Android Service

[Android][Android Tips]

・http://developer.android.com/reference/android/app/Service.html

概要

・Sevice は、長い期間ユーザーと対話なしに処理を実行させたり、他のアプリケーションに機能を提供する要件を実現するためのアプリケーションコンポーネント。
・すべてのサービスは、AndroidManifest.xml の <service> 宣言と一致する必要がある
・サービスは、Context.startService() や Context.bindServce() により開始することができる

サービスとは

サービスについての主な混乱は「何でないか」

・サービスは、分離されたプロセスではない
・特別に指定しない場合、サービスオブジェクトは自分自身のプロセスで実行しているという意味ではなく、アプリケーションの一部としてアプリケーションと同じプロセスで実行される
・サービスはスレッドではない
・(Application Not Respoinding エラーを回避するために) メインスレッドから離れて働くことを意味しない。

サービスはシンプルで、主な 2 つの機能を提供する

1. 処理をバックグラウンドで実行したい (ユーザーは直接アプリケーションと対話したくない場合)。この場合は、Context.startServece() を呼び出す。サービス自身か他から明示的に停止されるまで実行するようにシステムに作業のスケジューリングを依頼する。
2. アプリケーションの機能を他のアプリケーションに見えるようにする。この場合は、Context.bindService() を呼ぶ。長期間のコネクションをサービスとの対話のために許可する。

・サービスコンポーネントが実際に作成されると、コンポーネントをインスタンス化し、onCreate() を呼び出し、メインスレッドの他の適切なコールバックを呼び出すことが、システムが行うすべて。
・サービスに適切な振る舞い (サービスの中で働く二次的なスレッドを生成するような) を実装させることができる

サービスはとてもシンプルなので、希望によって、シンプルな命令や複雑な命令を、リモートインターフェースである AIDL を利用して、ローカル Java オブジェクトを取り扱うように直接メソッドをコールすることで実行できる。

ライフサイクル

・システムによるサービスの起動には 2 つの種類がある。
・Context.startService() を呼び出した場合、システムはサービスを検索する (必要があれば、生成し onCreate() を呼び出す)。
・onStartCommand(Intent,int,int) をクライアントから与えられた引数を伴って呼び出す。
・サービスは、Context.stopService() もしくは stopSelf() が呼び出されるまでは、起動し続けている。

Context.statrService() を多重で呼び出してもネストされない。(たとえそれらが、複数の onStartCommand() に対応する結果を返すとしても) なので、何回サービスをスタートしたとしても問題無いし、一度の Context.stopService() または、stopSelf(int) の呼び出しで、サービス

は停止する。しかしながら、サービスは stopSelf(int) を、開始されたインテントが処理中で、停止していないのを保証するために利用することが出来る。

・サービスを開始するのに、2 つの主な追加の操作モードがある。
・それらは、どのように起動かを onStartCommand() から返される値に依存して決定する。
  ・START_STICKY は、明示的に必要に応じてサービスを開始したり停止したりするのに使用する。
  ・一方、START_NOT_STICKY または、START_REDELIVER_INTENT は、送信されたコマンドを処理する間、唯一残って起動し続ける。

・クライアントは Content.bindService() も持続的なサービスへのコネクションを取得するために利用できる。
・これは、まだ起動されていない場合、onCreate を呼び出し、同様にサービスを生成する
・しかし、onStartCommand() は呼び出さない。
・クライアントは、サービスの onBind(Intent) メソッドが返す IBinder オブジェクト を受け取る。
・IBinder オブジェクトは、クライアントにサービスへのコールバックを行うことを許可する
・サービスはコネクションが確立している間 (クライアントがサービスの IBinder への参照を保持していてもいなくても) 起動し続ける。通常 IBinder は、aidl で書かれた 複雑なインターフェースのために返される。

・サービスは両方の方法により開始され、コネクションを接続する。
・このような場合には、システムは、サービスを起動状態を維持する
・どちらの状況も成立しない時には、サービスの onDestroy() メソッドが呼ばれ、サービスは事実上終了する。すべてのクリーンアップ (スレッドの停止、サービスの登録解除) は onDestroy() から戻るとすぐに完了される。

パーミッション

・マニフェストの <service> タグで宣言された時に、サービスへのグローバルアクセスが実行出来る。
・そうした場合、サービスの開始、終了またはバインドのために、他のアプリケーションは一致した <uses-permission> 要素を自信のマニフェストでを宣言する必要がある。

・さらに、checkCallingPermission(String) メソッドを その呼び出しの実装より前に呼び出すことにより、サービスは固有の IPC(プロセス間通信) 呼び出しをパーミッションで保護することが出来る。

プロセスライフサイクル

・Android システム は、サービスが開始されているか、クライアントが接続している間、プロセスをサービスを受け入れられる状態に維持することを試みる。
・メモリーが少なくなり、存在するプロセスを殺す必要があるとき、サービスを受け入れるプロセスのプライオリティが次の可能性では、高くなる。