SERVICES

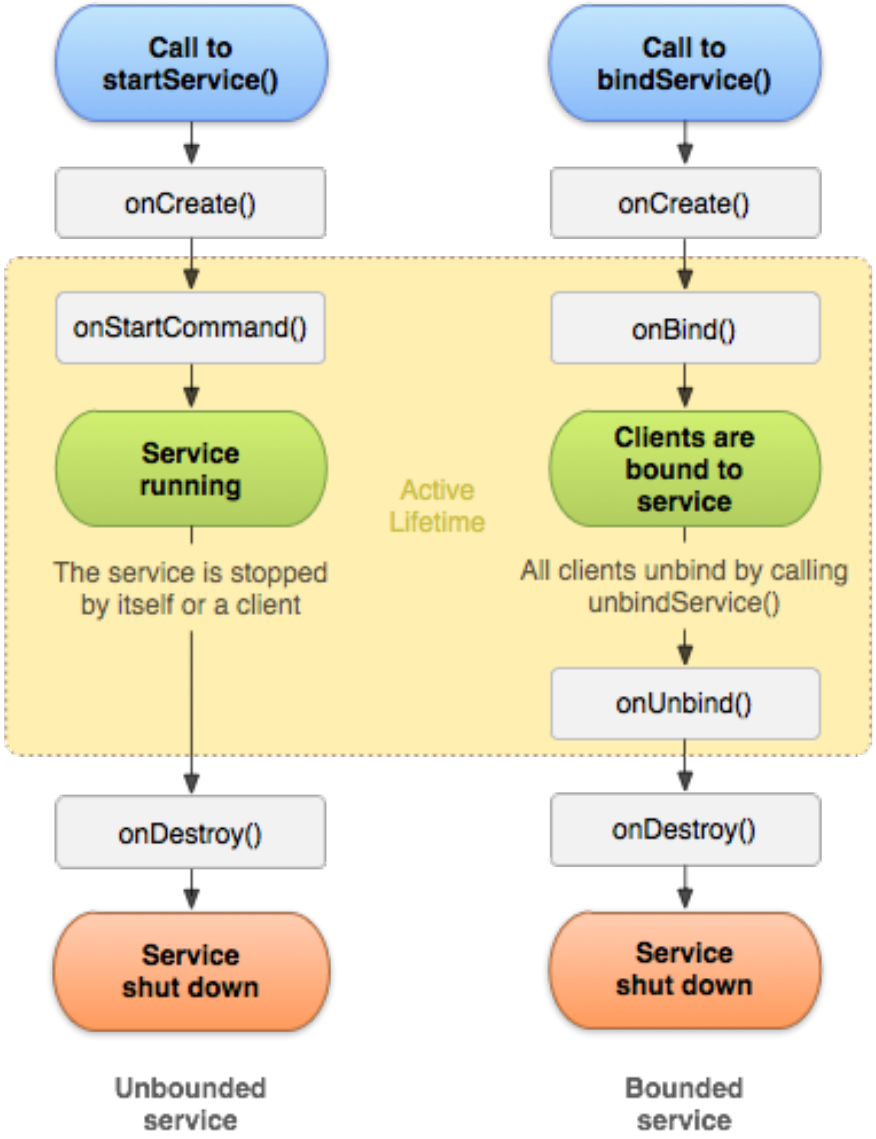
Services

- A Service is an application component that runs in background, not interacting with the user, for an **indefinite** period of time.
- Services, like other application objects (activities, broadcast receivers…), run in the main thread of their hosting process.
- For CPU intensive operations (such as networking),a service *should spawn its own thread in which to do that work.*
- So what is the difference between Threads and Services?

Service vs Thread

- A service is in a sense similar to a Unix's daemon, e.g, it can be used system-wide (e.g., from other applications) and for example be started automatically after the device boot ends
- A service is a component that Android is aware of (it must be declared in the manifest), with its own lifecycle
- Android platform provides a set of system wide services
  - *getSystemService()*

Service behaviour: Unbounded bounded



Some example of system service

- AccessibilityManager, for being notified of key system events (e.g., activities starting) that might be relayed to users via haptic feedback, audio prompts, or other non-visual cues
- **AccountManager**, for working with Android's system of user accounts and synchronization
- **ActivityManager,** for getting more information about what processes and components are presently running on the device
- AlarmManager, for scheduled tasks (a.k.a., "cron jobs"), covered elsewhere in this book
- AppOpsManager, for "tracking application operations on the device"
- AppWidgetManager, for creating or hosting app widgets
- AudioManager, for managing audio stream volumes, audio ducking, and other system-wide audio affordances
- BatteryManager, for finding out about the state of the battery
- BluetoothManager, for exposing or connecting to Bluetooth services
- ClipboardManager, for working with the device clipboard, covered elsewhere in this book
- **ConnectivityManager**, for a high-level look as to what sort of network the device is connected to for data (e.g., WiFi, 3G)
- ConsumerIrManager, for creating "IR blaster" or other IR-sending apps, on hardware that has an IR transmitter
- DevicePolicyManager, for accessing device administration capabilities, such as wiping the device
- DisplayManager, for working with external displays, covered elsewhere in this book
- DownloadManager, for downloading large files on behalf of the user, covered in elsewhere in the book
- **DropBoxManager**, for maintaining your own ring buffers of logging information akin to LogCat
- FingerprintManager, for working with fingerprint readers on Android 6.0+ devices
- InputMethodManager, for working with input method editors
- InputManager, for identifying external sources of input, such as keyboards and trackpads
- **JobScheduler**, for scheduling periodic background work, covered elsewhere in the book
- KeyguardManager, for locking and unlocking the keyguard, where possible
- LauncherApps, for identifying launchable apps on the device (e.g., for home screen launchers), taking into account device policies
- **LayoutInflater**, for inflating layout XML files into Views, as you saw earlier in the book
- **LocationManager**, for determining the device's location (e.g., GPS), covered in the chapter on location tracking
- MediaProjectionManager, for capturing screenshots and screencasts
- MediaRouter, for working with external speakers and displays
- MediaSessionManager, for teaching Android about media that you are playing back

Some example of system service

- MidiManager, for playing MIDI audio
- **NetworkStatsManager**, “for querying network usage stats”
- NfcManager, for reading NFC tags or pushing NFC content
- NotificationManager, for putting icons in the status bar and otherwise alerting users to things that have occurred asynchronously, covered in the chapter on Notification
- NsdManager, for network service discovery operations
- PowerManager, for obtaining WakeLock objects and such, covered elsewhere in this book
- PrintManager, for printing from Android
- RestrictionsManager, for identifying and working with restricted operations
- SearchManager, for interacting with the global search system
- **SensorManager**, for accessing data about sensors, such as the accelerometer, covered elsewhere in this book
- StorageManager, for working with expanded payloads (OBBs) delivered as part of your app’s installation from the Play Store
- SubscriptionManager, for dealing with data roaming and other telephony subscription rules
- TelecomManager, for dealing with incoming and outgoing phone calls
- **TelephonyManager**, for finding out about the state of the phone and related data (e.g., SIM card details)
- TextServicesManager, for working with spelling checkers and other “text services”
- TvInputManager, for Android-powered televisions, to find out about TV inputs
- UiModeManager, for dealing with different “UI modes”, such as being docked in a car or desk dock
- UsageStatsManager, “for querying device usage stats”
- UsbManager, for working directly with accessories and hosts over USB
- UserManager, for working with multiple user accounts on a compatible device (Android 4.2+ tablets, Android 5.0+ phones)
- Vibrator, for shaking the phone (e.g., haptic feedback)
- WallpaperService, for working with the device wallpaper
- WifiManager, for getting more details about the active or available WiFi networks
- WifiP2pManager, for setting up and communicating over WiFi peer-to-peer (P2P) networks
- WindowManager, mostly for accessing details about the default display for the device

Service types: Foreground services

- A foreground service performs some operation that is noticeable to the user. For example, an audio app would use a foreground service to play an audio track. Foreground services *must* display a **status bar icon**.
  - *startForeground(id,Notification)* method

- Google recommends to use WorkManager to replace Foreground services

Service types: Background services

- A background service performs an operation that isn't directly noticed by the user. For example, if an app used a service to compact its storage, that would usually be a background service.

- Stating from Oreo, for performance reasons the system imposes restrictions on running background services when the app itself is not in the foreground.
  - Switch to WorkManager