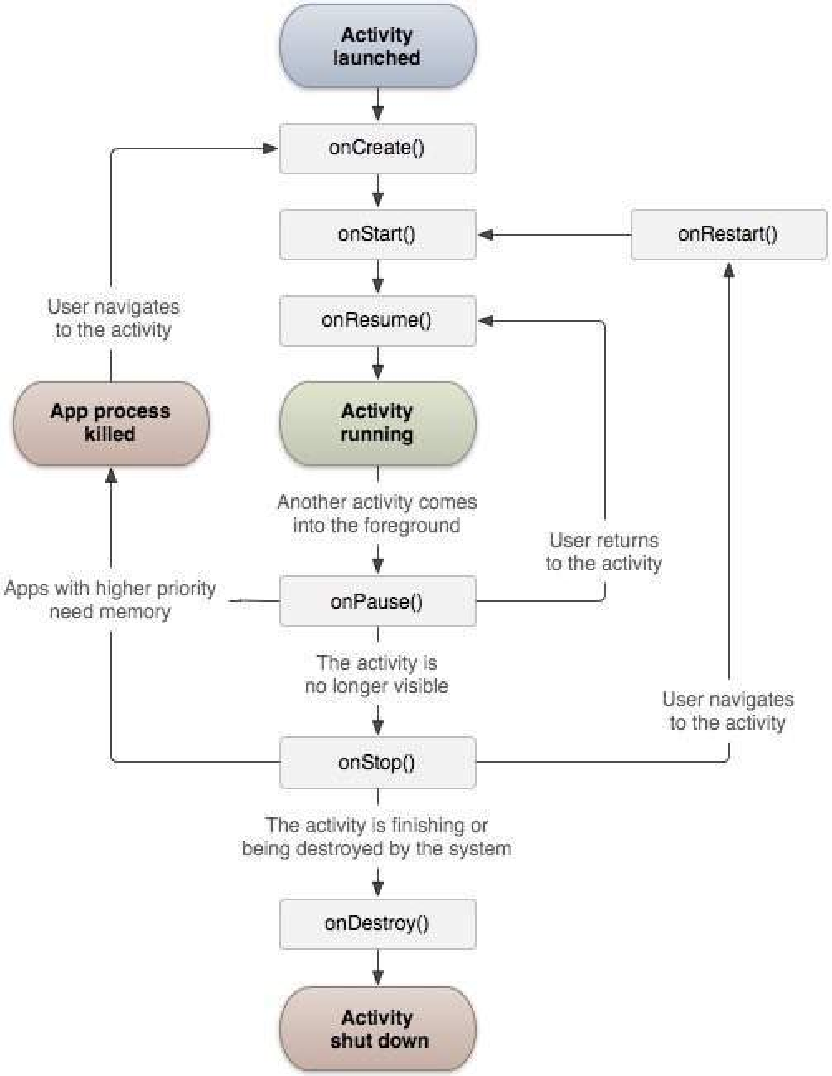
Android Activities and Services

Tushar B. Kute,
http://tusharkute.com

Activity

- If you have worked with C, C++ or Java programming language then you must have seen that your program starts from main() function.
- Very similar way, Android system initiates its program with in an Activity starting with a call on onCreate() callback method.
- There is a sequence of callback methods that start up an activity and a sequence of callback methods that tear down an activity as shown in the below Activity life cycle diagram.

Activity Life Cycle


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

Activity Methods

- onCreate()
  - This is the first callback and called when the activity is first created.
- onStart()
  - This callback is called when the activity becomes visible to the user.
- onResume()
  - This is called when the user starts interacting with the application.

Activity Methods

- onPause()
  - The paused activity does not receive user input and cannot execute any code and called when the current activity is being paused and the previous activity is being resumed.
- onStop()
  - This callback is called when the activity is no longer visible.
- onDestroy()
  - This callback is called before the activity is destroyed by the system.
- onRestart()
  - This callback is called when the activity restarts after stopping it.

Activity Example:

- onPau

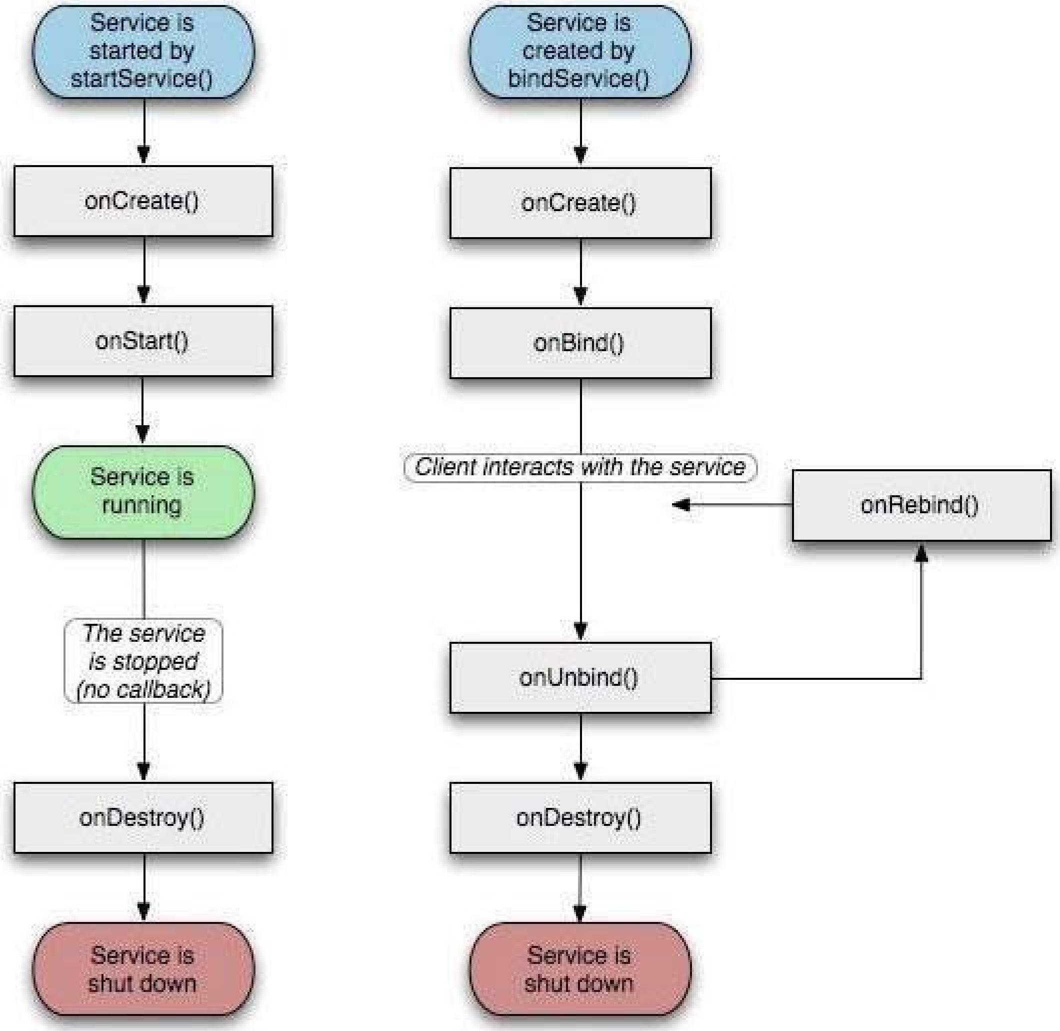
Service

- A service is a component that runs in the background to perform long-running operations without needing to interact with the user and it works even if application is destroyed.
- A service can essentially take two states –
  - Started
  - Bound

Service States

- Started
  - A service is started when an application component, such as an activity, starts it by calling startService(). Once started, a service can run in the background indefinitely, even if the component that started it is destroyed.
- Bound
  - A service is bound when an application component binds to it by calling bindService(). A bound service offers a client-server interface that allows components to interact with the service, send requests, get results, and even do so across processes with interprocess communication (IPC).

Service States


Service is started by startService()
onCreate()
onStart()
Service is running
The service is stopped (no callback)
onDestroy()
Service is shut down
Service is created by bindService()
onCreate()
onBind()
Client interacts with the service
onRebind()
onUnbind()
onDestroy()
Service is shut down

UN Bounded Service

Bounded services

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.
- 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 Methods

- 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).
- 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.

Service: Example

- Example.

Thank you

This presentation is created using LibreOffice Impress 4.2.8.2, can be used freely as per GNU General Public License

Web Resources
http://mitu.co.in
http://tusharkute.com

Blogs
http://digitallocha.blogspot.in
http://kyamputar.blogspot.in

tushar@tusharkute.com