Androidでサービスしよう！

日本Androidの会
2月21日 大阪セミナー
有山圭二 (有限会社シーリス)

目次

- サービスって何？
- サービスで何が出来るの？
- ハンズオン
  - Serviceを開始する
  - ServiceとActivityの違い？
  - AIDLによるインターフェースの定義と実装
  - Activityからの操作
- まとめ
- 発展
- 参考

サービスって何？

- 誤解を覚悟で言えば、画面を持たない Activityです (この誤解は後で解きます)

- Activityが、その表示を終えた段階で終了するのに対して、Serviceは生存期間が長く、原則として終了の指示があるまではシステム上に生存を続けます。

サービスで何が出来るの？

- Serviceを使えば、Android携帯用の常駐アプリを開発する事が出来ます。
- 画面を必要とせず、かつ継続的な処理が必要な処理、音楽の再生やセンサー系の監視が可能になります。
- 外部から接続するインターフェースを定義し、様々な操作を指示したり、逆にサービスの状態が変化した際に、その情報を受け取る事が出来ます。

ハンズオン

プロジェクトを作成する

- File → New → Android Project を選択
  - Project name : ServiceTester
  - Package name : hoge.foo
  - Activity name : MainActivity
  - Application name :ServiceTester

Serviceを追加する

- hoge.fooパッケージ下に、クラスを追加
  - 名前 : TestService
  - スーパークラス : android.app.Service

- AndroidManifest.xmlの
  Application → Application Nodesに追加

Serviceが起動した時の処理

- TestServiceのonCreateに、サービスが開始した時に解るように、以下を追加しておく。

```
@Override
public void onCreate() {
    super.onCreate();

    // Toastを表示する
    Toast.makeText(this, "Service start", Toast.LENGTH_LONG).show();
}
```

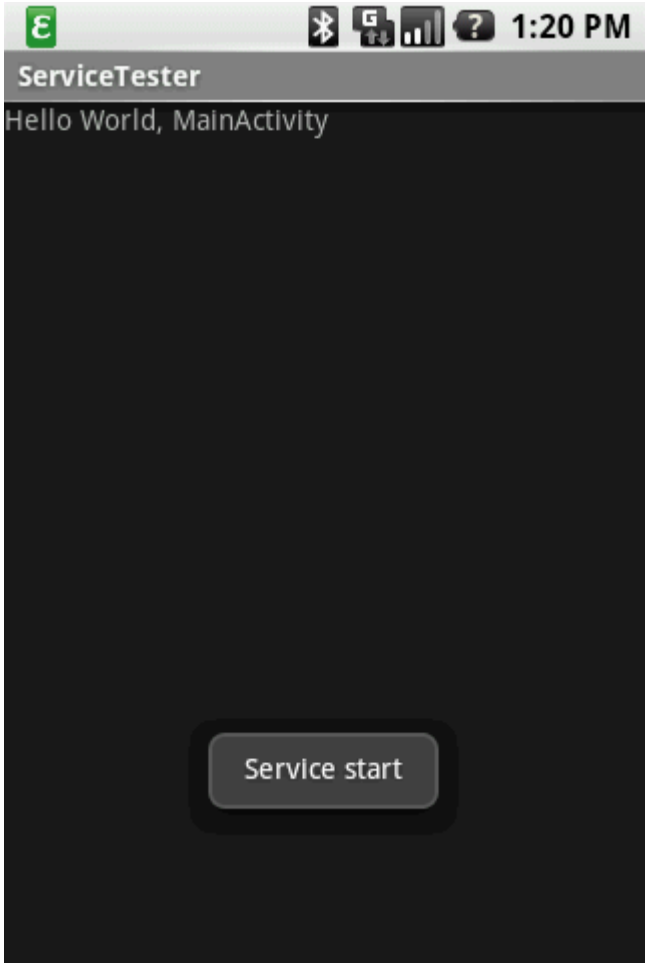
Activityから Serviceを開始する

- MainActivityのonCreateで、サービスを開始する。

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    // サービスを開始
    Intent intent = new Intent(this, TestService.class);
    startService(intent);
}
```

エミュレーターでアプリを起動

- **標準の画面の下に、**
  “Service start“**と、**
  表示されれば成功です。



Toastを定期的に表示する

- TestServiceの中で、Timerを使って、Toastを定期的に表示してみましょう。

追加するコード1 → TestService

```
// 連続表示するToast
private Toast _mToast = null;

// ハンドラ
private final Handler _mHandler = new Handler();

@Override
public void onCreate() {

+    // Toastを作成
+    _mToast = Toast.makeText(this, "Service start", Toast.LENGTH_LONG);

}
```

- Handlerを忘れないで下さい。
- Toastはクラスメンバにして、onCreate内でインスタンス化しておきます。

追加するコード2 → TestService

```
// タイマ
private final Timer _mTimer = new Timer();

// タイマに登録するタスク
private TimerTask _mTimerTask = new TimerTask() {
        @Override
        public void run() {

                // Handlerを介してpostする
                _mHandler.post(new Runnable() {
                        @Override
                        public void run() {
                                // Toastの表示
                                _mToast.show();
                        }
                });
        }
};
```

- TimerTask内から画面表示を変更する場合は、Handlerを介してpostして行います。

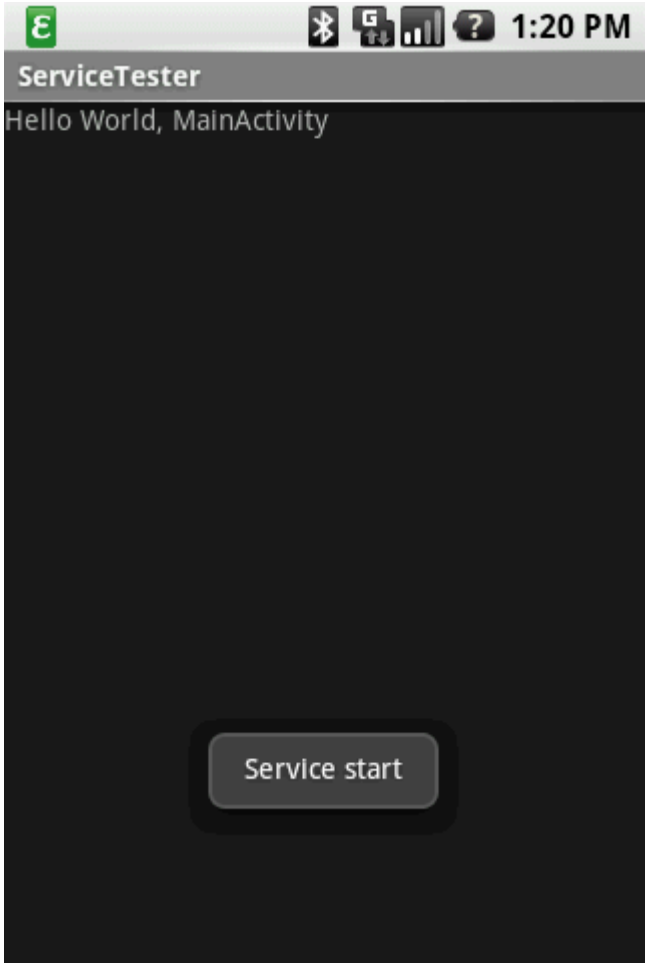
追加するコード3 → TestService

```
@Override
public void onCreate() {

+    // タイマに登録 1秒後から、10秒に1回、TimerTaskを実行する
+    _mTimer.schedule(_mTimerTask, 1000, 10 * 1000);

}
```

- Timerに登録しましょう。

エミュレーターでアプリを起動

- **標準の画面の下に、１０秒毎に**
  “Service start“**と、表示され続ければ成功です。**



Activityで同じ事をやってみる

- TestServiceに追加したコードを、MainActivityに追加して、どうなるかやってみましょう。

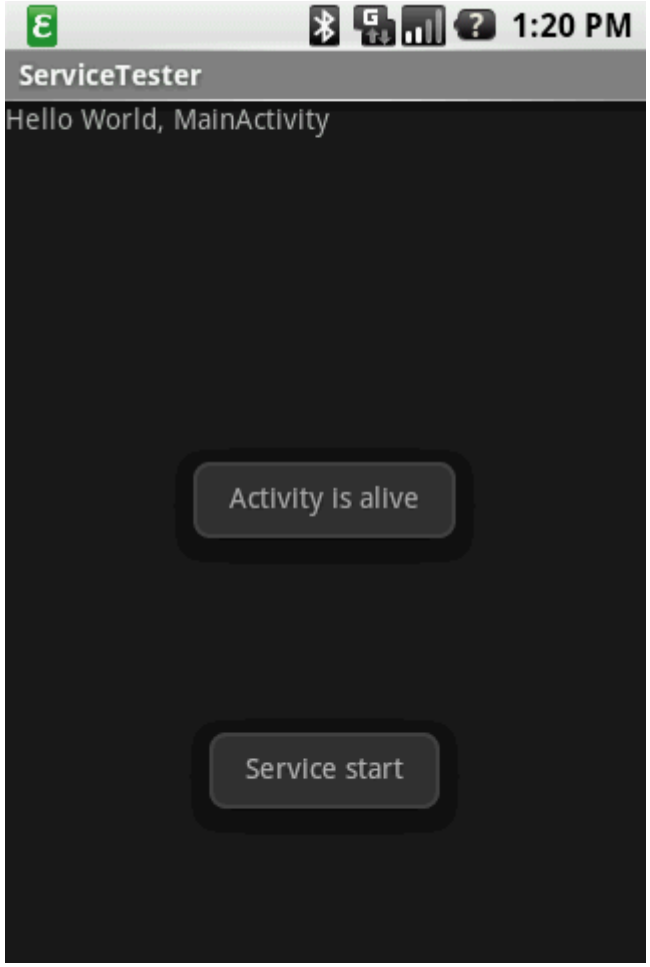
変更するコード → MainActivity

```
@Override
public void onCreate(Bundle savedInstanceState) {

+    // Toastを作成
+    _mToast = Toast.makeText(this, "Activity is alive", Toast.LENGTH_LONG);

+    // 表示する場所を変更 - 画面中央
+    _mToast.setGravity(Gravity.CENTER, 0, 0);

+    // タイマに登録 1秒後から、5秒に1回、TimerTaskを実行する
+    _mTimer.schedule(_mTimerTask, 1000, 5 * 1000);

}
```

- 分かり易くする為、Toastの表示場所を変更しておく。
- 表示間隔も5秒毎に。

エミュレーターでアプリを起動

- **標準の画面の下に、１０秒毎に**
  “Service start“**と、表示ます。**
- **また、画面中央には、**
  “Activity is alive”
  と、表示されます。



バックボタンを押して、

Activityを終了しましょう。

どうなるでしょうか。

何故か表示され続けるToast

- Activityは終了しているのに、Activityで処理しているはずのToastは、依然として表示され続けます。
- Timerが実行されていると、画面を消してもActivityは生存する。

では、Activity**も**
Service**も同じなのでしょうか？**

答えは、No

- 今回の例では、ただActivityが制御不能になっただけで、こうすればActivityをサービスと同様に使える訳ではない。
  - その証拠に、ServiceTestを起動する度に、Activityからの Toastが表示される間隔が重複していく。
    - 複数のActivityのTimerから、同時にToastが表示されてしまう。

サービスは、Activityから接続して、操作する事が出来る

- サービスは、Activityからbindすることにより操作する事が出来ます。
- サービスにbindして、”Service start”のメッセージ表示を止める事の出来る処理を追加してみましょう。

aidlを追加する

- **外部から可能な操作は、**AIDL[Android Interface Definition Language]**を用いて**.aidl**ファイルに定義します。**
- hoge.foo**パッケージ下に、ファイルを追加**
  - **ファイル名**： ITestService.aidl

追加するコード → ITestService.aidl

```
package hoge.foo;

interface ITestService {

    // stopServiceと書いてあるけど、Toastを停止するだけです。
    void stopService();

}
```

- **プロジェクトをBuildすると、.aidlは、Javaの**
  interface形式に自動的にコンバートされます。

Service側の準備

追加するコード → TestService

```
// サービスバインダー
private final ITestService.Stub mITestServiceBinder = new ITestService.Stub() {

	@Override
	public void stopService() throws RemoteException {
		// タイマタスクの停止
		_mTimerTask.cancel();
	}
};

@Override
public IBinder onBind(Intent arg0) {
	return mITestServiceBinder;
}
```

- stopServiceを指示されるとタイマタスクを停止するようにしました。

Activity**側の準備**

追加するコード → MainActivity

```
// サービスとの接続
private ITestService _mITestServiceBind = null;
private ServiceConnection _mServiceConnection = new ServiceConnection() {
      @Override
      public void onServiceConnected(ComponentName name, IBinder service) {
            // インターフェースを取得
            _mITestServiceBind = ITestService.Stub.asInterface(service);
      }
      @Override
      public void onServiceDisconnected(ComponentName name) {
      // TODO Auto-generated method stub
      }
};

 @Override
  public void onCreate(Bundle savedInstanceState) {

+     // サービスとの接続
+       // サービスが起動していない場合は自動で起動する
+        bindService(intent, _mServiceConnection, BIND_AUTO_CREATE);
}
```

準備完了！ でもその前に

- Activityからを操作する為に、ユーザーからの入力を受けられるようにしましょう。
- Activityに“stop service”ボタンを表示して、ボタンがクリックされたら、Toast表示を停止するようにに対して指示します。

追加するコード1 → res/layout/main.xml

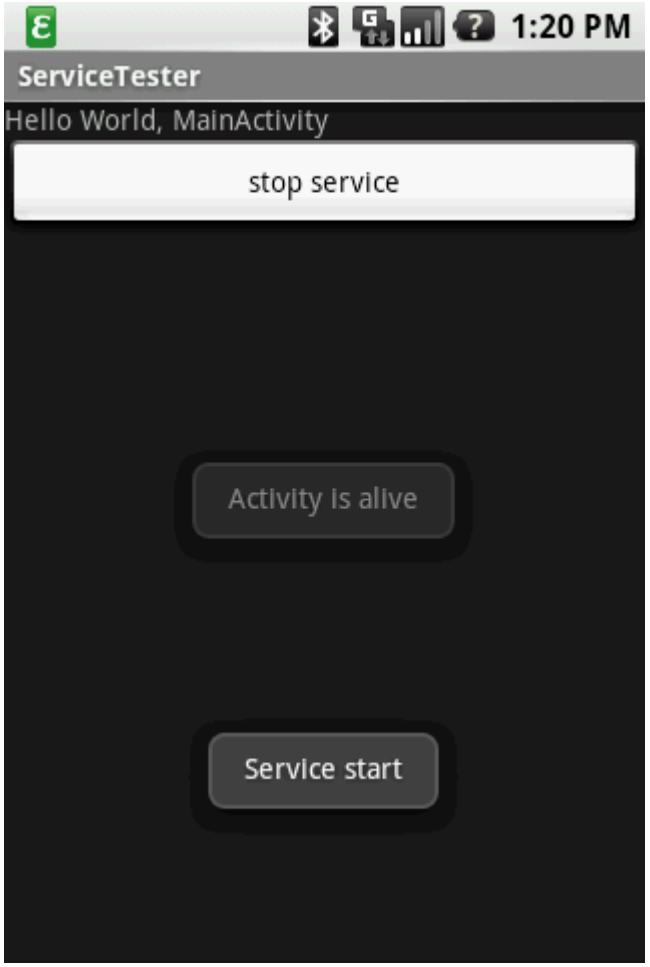
```
<Button  android:id="@+id/main_btn_stop"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="stop service"
/>
```

追加するコード2 → MainActivity

```
private Button _mBtnStop = null;
@Override
public void onCreate(Bundle savedInstanceState) {
    _mBtnStop = (Button)findViewById(R.id.main_btn_stop);
    _mBtnStop.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View arg0) {
            // サービスを停止
            // ＊正確には、Toastの表示を停止する
            try {
                _mITestServiceBind.stopService();
            } catch (RemoteException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    });
}
```

エミュレーターでアプリを起動

- “stop service”**のボタンを押すと、**
  “Service start”**の表示が止まれば成功です。**



Activityの方はどうする？

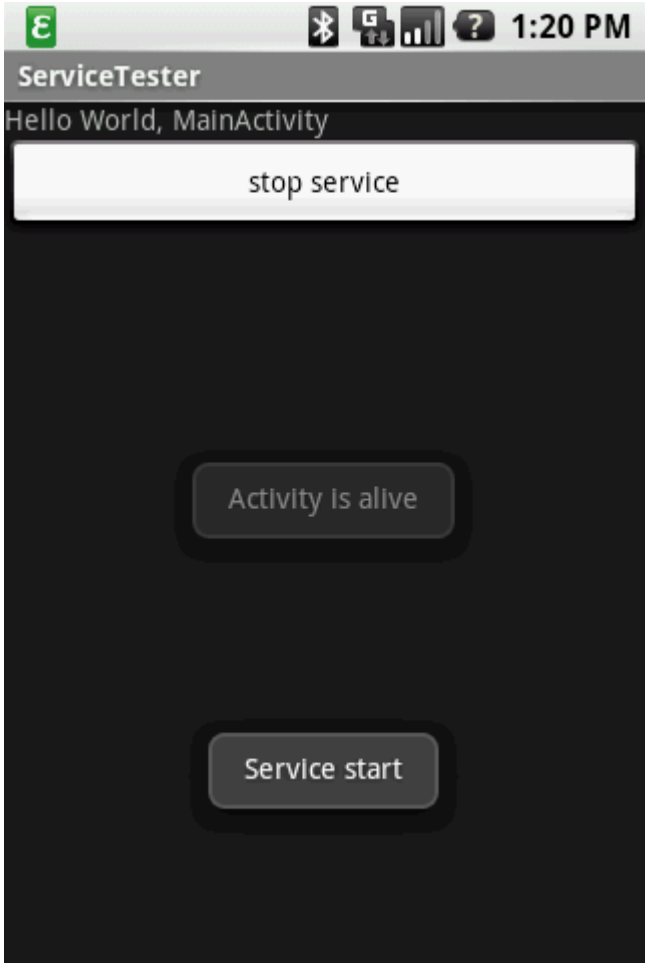
- MainActivityのonStop()をオーバーライドして、タイマを停止、サービスとの接続も解除する。

追加するコード → MainActivity

```
 @Override
public void onStop() {
    super.onStop();

    // タイマを停止する
    _mTimer.cancel();

    // サービスとの接続を解除する
    unbindService(_mServiceConnection);
}
```

エミュレーターでアプリを起動

- **起動すると、**
  “Activity is alive”(A) **が** 5**秒毎、**
  “Service start”(B) **が**10**秒毎に表示されます。**
- **バックキーでアプリを終了すると、**
  A**のメッセージは表示されなくなりますが、**B**のメッセージは引き続き表示されます。**
- **再びアプリを起動し、**”stop service”**ボタンを押すと、**B**のメッセージは表示されなくなります。**
- **アプリを終了すると、以後、全てのメッセージは表示されません。**



まとめ

- サービスとはシステムに常駐して、継続的な処理を行う際に利用されます。
- Activityは、終了と同時にTimer等に対して適切な処置をしておかなければ、制御不能に陥る恐れがあります。
- 一方でサービスは、AIDLでインターフェースを定義して、それを実装する事により、外部から操作する事が出来ます。
- また、サービス側から自らに接続しているActivityに対して、状態の変化を通知する事も可能です。(同じく、AIDLを使います)

発展

- 今回の実装では、Timerを停止しただけでService本体は停止していません。
- ITestService.aidlに、stopServiceとは別に、サービスを完全に停止するインターフェースを追加して、実装してみましょう。

参考

- Android SDK
  http://developer.android.com/

- Android: Service : **小山 圭氏 発表資料**