

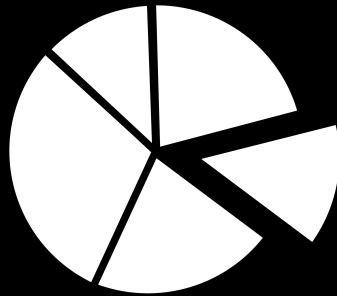


The Secret Life of Services

Maksim Lin
Freelance Android Developer



www.manichord.com



Poppy App

calculating digits of π

Compute

```
public class PiCalculator {  
    /**  
     * Calculate digits of Pi  
     */  
    public static void compute() {  
        // calc tight-loop ...  
    }  
}
```

Why Services?

Because...

*“A Service is an application component that can perform **long-running** operations in the **background** and does **not** provide a user interface. Another application component can start a service and it will continue to run in the background even if the **user switches to another application**.”*

- developer.android.com

Staying Alive...

*Because a process running a service is **ranked higher** than a process with **background activities**, an activity that initiates a long-running operation might do well to start a service for that operation, rather than simply create a **worker thread**—particularly if the operation will likely **outlast the activity**.*

- developer.android.com

Android is a Framework

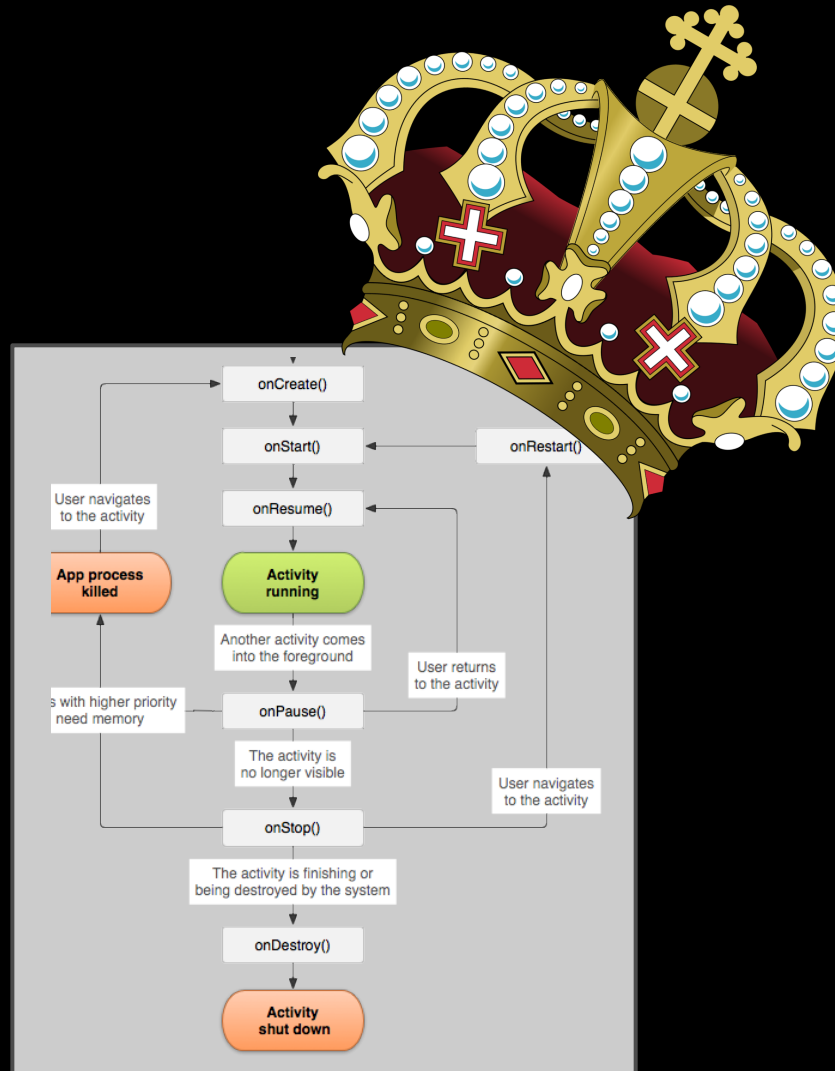
Frameworks ?



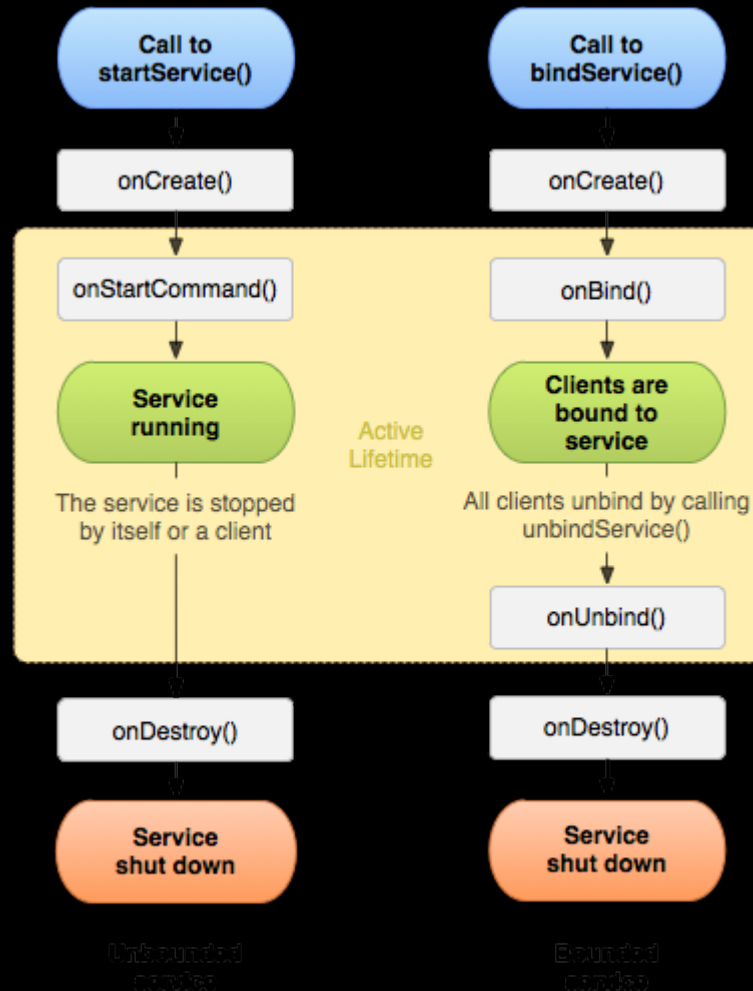
CALLBACKS! CALLBACKS! CALLBACKS!



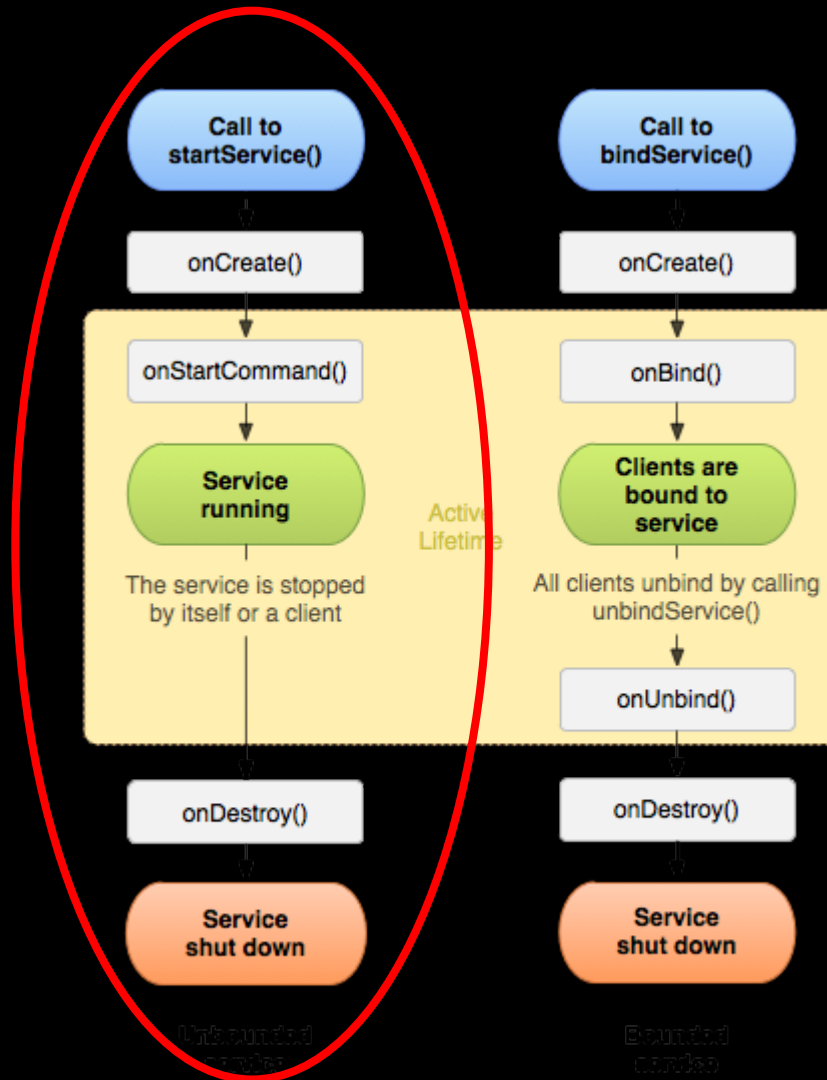
Android Framework: The Lifecycle Rules



Service Lifecycle



Service Lifecycle



```
Intent intent = new Intent(this, PoppyStandardService.class);  
startService(intent);
```

```
@Override  
public int onStartCommand(Intent intent, int flags, int startId) {  
    // do our stuff...  
    PiCalculator.compute();  
  
    return super.onStartCommand(intent, flags, startId);  
}
```

Don't forget the Manifest

```
<service  
    android:name=".PoppyStandardService"  
    android:enabled="true"  
    android:exported="false" >  
</service>
```

Poppy App isn't responding. Do you
want to close it?

WAIT

OK

Poppy is being sloppy!



Services Are...

- ★ **NOT** a Process/Thread
- ★ *A Lifecycle container*

So instead...

```
public int onStartCommand(Intent intent, int flags, int  
startId) {  
    new Thread(new Runnable() {  
        @Override  
        public void run() {  
            // do our stuff...in a *NEW* thread  
            PiCalculator.compute();  
        }  
    }).start();  
    return super.onStartCommand(intent, flags, startId);  
}
```



IntentService is ...

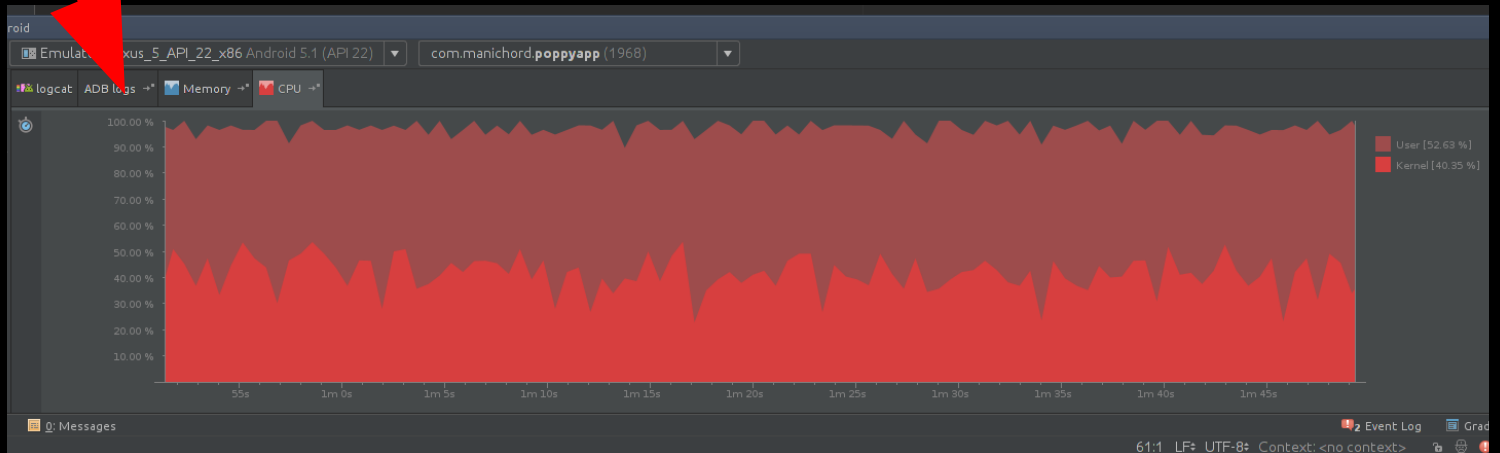
- ★ *badly* named
- ★ Very useful and easy to use
- ★ Callback on **background** thread
- ★ Only **ONE** background thread: serialised work
- ★ Queues intents
- ★ Stops itself when no more work

Easy....

```
public class PoppyIntentService extends IntentService {  
    ...  
    @Override  
    protected void onHandleIntent(Intent intent) {  
        // do our stuff...on a thread prepared for us earlier...  
        PiCalculator.compute();  
    }  
}
```

But

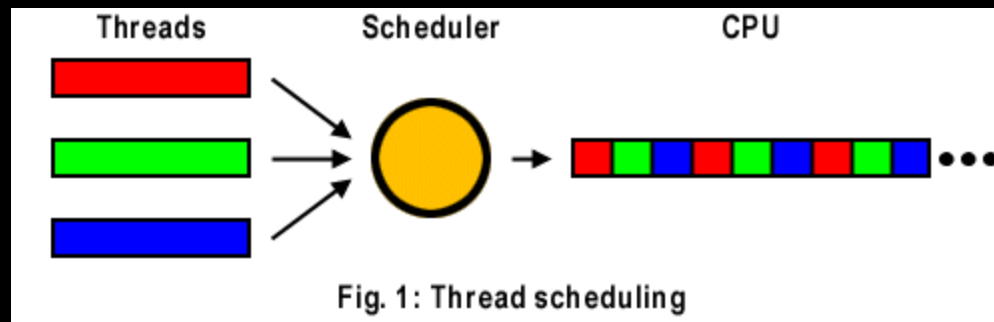
100% !!!!



But



Why ??





Thread Priority

What is the IntentServices thread priority ?

Lets look at the src...

```
public void onCreate() {
```

```
...
```

```
HandlerThread thread = new HandlerThread("IntentService[" + mName + "]");
```

Hmmm...

Thread Priority - Cont.d

```
public class HandlerThread extends Thread {
```

```
...
```

```
    public HandlerThread(String name) {
```

```
        super(name);
```

```
        mPriority = Process.THREAD_PRIORITY_DEFAULT;
```

```
    }
```

Thread Priority - Cont.d

public static final int `THREAD_PRIORITY_DEFAULT`

Added in API level 1

Standard priority of application threads.

public static final int `THREAD_PRIORITY_BACKGROUND`

Added in API level 1

Standard priority background threads. This gives your thread a slightly lower than normal priority, so that it will have less chance of impacting the responsiveness of the user interface.

public static final int `THREAD_PRIORITY_LOWEST`

Added in API level 1

Lowest available thread priority. Only for those who really, really don't want to run if anything else is happening.



- ★ Bug filed and patch submitted but *Abandoned* in Gerrit
- ★ DIY:

```
@Override
```

```
protected void onHandleIntent(Intent intent) {
```

```
    Process.setThreadPriority(
```

```
        Process.THREAD_PRIORITY_BACKGROUND);
```

```
    // do our stuff...in a thread prepared for us already
```

```
    PiCalculator.compute(1000);
```

```
}
```



from d.a.com example code:

```
public void onCreate() {  
    // Start up the thread running the service.  
    // Note that we create a separate thread because the service  
    // normally runs in the process's main thread, which we don't  
    // want to block. We also make it background priority so  
    // CPU-intensive work will not disrupt our UI.  
    HandlerThread thread = new HandlerThread(  
        "ServiceStartArguments", Process.  
        THREAD_PRIORITY_BACKGROUND);  
    thread.start();  
    ...  
}
```

Foreground Services

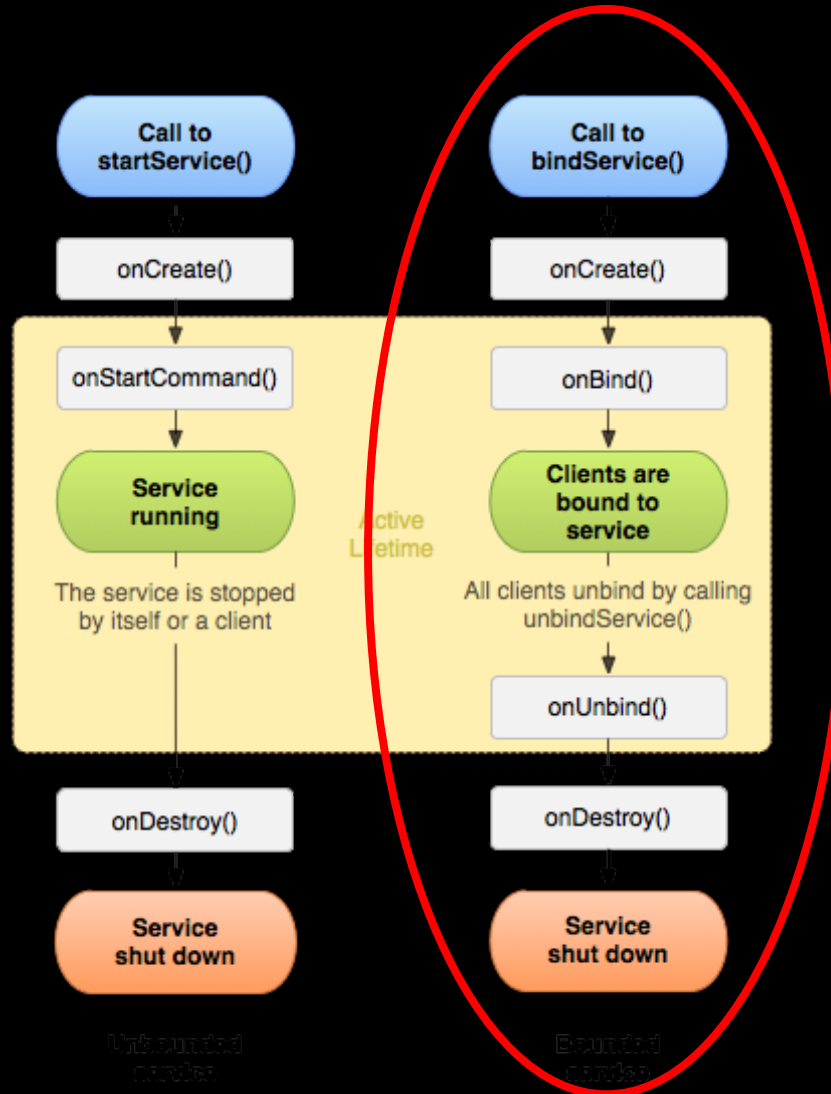
- ★ *Actively* being used by User
(eg. Music Player, Pedometer, etc)
- ★ (almost) Never be killed
- ★ **MUST** display a On-Going Notification

Poppy: comms back to UI

- ★ Direct: Notifications / Toasts
- ★ Broadcasts (System, Local or Pending)
- ★ Binding

Binding ?


More Callbacks!



Binding...

- ★ Clients (Activities) (un)-bind to Services
- ★ Binding is Async (more callbacks!)
- ★ 2-way RPC
- ★ Bound Service auto-shutdown if not “*started*”
- ★ Can be used with or **without** starting Service
- ★ Learn about IBinder...

What did we learn?

- ★ Services are a Lifecycle container
- ★ Avoid doing your own Thread Management
- ★ Use IntentService
- ★ Look at (Android) source when in doubt
- ★ Read the Android docs 

Thank You!

References

Android Threading:

<http://www.androiddesignpatterns.com/2014/01/thread-scheduling-in-android.html>

<http://stackoverflow.com/questions/8955458/asynctask-must-it-take-such-a-performance-penalty-hit>

IntentService Thread Priority:

<https://code.google.com/p/android/issues/detail?id=35440>

Image Credits

“square wheel trike” CC <https://www.flickr.com/photos/25831992@N03/2724551187>

“111 Hollywood Sign” www.flickr.com/photos/7294653@N07/ (CC BY-NC 2.0)

“Activity Lifecycle” <http://developer.android.com/guide/components/activities.html> Creative Commons Attribution 2.5

“Service Lifecycle” <http://developer.android.com/guide/components/services.html> Creative Commons Attribution 2.5

Seinfeld “The Pie” episode

“thread scheduling diagram” <http://www.gameprogrammer.net/delphi3dArchive/multithreading.htm>

“Android Eating Key Lime Pie” Manu Cornet (CC BY-NC-ND 3.0) <http://www.bonkersworld.net>

“Android Emoji” Android Emoji fonts Copyright © 2008 The Android Open Source Project. Licensed under the Apache License http://www.heyfr.fun/emoji/android/en/emoji_android_large_list.html

Questions?

<http://www.manichord.com>

github.com/maks

@mklin

<https://plus.google.com/+MaksimLin>

