Service란?

화면 없이 백그라운드에서 실행되는 프로세스

서비스도 애플리케이션의 구성 요소이므로 시스템에서 관리한다. 따라서 새로 만든 서비스는 항상 매니페스트 파일에 등록해야 하며 서비스를 실행 시킬 때는 메인 액티비티에서 startService()메소드를 호출하면 된다.

서비스가 하는 중요한 역할은 단말이 항상 실행되어 있는 상태로 다른 단말과 데이터를 주고 받거나 단말의 상태를 모니터링하는 것이다. 따라서 서비스를 한번 실행해두면 실행된 상태가 계속 유지되어야 한다. 이를 위해 서비스가 비정상적으로 종료되더라도 시스템이 자동을 실행된다.

그런데 서비스가 한번 실행된 후 종료되지 않고 계속 실행 중인 상태라면 startService() 메소드를 여러 번 호출하더라도 서비스의 상태에는 변화가 없게 된다. 따라서 startService()는 서비스를 시작하는 목적보다는 인텐트를 전달하는 목적으로 더 많이 사용된다.

예를 들어, 액티비티에서 서비스 쪽으로 데이터를 전달하고 싶은 경우 인텐트 객체를 만들고 부가 데이터를 넣은 후 startService() 메소드를 호출하면서 전달하면 된다. 이때 서비스 객체가 메모리에 만들어져 있으면 onCreate() 메소드가 호출되지 않으므로 onStartCommand() 메소드를 사용해 전달 받은 인텐트는 처리하게 된다.

**Service 콜백 메소드** (다른 함수의 인자로써 이용되는 함수, 어떤 이벤트에 의해 호출되는 함수)

서비스를 생성하려면 Service의 서브클래스를 생성해야 한다. 구현에서는 서비스 수명 주기의 주요 측면을 처리하는 콜백 메서드를 몇 가지 재정의해야 하며 서비스에 바인드할 구성 요소에 대한 메커니즘을 제공해야 한다.

onStartCommand()

시스템이 이 메서드를 호출하는 것은 또 다른 구성 요소가 서비스를 시작하도록 요청하는 경우

이때 startService()를 호출하는 방법을 사용한다. 이 메서드가 실행되면 서비스가 시작되고 백그라운드에서 무기한으로 실행될 수 있으며 stopSelf() 또는 stopService()를 호출할떄 멈추지 않는다

onBind()

시스템이 이 메소드를 호출하는 것은 또 다른 구성 요소가 해당 서비스에 바인드되고자 하는 경우이다. 이때 bindService()를 호출하는 방법을 쓴다. 이 메소드를 구현할 때에는 클라이언트가 서비스와 통신을 주고받기 위해 사용할 인터페이스를 제공해야 한다. 이때 IBinder를 리턴하면 된다. 이 메서드는 항상 구현해야 하지만, 바인딩을 허용하지 않고자 하면 null을 리터하면 된다.

바인딩(연결)된 서비스란 일종의 [Service](https://developer.android.com/reference/android/app/Service?hl=ko) 클래스 구현으로, 이를 통해 다른 애플리케이션이 이 서비스에 바인딩하여 상호작용할 수 있도록 합니다. 서비스에 대한 바인드를 제공하려면 [onBind()](https://developer.android.com/reference/android/app/Service?hl=ko" \l "onBind(android.content.Intent)) 콜백 메서드를 구현해야 합니다.

startService() 를 통해 서비스가 시작되고 이러한 서비스는 한번 시작되면 백그라운드에서 무한정 실행될수 있습니다.  심지어 서비스를 시작한 액티비티가 소멸되어도 마찬가지 입니다.  그러나 이렇게 시작한 서비스는 호출한 쪽에 어떠한 결과를 반환할수 없습니다.  이를 Unbound Service 라고도 합니다.

startService() 메소드 대신 **bindService()**메소드를 통해 시작되는 서비스를 서비스 바인딩 (Service Bind 혹은 Bound Service) 라 합니다.

- 액티비티는 서비스에 어떠한 요청을 할수 있고, 서비스로부터 어떠한 결과를 받을수 있습니다.

- 프로세스간 통신에도 사용됩니다.

- 서비스 바인딩은 연결된 액티비티가 사라지면 서비스도 소멸됩니다. (즉 백그라운드에서 무한히 실행되진 않습니다)

- 하나의 서비스에 다수의 액티비티 연결 가능합니다

- 애플리케이션 안의 기능을 외부에 제공하는 경우에 많이 사용합니다.

onCreate()

시스템이 이 메소드를 호출하는 것은 서비스가 처음 생성되어 일회성 설정 절차를 수행하는 경우이다. (onStartCommand()또는 onbind()호출하기 전). 서비스가 이미 실행중인 경우, 이 메소드는 호출되지 않는다.

On Destory()

시스템이 이 메소드를 호출하는 것은 해당 서비스를 더 이상 사용하지 않고 소멸시키는 경우이다. 서비스에서 이를 구현해야 하는 스레드, 리스너, 리시버 등 모든 리소스를 정리할 수 있다. 이는 서비스가 수신하는 마지막 호출이다.

onStartCommand() 메서드 리턴 값

START\_NOT\_STICKY

시스템에 의해 강제로 종료 되어도 Service가 재시작 하지 않습니다.

START\_STICKY

Service가 강제 종료되었을 경우 시스템이 다시 Service를 재시작 시켜 주지만 intent 값을 null로 초기화 시켜서 재시작 합니다.

START\_REDELIVER\_INTENT

START\_STICKY와 마찬가지로 Service가 종료 되었을 경우 시스템이 다시 Service를 재시작 시켜 주지만 intent 값을 그대로 유지 시켜 줍니다

서비스바인딩?

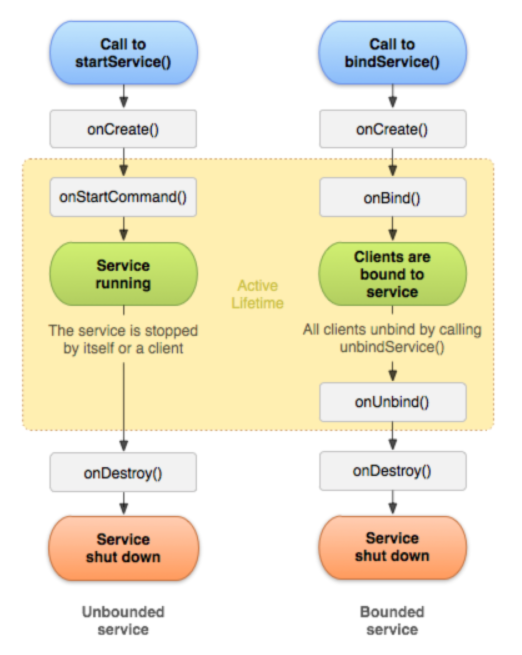
startService() 메소드 대신 bindService() 메소드를 통해 시작되는 서비스를 서비스 바인딩이라 한다.

이 서비스는 마치 클라이언트-서버 관계처럼 동착하며 서비스가 서버 역할을 한다.

액티비티는 서비스에 어떠한 요청을 할 수 있고, 서비스로부터 어떠한 결과를 받을 수 있다.

서비스 바인딩은 연결된 액티비티가 사라지면 서비스도 소멸된다.(무한히 실행 x)

하나의 서비스에 다수 액티비티를 연결할 수 있으며 애플리케이션 안의 기능을 외부에 제공하는 경우에 사용된다.



<서비스 생명 주기>

서비스의 전체 수명 주기는 onCreate가 호출된 시점과 onDestroy가 반환된 시점 사이에 일어납니다.

액티비티와 마찬가지로 서비스는 자신의 초기 설정을 onCreate에서 수행하며, 남은 리소스를 모두 onDestroy에서 릴리스합니다.

서비스의 활성 수명 주기는 onStartCommand 또는 onBind 대한 호출로 시작됩니다.

각 메서드에 Intent가 전달됩니다. 이것은 각각 startService 또는 bindService에 전달된 것입니다.

서비스가 시작되면 수명 주기 전체가 종료되는것과 동시에 활성 수명 주기도 종료됩니다. (서비스는 onStartCommand가 반환된 뒤에서 여전히 활성상태입니다.)

서비스가 바인드된 경우, onUnbind가 반환되면 활성 수명 주기도 종료됩니다.

**Intent Service**

서비스는 호출 앱의 메인 스레드상에서 실행된다. 따라서 서비스가 시간을 너무 오래 끌면 그동안 사용자의 입력에 반응할 수 없다. 아주 짧은 작업이라면 상관없지만 오래 걸린다면 별도의 스레드로 분리하는 것이 바람직하다.

서비스에서 하는 작업은 대체로 시간이 오래 걸리므로 스레드를 생성하여 작업을 분담하는 것이 일상적이다. 보통 멀티 스레드로 작성하며 구조가 뻔하므로 이런 구조를 자동으로 처리하는 IntentService 클래스가 제공된다.

Service의 서브 클래스이며 인텐트로 요청을 전달받아 별도의 스레드를 생성하고 onHandleIntent 메서드를 호출한다.

뻔한 코드를 미리 구현해 놓은 편의 메서드이므로 생명 주기 메서드는 일체 구현할 필요없이 onHandleIntent에서 요청에 대한 처리만 수행하면 된다.

한번에 하나의 요청만 처리할 수 있으며 시간이 좀 오래 걸리더라도 분리된 스레드에서 실행되므로 메인 스레드의 작업을 방해하지 않는다.

Service의 서브클래스로, Worker Thread를 사용하여 모든 시작 요청을 처리하되 한 번에 하나씩 처리한다. 서비스가 여러 개의 요청을 동시에 처리하지 않아도 되는 경우 Intent Service를 사용하는 것이 좋다.

기본 스레드와는 별도로, onStartCommand에 전달된 모든 인텐트를 실행하는 기본 작업자 스레드를 생성한다.

한번에 인텐트를 하나씩 onHandleIntent 구현에 전달하는 작업 큐를 생성하므로, 다중 스레딩에 대해 걱정할 필요가 없다.

시작 요청이 모두 처리된 후 서비스를 중단하므로 stopself를 호출할 필요가 없다.

onBind 기본 구현을 제공하여 null을 반환하도록 한다.

onStartCommand의 기본구현을 제공하여, Intent를 작업 큐로 보내고 그 다음은 onHandleIntent 구현으로 보낸다.

**Foreground Service**

항상 서비스가 전면에 올라와있다. 서비스가 백그라운드에서 계속 동작할 때 서비스가 다른 앱에서 메모리를 더 필요로 하거나 하는 상황이 오면 안드로이드 시스템에 의해 종료될 가능성이 있다.

지속적으로 강제종료되지 않는 안전한 서비스를 사용하기 위해서는 포그라운드 서비스를 사용해야한다. 하지만 foreground service를 남용하게 되면 안되기 때문에 몇가지 제약을 지켜야한다.

Notification와 함께 제공해야 하며 알림 id를 0이 아닌 값을 줘야 하며 startForeground메서드를 서비스 내부에서 별도로 실행해서 foreground로 변경시켜주어야 한다.

MainAcitivity에서 startForegorundServie를 실행한 후에 서비스 클래스에서 startForeground를 5초 이내에 실행하지 않으면 ANR가 발생할 수 있다.

Android 앱의 UI 스레드가 너무 오랫동안 차단되면 'ANR(애플리케이션 응답 없음)' 오류가 트리거됩니다

ANR은 Application Not Responding의 약자로 그대로 해석해보면 의미를 쉽게 파악할 수 있다. '애플리케이션이 응답하지 않는다.' 인 것이다. 이 에러의 원인은 Main Thread(UI Thread)가 일정 시간 어떤 Task에 잡혀 있으면 발생하게 된다.

**Bind Service**

startService()를 통해 시작되는 UnBound Service와는 다르게 액티비티 및 프래그먼트와 데이터를 주고 받을 수 있으며 프로세스간의 통신에도 사용된다. 서비스를 실행시켜두고 필요할 때마다 서비스의 메소드에 접근하여 통신을 할 수 있는 구조이다.