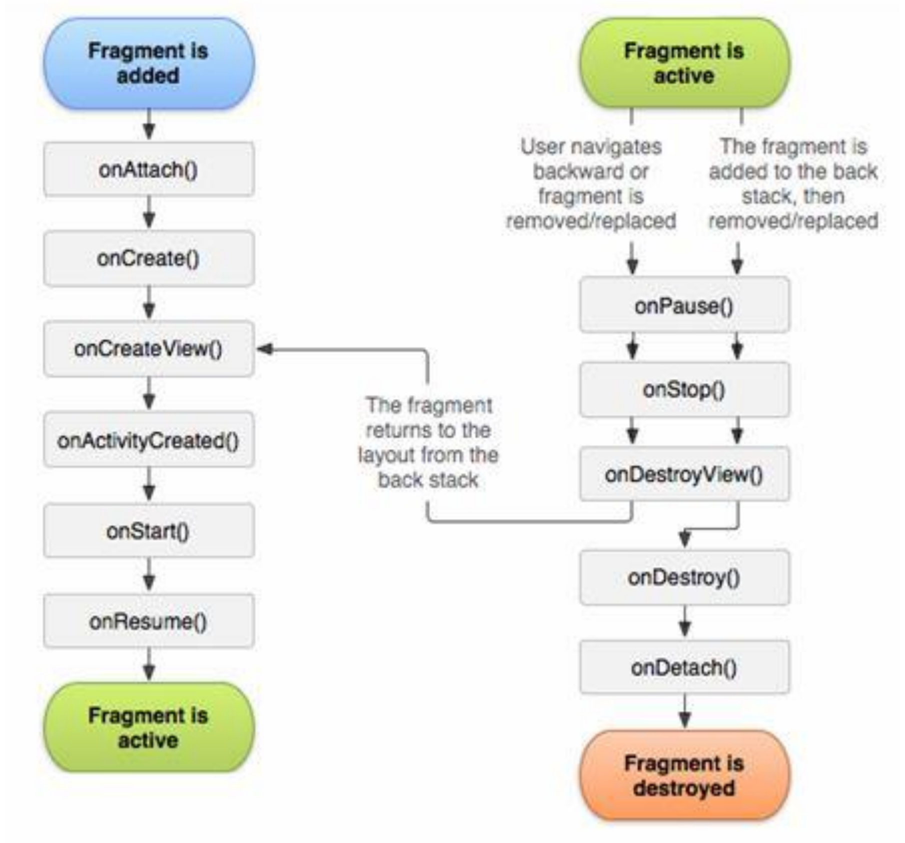

Fragment A
Fragment B
Fragment A
Fragment B


Fragment One
Fragment Two
Fragment Three
Android Activity Layout ONE - Three Fragment in single screen
Fragment One
Fragment Two
Fragment Three
Android Activity Layout Two - Single Fragment per screen
Fragment One
Fragment Two
Android Activity Layout Three - Two Fragment per screen

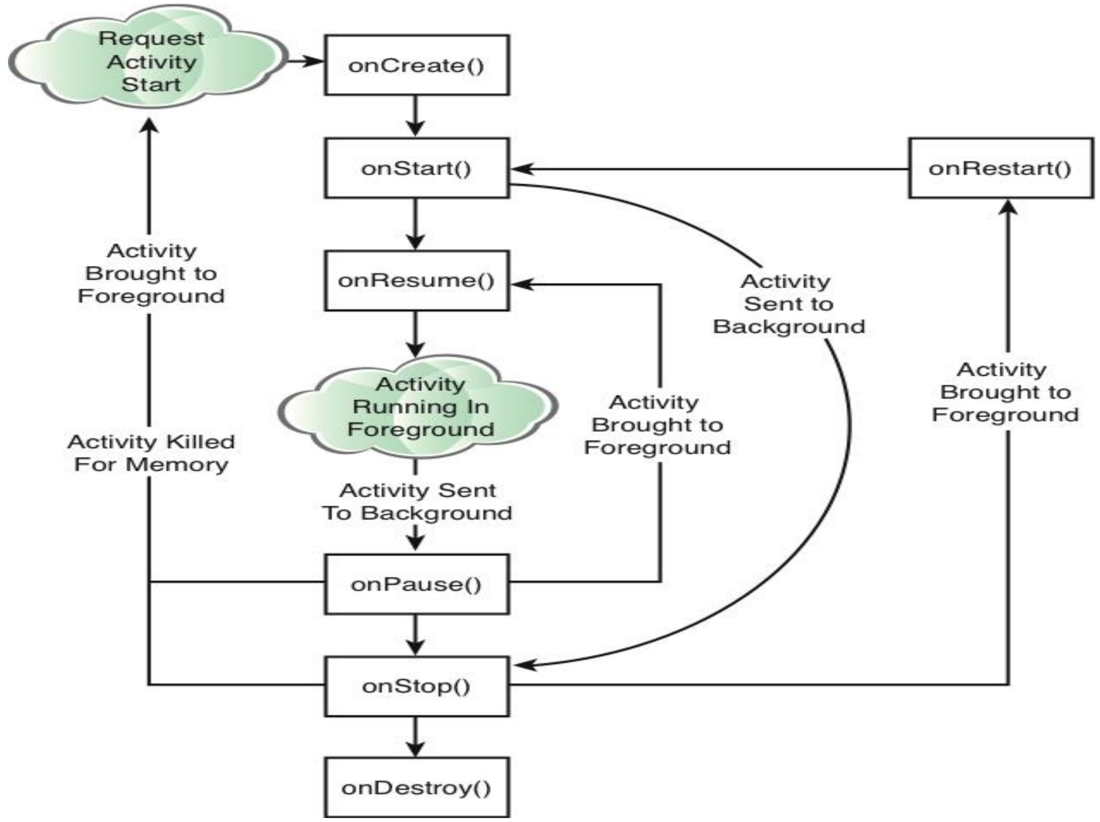

Activity
Fragment added
onCreate()
onCreateView()
Active
Stopped
onDestroyView()
onPause()
Paused


Activity
Fragment 1
Fragment 2
Interface
Fragment Manager
(Transactions)
Results


Activity
hosts
Fragment Manager
4. The found fragment
3. Found requested fragment
1. Activity or Fragment looking for a fragment
Find Fragment
Fragment
2. Searches fragments

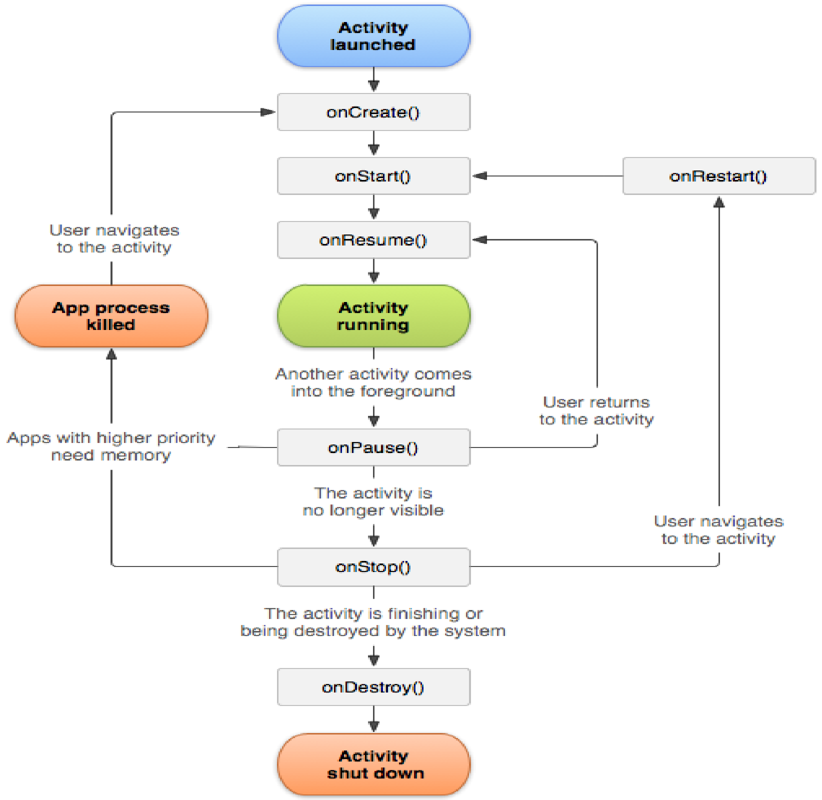

Fragment is added
onAttach()
onCreate()
onCreateView()
onActivityCreated()
onStart()
onResume()
Fragment is active
Fragment is active
User navigates backward or fragment is removed/replaced
The fragment is added to the back stack, then removed/replaced
onPause()
onStop()
onDestroyView()
The fragment returns to the layout from the back stack
onDestroy()
onDetach()
Fragment is destroyed

Life Cycle of an Activity:

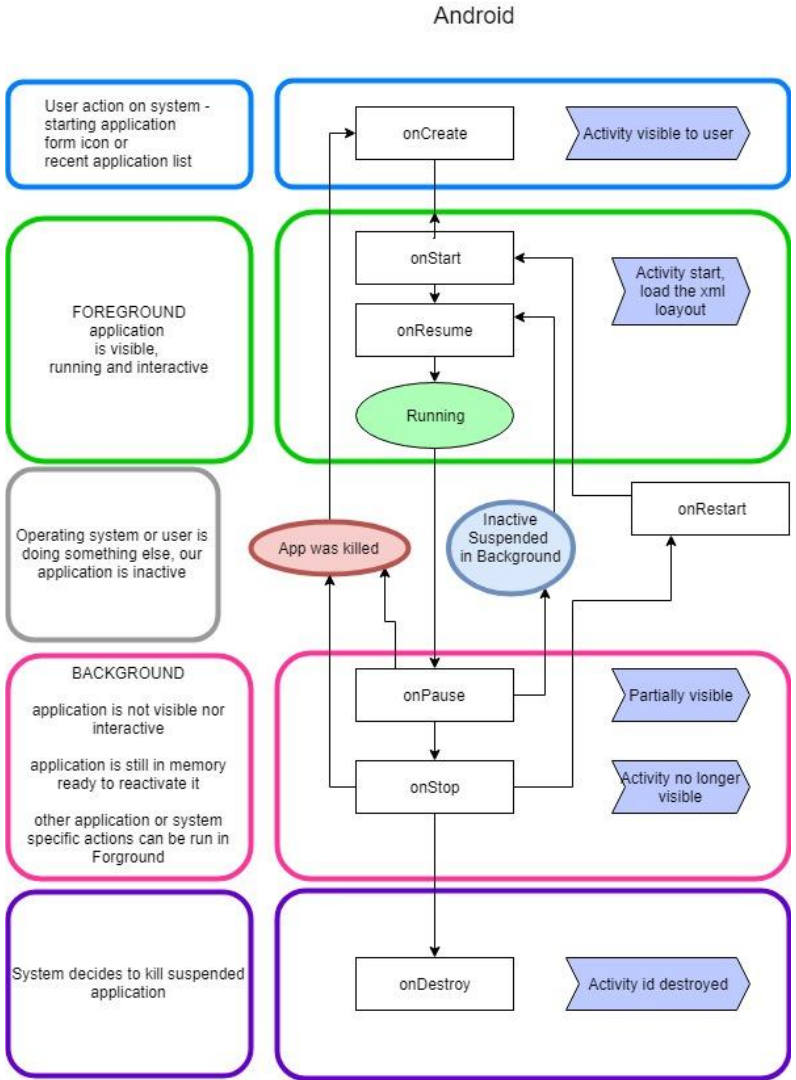

Request Activity Start
onCreate()
onStart()
onRestart()
onResume()
Activity Brought to Foreground
Activity Sent to Background
Activity Running In Foreground
Activity Brought to Foreground
Activity Brought to Foreground
Activity Killed For Memory
Activity Sent To Background
onPause()
onStop()
onDestroy()

Ganpat University ॥ विद्यया समाजोत्कर्षः ॥ | U.V. Patel College of Engineering

Life Cycle of an Activity:



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

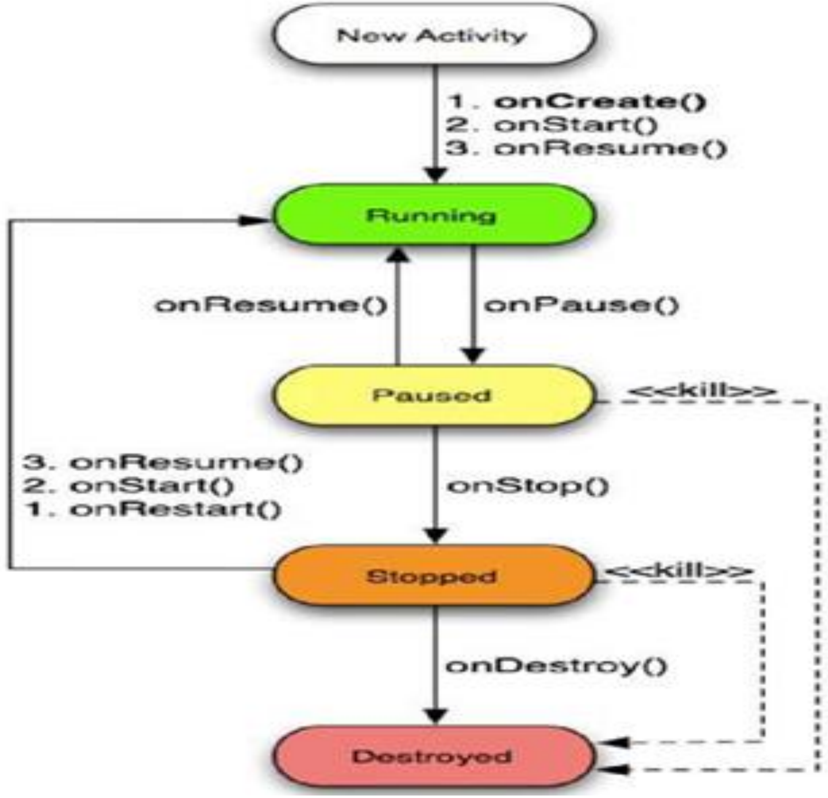

Android
User action on system - starting application form icon or recent application list
onCreate
Activity visible to user
FOREGROUND application is visible, running and interactive
onStart
Activity start, load the xml loayout
onResume
Running
Operating system or user is doing something else, our application is inactive
App was killed
Inactive Suspended in Background
onRestart
BACKGROUND
application is not visible nor interactive
application is still in memory ready to reactivate it
other application or system specific actions can be run in Forground
onPause
Partially visible
onStop
Activity no longer visible
System decides to kill suspended application
onDestroy
Activity id destroyed

Ganpat University | U.V. Patel College of Engineering
॥ विद्यया समाजोत्कर्षः ॥

Life Cycle States:

It has three states

1. *active / running*
2. *paused or*
3. *stopped*



Life Cycle States

- **Running**
  - It is *active or running when it is in the foreground of the screen(at the top of the activity stack for the current task).*
  - *This is the activity that is the focus for the user's actions.*

Life Cycle States

- **Paused**
  - It is *paused if it has lost focus but is still visible to the user.*
  - *That is, another activity lies on top of it and that new activity either is transparent or doesn't cover the full screen.*
  - *A paused activity is completely alive(it maintains all state and member information and remains attached to the window manager), but can be killed by the system in extreme low memory situations*

Life Cycle States

- **Stopped**
  - It is ***stopped*** *if it is completely obscured by another activity.*
  - *It still retains all state and member information. However, it is no longer visible to the user so its window is hidden and it will often be killed by the system when memory is needed elsewhere*

Life time

- **Life cycle of an activity:**
- **Visible life time:** onStart() to onStop()
    - During this life time the activity will not be interacting with the user. Between these two activities, resources are maintained that are needed by the system to show the activity on the screen

Life time

- **Foreground life time:** onResume() to onPause().
  - During this life time the activity will be in front of all other activities.
- **Entire Life time:** onCreate() to onDestroy()

Life time

- When you open the app it will go through below states:
- onCreate() –> onStart() –> onResume()
- When you press the back button and exit the app
- onPaused() — > onStop() –> onDestory()

Life time

- When you press the home button
- After pressing the home button, again when you open the app from a recent task list
- After dismissing the dialog or back button from the dialog
- ?

Life time

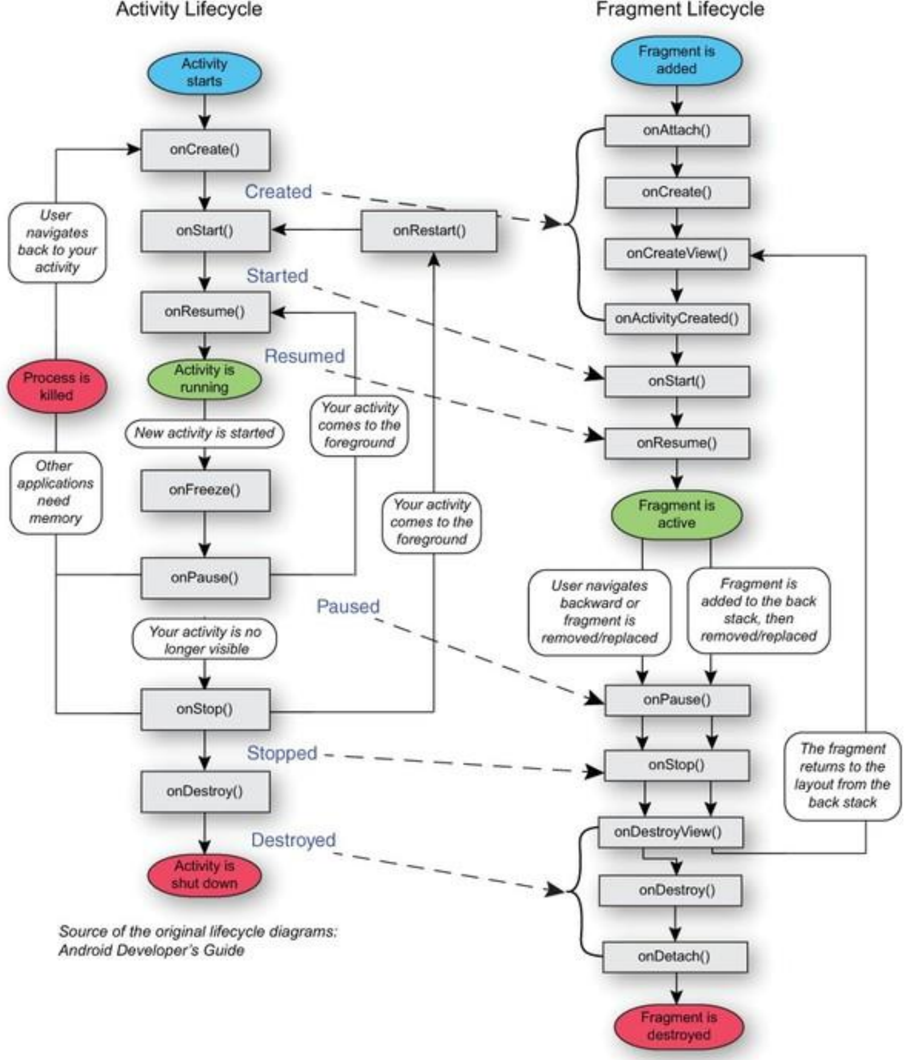
- **When you press the home button**
- onPaused() –> onStop()
- **After pressing the home button, again when you open the app from a recent task list**
- onRestart() –> onStart() –> onResume()
- **After dismissing the dialog or back button from the dialog**
- onResume()

Life time

- If a phone is ringing and user is using the app
- After the call ends
- When your phone screen is off
- When your phone screen is turned back on

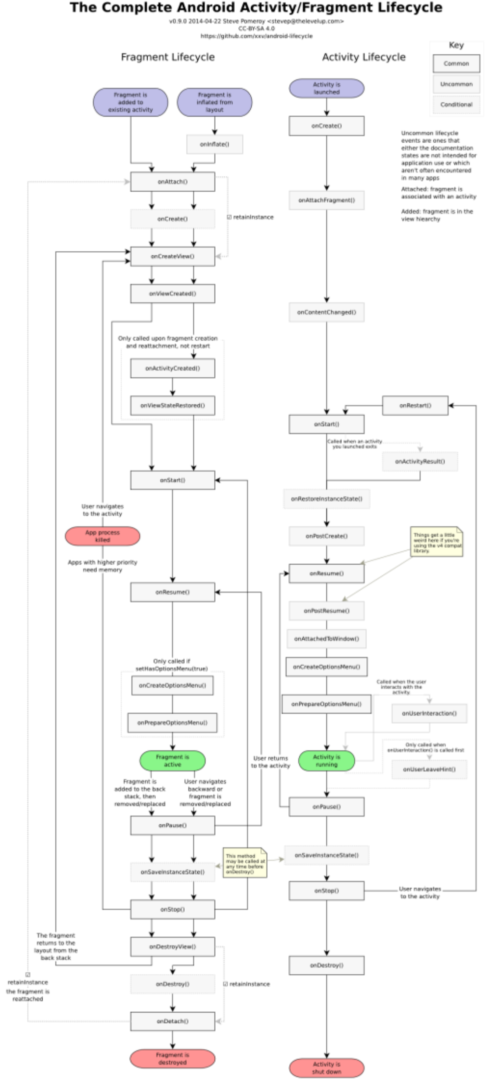
Life time

- **If a phone is ringing and user is using the app**
- onPause() –> onResume()
- **After the call ends**
- onResume()
- **When your phone screen is off**
- onPaused() –> onStop()
- **When your phone screen is turned back on**
- onRestart() –> onStart() –> onResume()



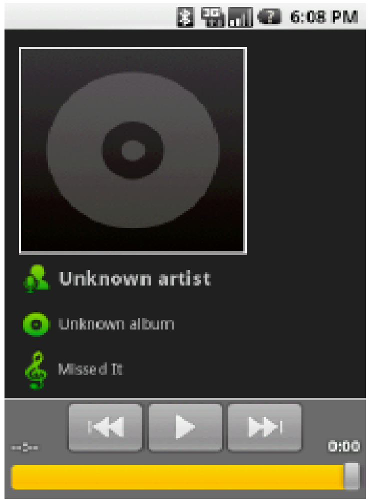
Source of the original lifecycle diagrams: Android Developer's Guide

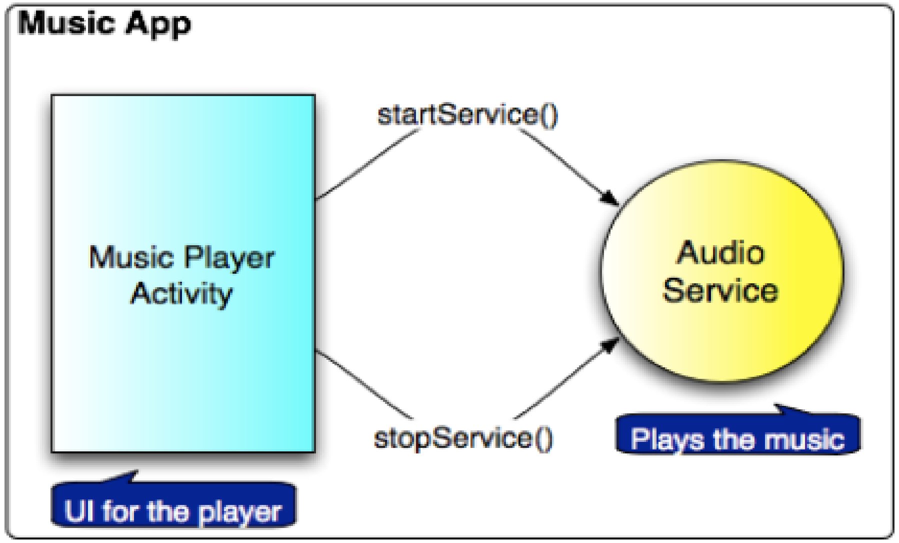
Ganpat University | U.V. Patel College of Engineering
॥ विद्यया समाजोत्कर्षः ॥


The Complete Android Activity/Fragment Lifecycle
v0.9.0 2014-04-22 Steve Pomeroy <stevep@thelevelup.com>
CC-BY-SA 4.0
https://github.com/xxv/android-lifecycle
Fragment Lifecycle
Activity Lifecycle
Key
Common
Uncommon
Conditional
Fragment is added to existing activity
Fragment is inflated from layout
Activity is launched
onInflate()
onCreate()
onAttach()
onAttachFragment()
Uncommon lifecycle events are ones that either the documentation states are not intended for application use or which aren't often encountered in many apps
Attached: fragment is associated with an activity
Added: fragment is in the view hiearchy
onCreate()
☑ retainInstance
onCreateView()
onViewCreated()
onContentChanged()
Only called upon fragment creation and reattachment, not restart
onActivityCreated()
onViewStateRestored()
onRestart()
onStart()
Called when an activity you launched exits
onActivityResult()
onStart()
onRestoreInstanceState()
User navigates to the activity
App process killed
Apps with higher priority need memory
onPostCreate()
Things get a little weird here if you're using the v4 compat library.
onResume()
onResume()
onPostResume()
onAttachedToWindow()
onCreateOptionsMenu()
Only called if setHasOptionsMenu(true)
onCreateOptionsMenu()
onPrepareOptionsMenu()
Called when the user interacts with the activity
onPrepareOptionsMenu()
onUserInteraction()
Only called when onUserInteraction() is called first
Fragment is active
User returns to the activity
Activity is running
onUserLeaveHint()
Fragment is added to the back stack, then removed/replaced
User navigates backward or fragment is removed/replaced
onPause()
onPause()
This method may be called at any time before onDestroy()
onSaveInstanceState()
onSaveInstanceState()
onStop()
User navigates to the activity
onStop()
The fragment returns to the layout from the back stack
onDestroyView()
onDestroy()
☐ retainInstance
the fragment is reattached
onDestroy()
☑ retainInstance
onDetach()
Fragment is destroyed
Activity is shut down

Services

Services are code that runs in the background. They can be started and stopped. Services doesn't have UI.



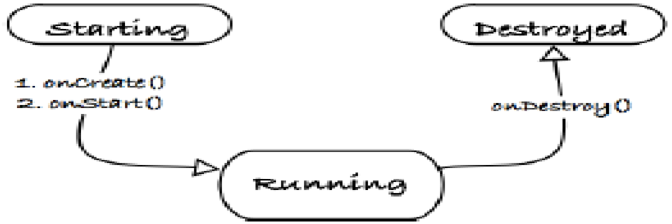


Service Life Cycle

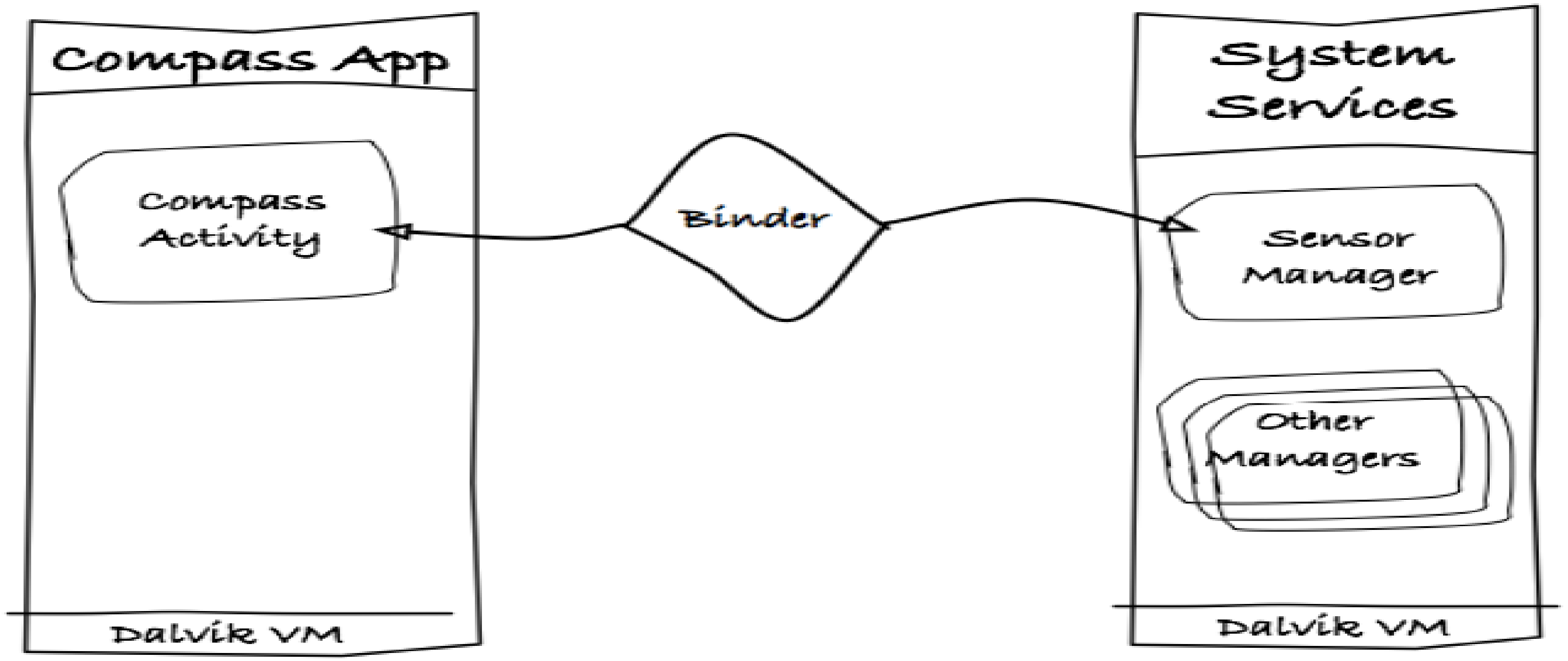
Service also has a lifecycle, but it's much simpler than activity's.

An activity typically starts and stops a service to do some work for it in the background, such as play music, check for new tweets, etc.

Services can be bound or unbound.

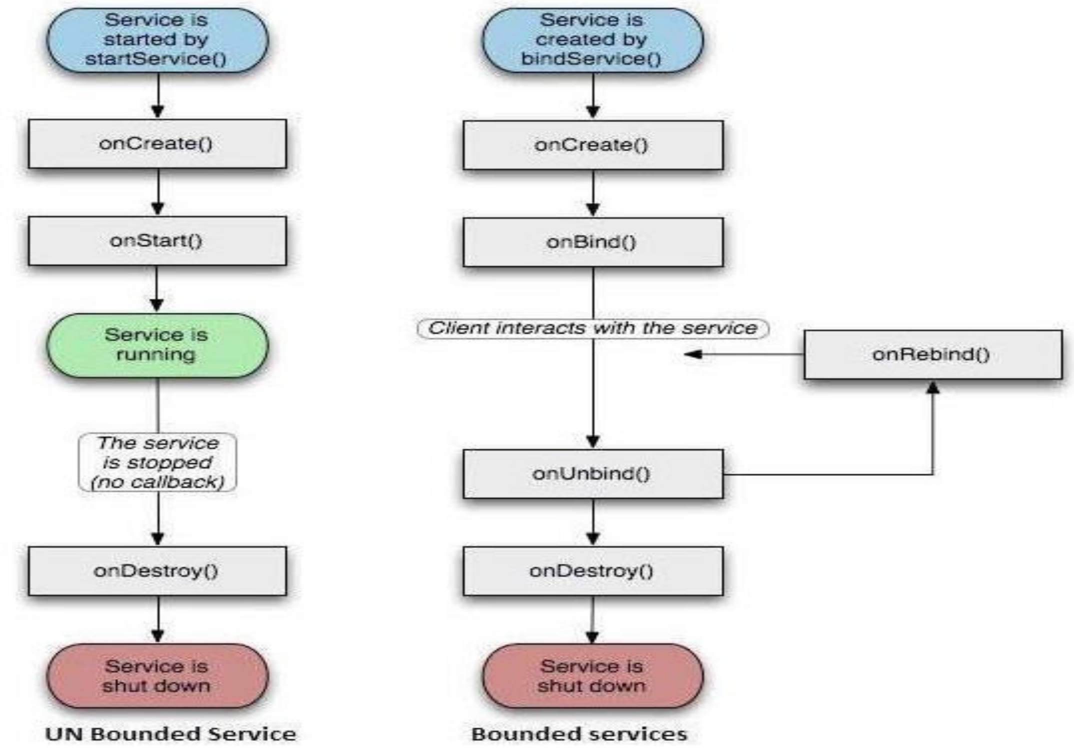


Remote Service



Ganpat University
॥ विद्यया समाजोत्कर्षः ॥
U.V. Patel College of Engineering

Service Life cycle



Service methods

- **onStartCommand()**

The system calls this method when another component, such as an activity, requests that the service be started, by calling ***startService().*** If you implement this method, it is your responsibility to stop the service when its work is done, by calling ***stopSelf()*** or ***stopService()*** methods.

Service methods

- **onBind()**

The system calls this method when another component wants to bind with the service by calling *bindService()*. If you implement this method, you must provide an interface that clients use to communicate with the service, by returning an *IBinder* object. You must always implement this method, but if you don't want to allow binding, then you should return *null*.

Service Life cycle

- **onUnbind()**

The system calls this method when all clients have disconnected from a particular interface published by the service.

- **onRebind()**

The system calls this method when new clients have connected to the service, after it had previously been notified that all had disconnected in its *onUnbind(Intent)*.

Service Life cycle

- **onCreate()**

The system calls this method when the service is first created using *onStartCommand()* or *onBind()*. This call is required to perform one-time set-up.

- **onDestroy()**

The system calls this method when the service is no longer used and is being destroyed. Your service should implement this to clean up any resources such as threads, registered listeners, receivers, etc.

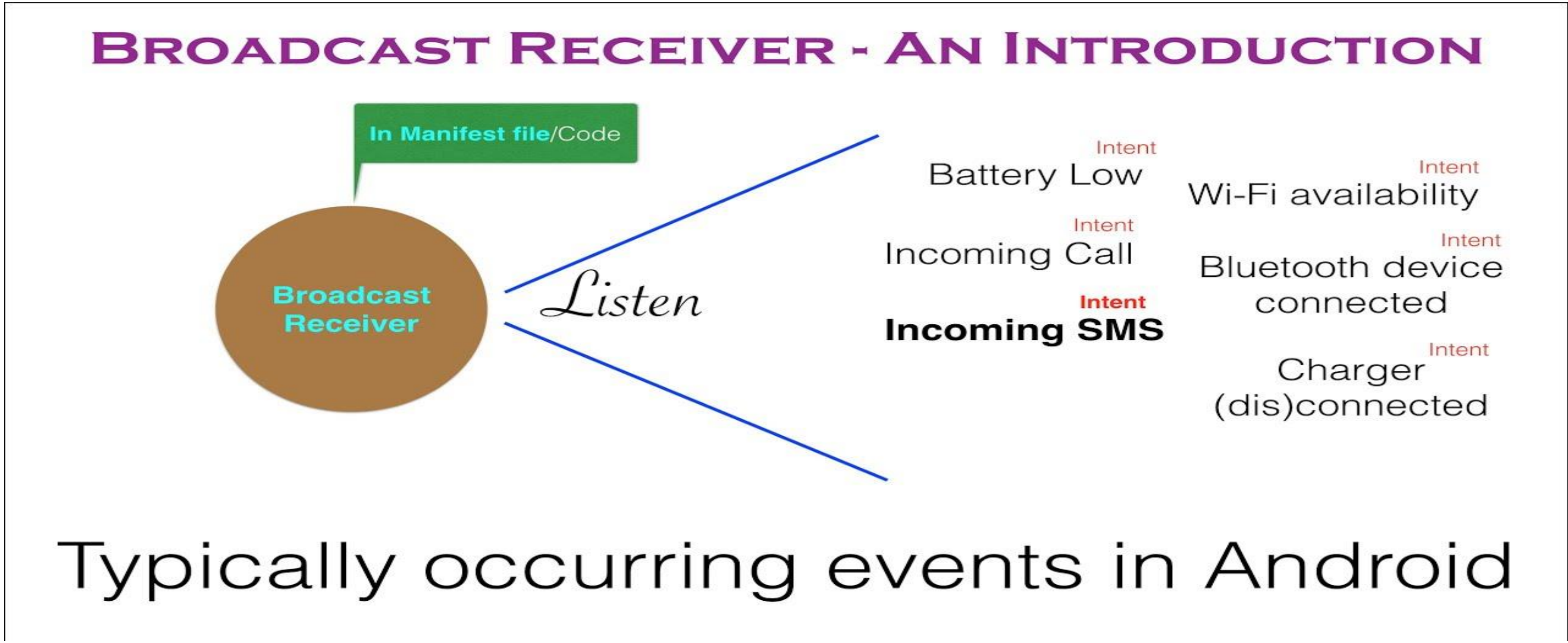
Broadcast Receiver

- Android BroadcastReceiver is a dormant component of android that listens to system-wide broadcast events or intents.

- When any of these events occur it brings the application into action by either creating a status bar notification or performing a task.

- Unlike activities, android BroadcastReceiver doesn’t contain any user interface. Broadcast receiver is generally implemented to delegate the tasks to services depending on the type of intent data that’s received.

Broadcast Receiver



Broadcast Receiver

- Following are some of the important system wide generated intents.

1. **android.intent.action.BATTERY_LOW** : Indicates low battery condition on the device.
2. **android.intent.action.BOOT_COMPLETED** : This is broadcast once, after the system has finished booting
3. **android.intent.action.CALL** : To perform a call to someone specified by the data
4. **android.intent.action.DATE_CHANGED** : The date has changed
5. **android.intent.action.REBOOT** : Have the device reboot
6. **android.net.conn.CONNECTIVITY_CHANGE** : The mobile network or wifi connection is changed(or reset)

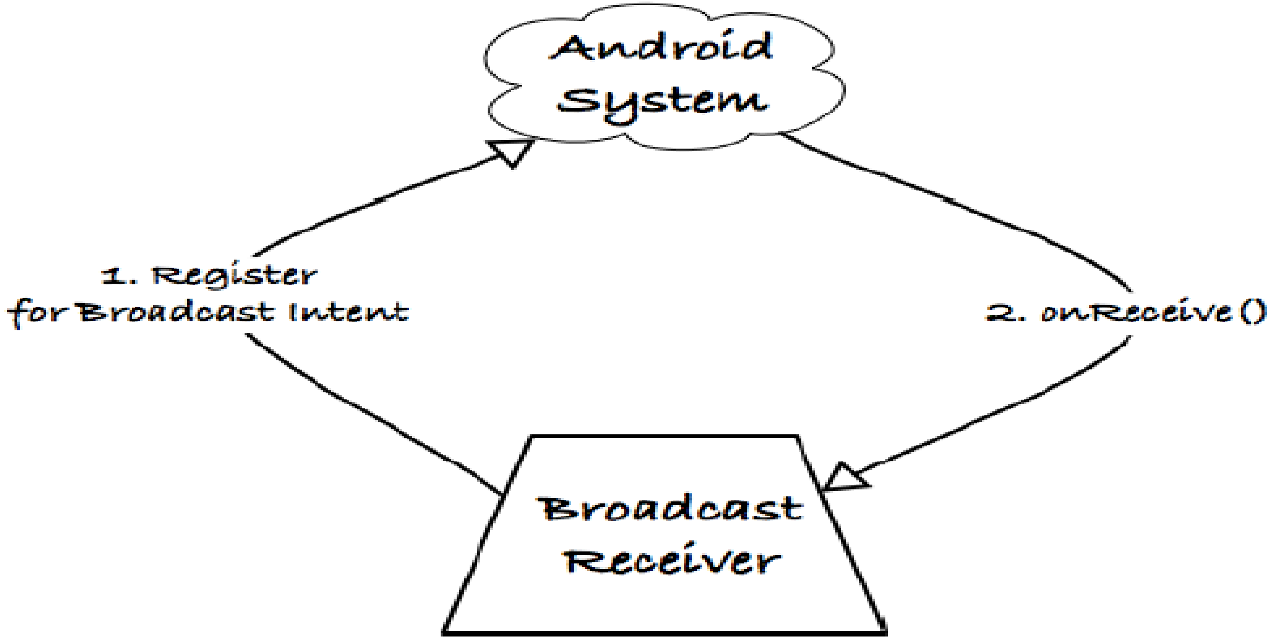
Broadcast Receiver in Android

- To set up a Broadcast Receiver in android application we need to do the following two things:

1. Creating a BroadcastReceiver
2. Registering a BroadcastReceiver

Broadcast Receiver

Ganpat University | U.V. Patel College of Engineering
॥ विद्यया समाजोत्कर्षः ॥



An Intent-based publish-subscribe mechanism. Great for listening system events such as SMS messages.

Broadcast Receiver

Creating the Broadcast Receiver

A broadcast receiver is implemented as a subclass of **BroadcastReceiver** class and overriding the onReceive() method where each message is received as a **Intent** object parameter.

```
import android.content.BroadcastReceiver
import android.content.Context
import android.content.Intent

class MyReceiver : BroadcastReceiver() {

    override fun onReceive(context: Context, intent: Intent) {
        // TODO: This method is called when the BroadcastReceiver is receiving
        // an Intent broadcast.
        throw UnsupportedOperationException("Not yet implemented")
    }
}
```

Broadcast Receiver

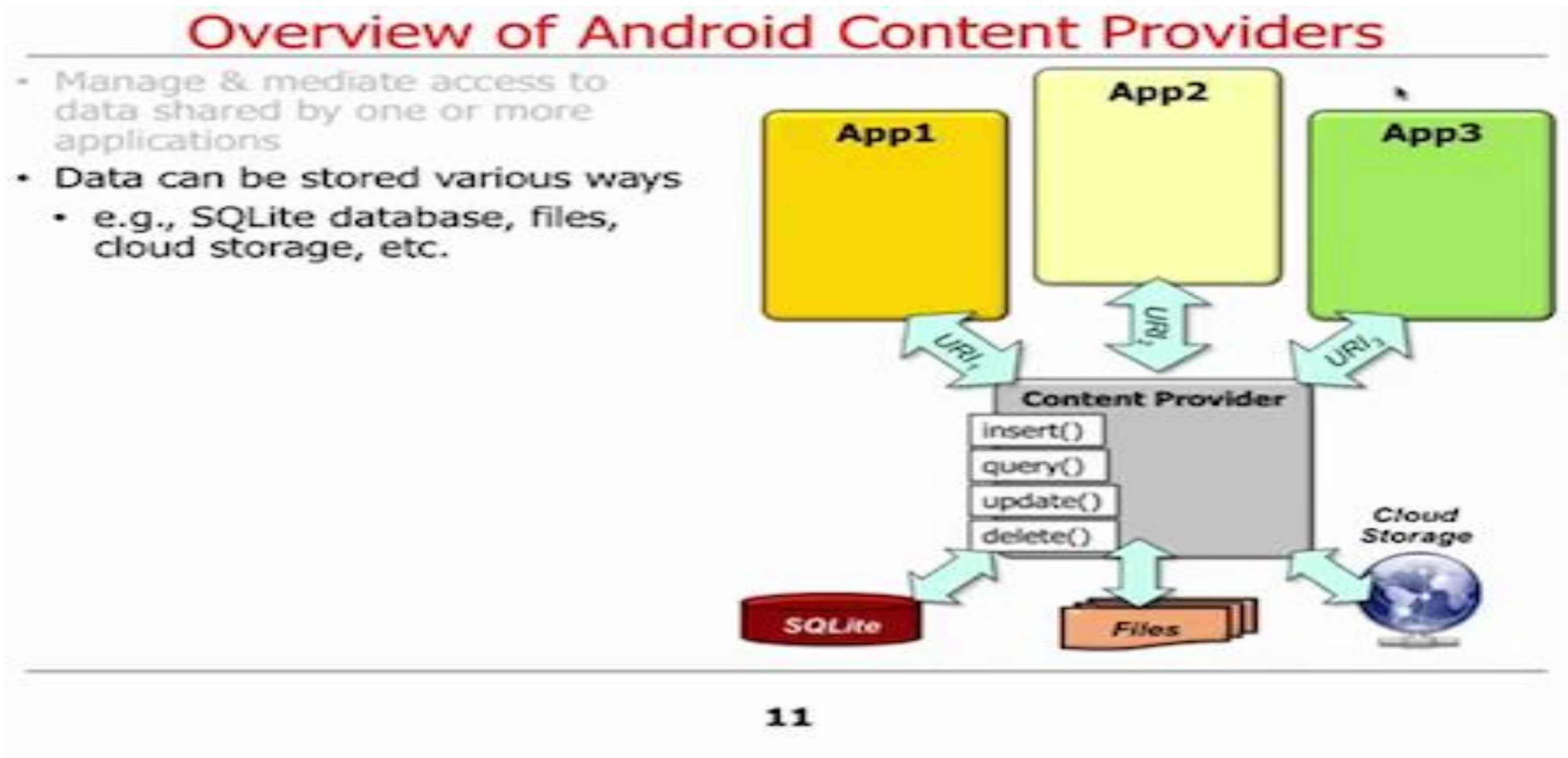
Registering Broadcast Receiver

An application listens for specific broadcast intents by registering a broadcast receiver in *AndroidManifest.xml* file. Consider we are going to register *MyReceiver* for system generated event ACTION_BOOT_COMPLETED which is fired by the system once the Android system has completed the boot process.

```
<application android:icon="@drawable/ic_launcher" android:label="@string/app_name"
android:theme="@style/AppTheme" >
    <receiver android:name="MyReceiver">
        <intent-filter>
            <action android:name="android.intent.action.BOOT_COMPLETED">
        </action>
        </intent-filter>
    </receiver>
</application>
```

Ganpat University
॥ विद्यया समाजोत्कर्षः ॥
U.V. Patel College of Engineering

Content Provider



Overview of Android Content Providers

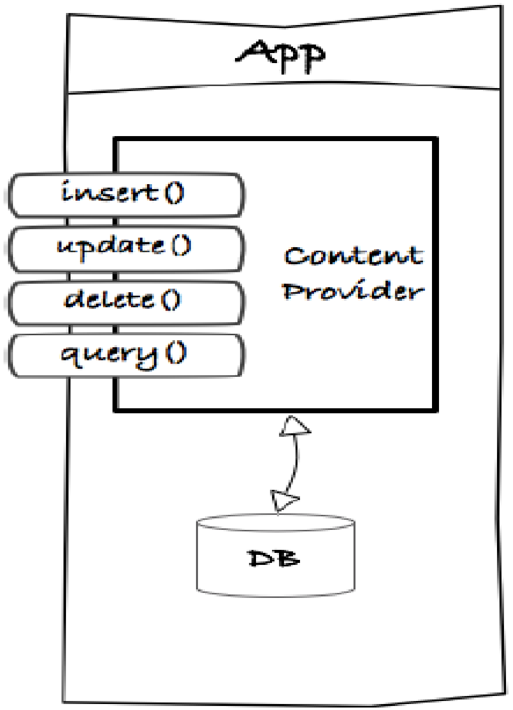
- Manage & mediate access to data shared by one or more applications
- Data can be stored various ways
  - e.g., SQLite database, files, cloud storage, etc.

11

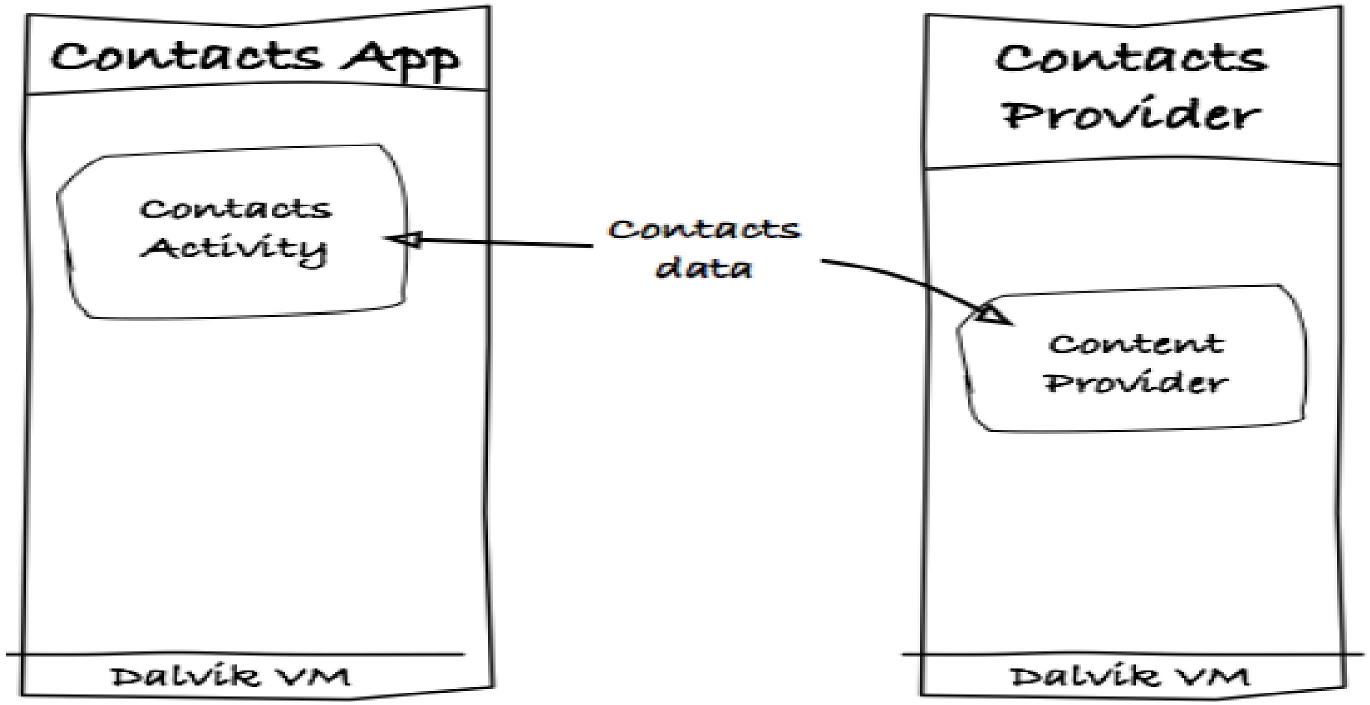
Ganpat University || विद्यया समाजोत्कर्षः || U.V. Patel College of Engineering

Content Provider

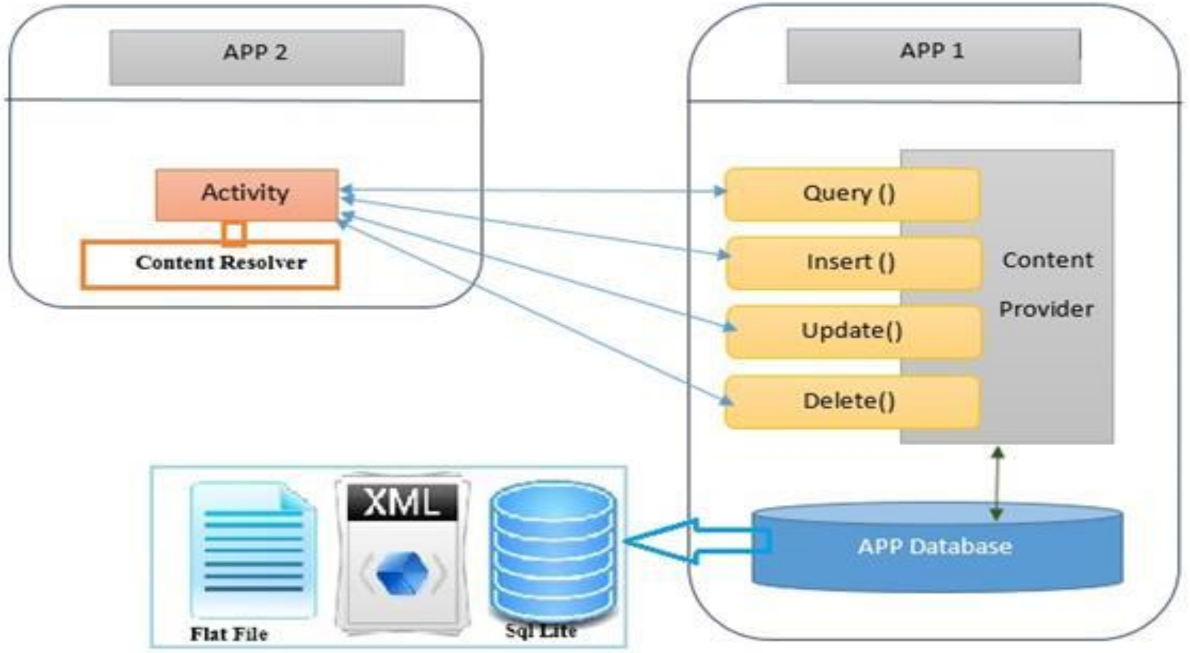
Content Providers share content with applications across application boundaries.
Examples of built-in Content Providers are: Contacts, MediaStore, Settings and more.



Content Provider



Content Provider


APP 2
Activity
Content Resolver
APP 1
Query ()
Insert ()
Update()
Delete()
Content Provider
APP Database
XML
Flat File
Sql Lite