Android Activityのライフサイクル

2015/10/03

東北TECH道場 郡山道場

小俣 博司

Androidアプリの基本動作

通常のJavaのプログラムは、main()からプログラムが開始される構造になっていますが、Androidのプログラムは イベントドリブン(イベント駆動)のプログラムを作成することになります

「ボタンを押した」「マウスをクリックした」「アプリの状態が変化した」等の変化や通知がフレームワークやOSからアプリのプログラムに通知される仕組みのものです。

通常のJavaプログラム (関数呼び出し)

```
public class hoge {
    public static void main(String[] args){
        System.out.println("Hello, world.");
    }
}
```

余談：手続き型プログラミングの他に関数型プログラミングというのもある

Androidアプリ作成の基本“Activity”

Activityとは、“Androidアプリの画面”とイメージするとわかりやすいです。

Activityというクラスが、アプリの画面と紐付けられ、画面におけるライフサイクル(状態遷移)のイベントを処理します。

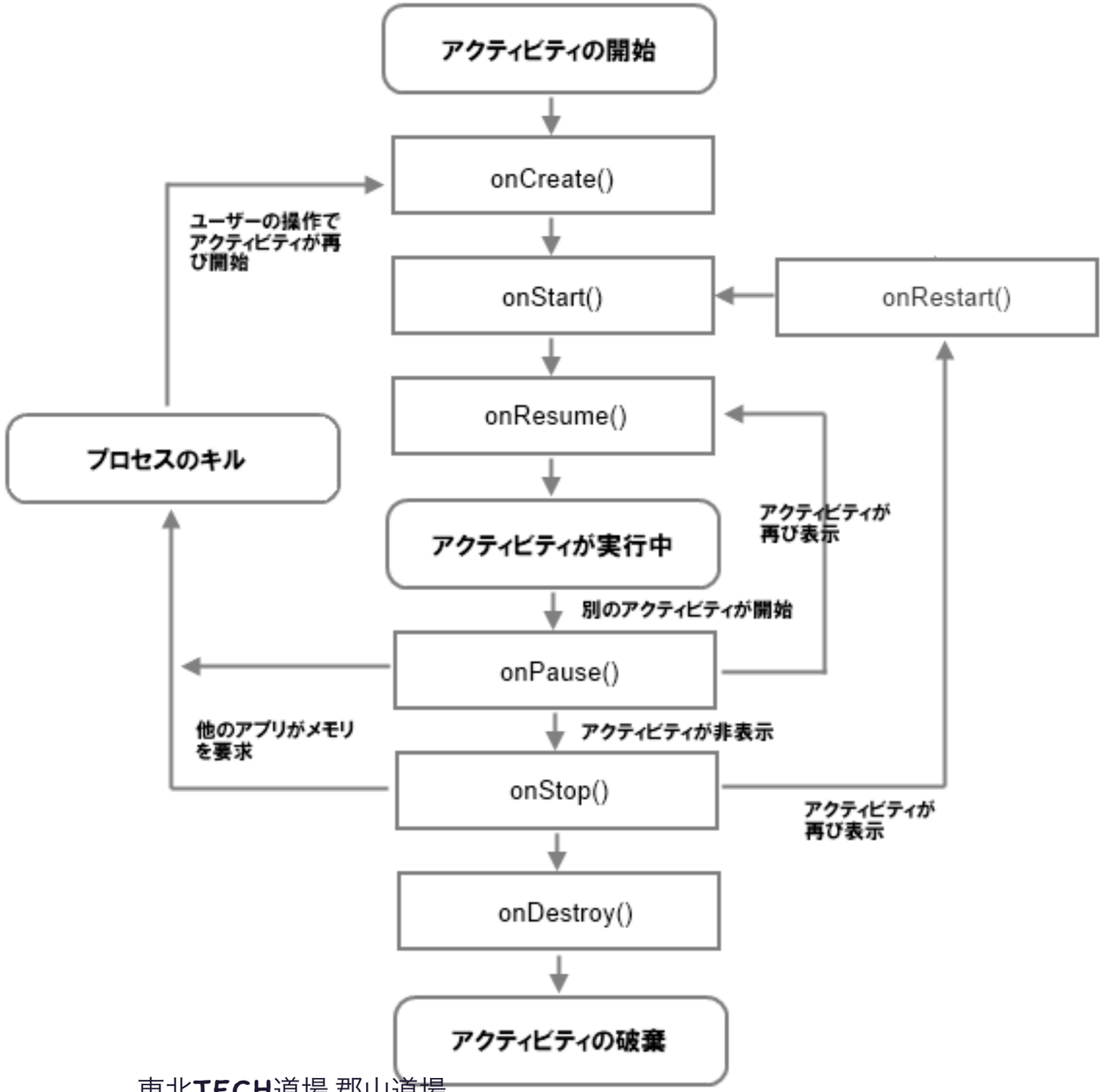
画面を始める準備 → 終了までの動作等をActivityで制御します

基本的なメソッド

メソッド名と呼ばれるタイミング[1]

- **onCreate() - Activity**が初めて作られたとき
- **onStart() - Activity**が開始されたとき
- **onResume() - Activity**が表示されたとき
- **onRestart() - Activity**が再度開始されたとき
- **onPause() -** 別の**Activity**が表示されるとき
- **onStop() - Activity**が表示されなくなったとき
- **onDestroy()- Activity**がメモリから開放される直前

[1] http://qiita.com/ymotongpoo/items/d8a054f6fc93d069cb37



アクティビティのライフサイクルと状態が変わる時に呼び出されるメソッドのフロー図。[1]

[1] http://qiita.com/ymotongpoo/items/d8a054f6fc93d069cb37

Activityクラスの基本コード

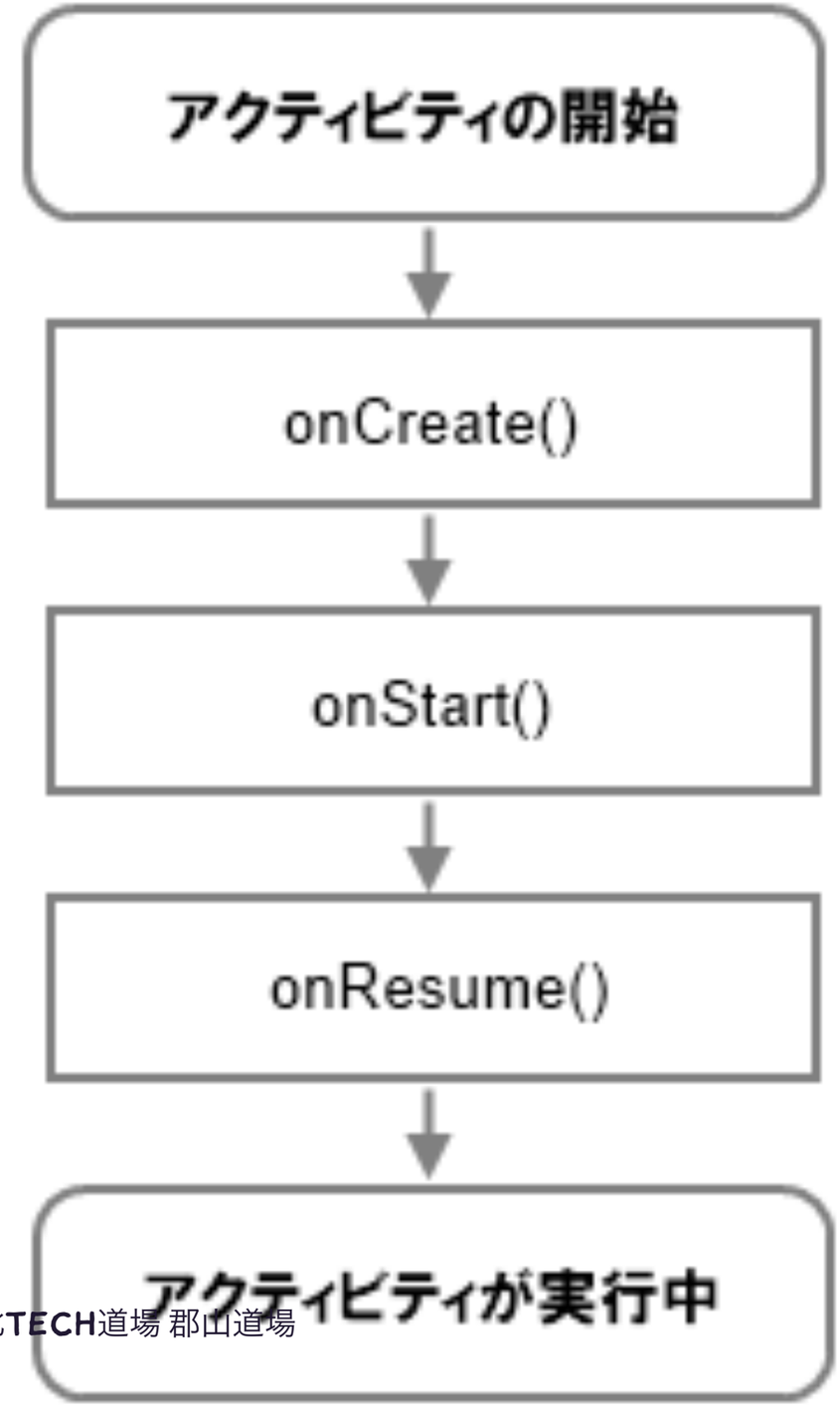
アンドロイドアプリはextends Activityという文言をclassに付加(Activityクラスを継承)をさせる事がデフォルトとなっている。[1]

```
public class Activity extends ApplicationContext {
    protected void onCreate(Bundle savedInstanceState);
    protected void onStart();
    protected void onRestart();
    protected void onResume();
    protected void onPause();
    protected void onStop();
    protected void onDestroy();
}
```

[1] http://qiita.com/ymotongpoo/items/d8a054f6fc93d069cb37

Android Studioの生成例

```java
package com.homata.myapplication;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```



アプリを起動しアクティビティが表示

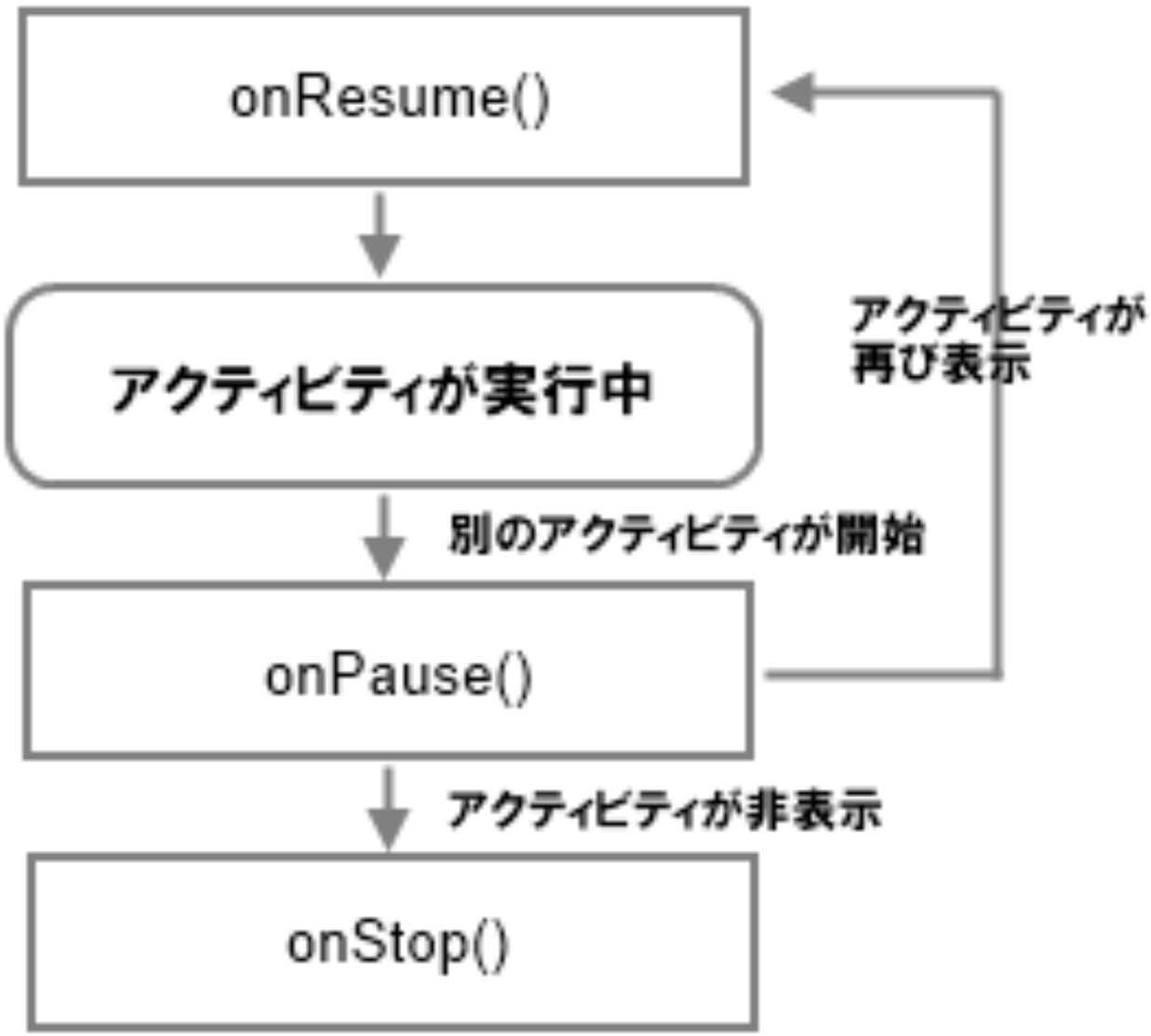
「アクティビティが開始」されると「**onCreate()**」メソッド、「**onStart()**」メソッド、「**onResume()**」メソッドが順に呼ばれて「実行中」の状態になります。メソッドが**3**つに分かれているのは、実行中になるまでに厳密には状態が細かく分かれているためで、その状態に応じて処理を行えるように分かれています。[1]

[1] http://qiita.com/ymotongpoo/items/d8a054f6fc93d069cb37

アプリ起動時のActivity

アプリ起動に利用されるActivityは「AndroidManifest.xml」ファイルで指定をします

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.homata.myapplication" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:theme="@style/AppTheme.NoActionBar" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

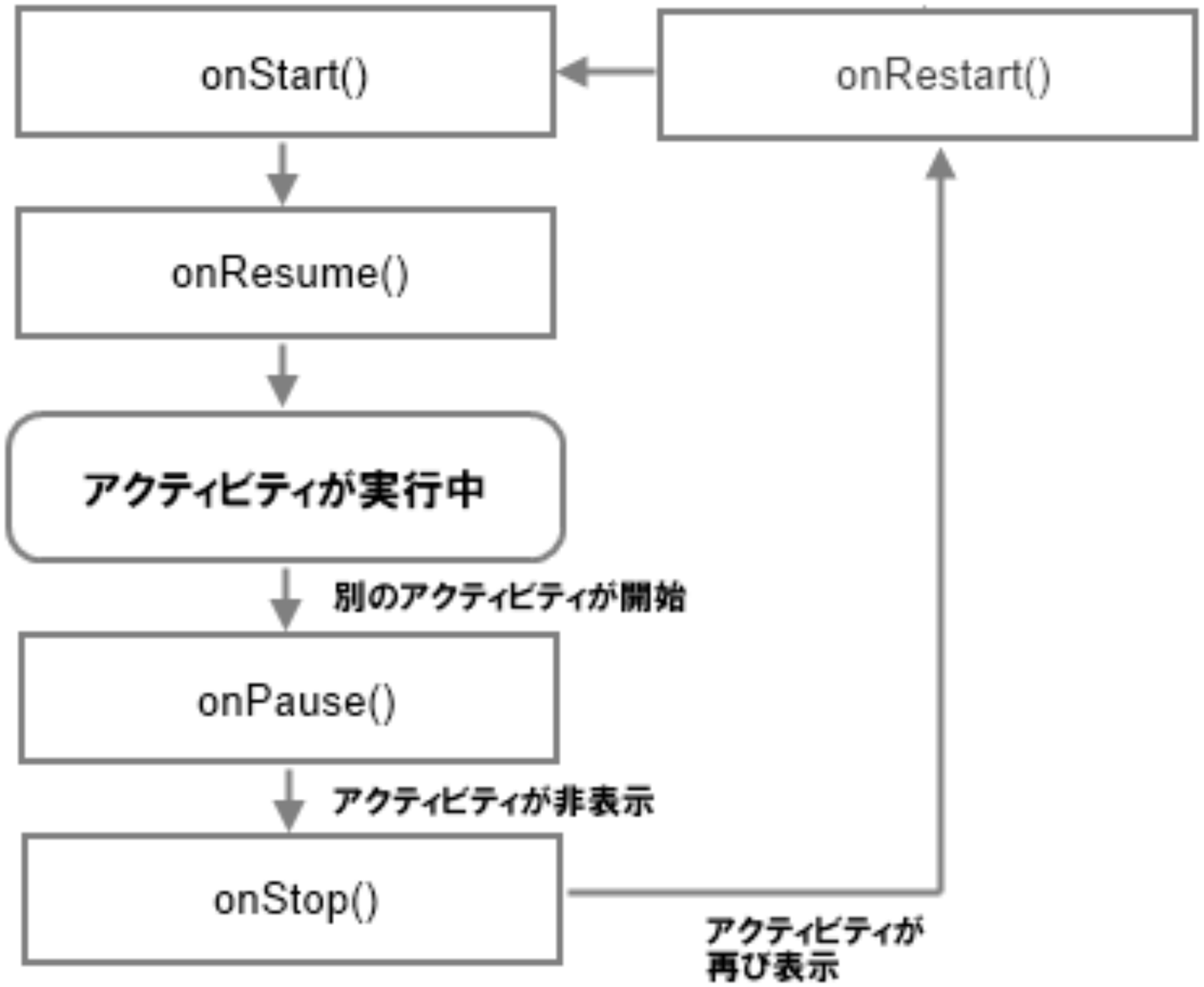


別のアクティビティが開始される時

同じアプリの別のアクティビティが開始されようとしたり、別のアプリのアクティビティが開始されようとすると、元のアクティビティは他のアクティビティによって見えなくなります。[1]

この時、別のアクティビティが画面に表示される前に「**onPause()**」メソッドが呼び出されます。

[1] http://qiita.com/ymotongpoo/items/d8a054f6fc93d069cb37



再び表示される時

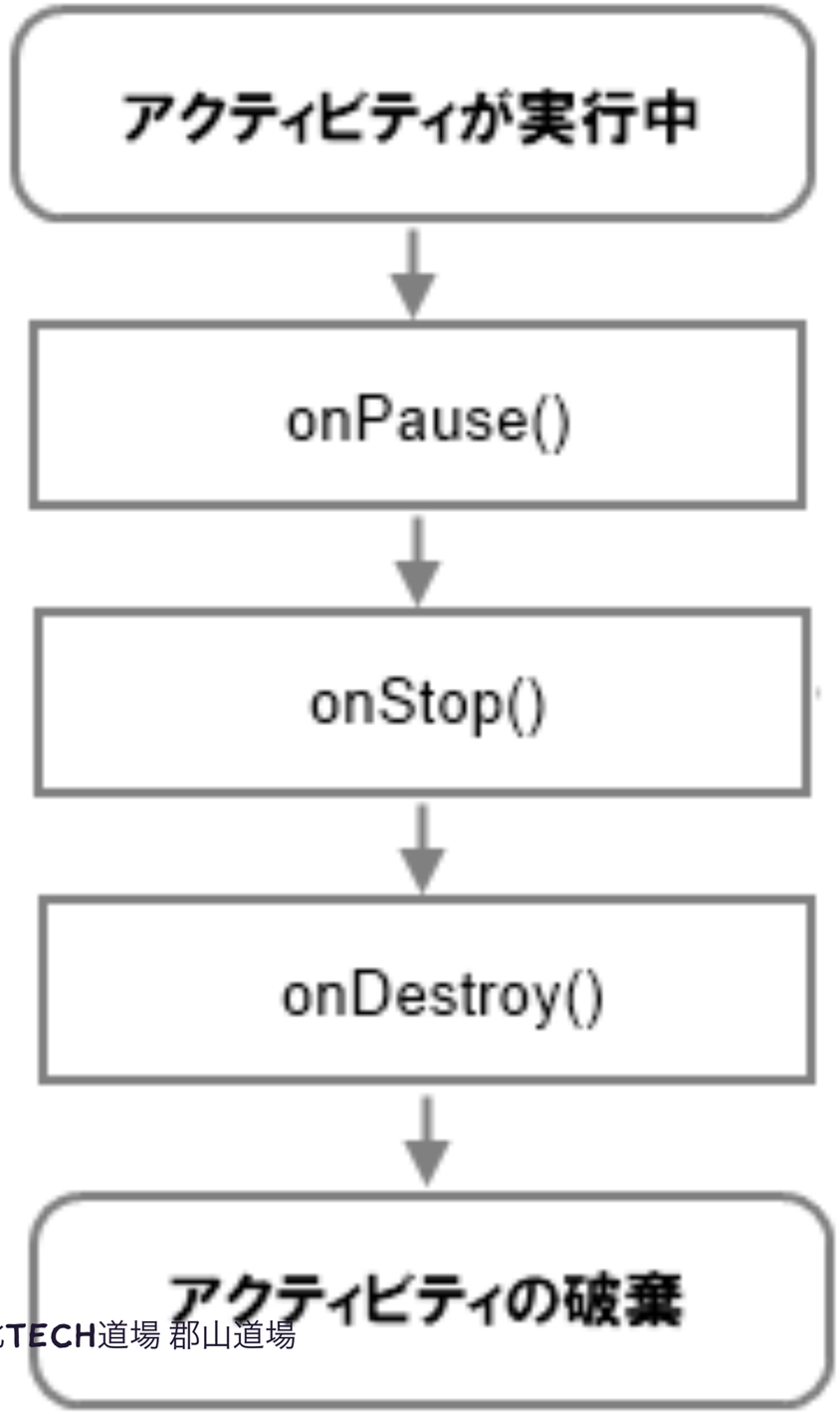
アクティビティは他のアクティビティの裏に隠れて表示されていない状態でも終了してはいません。停止の状態のままです。ユーザーによって非表示の状態から再び画面に表示されようとした場合には、まず「**onRestart()**」メソッドが呼ばれた後、「**onStart()**」メソッドと「**onResume()**」メソッドが呼ばれて再び実行中となります。[1]

ただ注意点があります。停止中でもアクティビティは終了はしていないのですが、**Android**のメモリが足りなくと、メモリを確保するために実行中でないアクティビティが強制的に終了となる場合があります。

[]^1]: 参照元**: http://www.javadrive.jp/android/activity/index2.html**

---

[1] **http://qiita.com/ymotongpoo/items/d8a054f6fc93d069cb37**



アクティビティの終了

アクティビティが終了する場合には最後に「**onDestroy()**」メソッドが呼ばれて終了します。

他のアクティビティが表示されたことによって見えなくなっても直ぐに終了とはならないですが、アクティビティが開始される前の状態に戻るとアクティビティはいったん終了となります。

アクティビティは開始された後で表示されたり他のアクティビティの後ろに隠れたりといった状態の変化を繰り返していきます

他の画面に遷移する場合

他の画面に遷移する場合は、インテントを利用します。インテントとｈはアクティビティを繋ぐものです。

1. 明示的Intent: Activityをクラス名で指定して呼び出す方法＝[1]
2. 暗黙的Intent: 動作(振る舞い)で呼び出し、Activityを指定しない方法

明示的Intent

```
java
import android.content.Intent;
...
Intent intent = new Intent(this, SubActivity.class);
intent.putExtra("foo", someData);
startActivity(intent)
java
```

暗黙的Intent

```
java
Uri uri = Uri.parse("http://www.bar.com/hoge");
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(intent);
```

[1] http://qiita.com/ymotongpoo/items/d8a054f6fc93d069cb37

公式ドキュメント

- アクティビティのライフサイクル 管理
- activity
- Intents and Intent Filters

The End