［スマホプログラミング］

② Android プログラミング入門

石丸宗平

Android プログラミングの概要

Android は，スマートフォンやタブレットなどの情報端末を主なターゲットとするプラットフォームとして開発され，主要な部分は無償で使用可能なオープンソースとして公開されています．現在は世界中のあらゆる情報端末や家電製品で利用され，Android は世界中で多くの人々に日々利用されています．

今回のチュートリアルではこの Android の端末で動作するアプリを作成する方法について解説していきます．お手持ちのパソコンを使ってアプリの開発にぜひチャレンジしてみましょう．

アプリの開発準備

アプリの開発について

Android のアプリは，Java や HTML，C＋＋，C#，そして Kotlin が開発言語として正式に採用されるなど，さまざまな言語やプラットフォームで開発することができます．ゲームのアプリを作るなら Unity や Cocos2d-x，iOS と Android の両方でアプリを作成する場合には Xamarin や Apache cordova など，作成するアプリの種類や要件を考え，どの言語やフレームワークを使用して開発するかを選定する必要があります．開発言語の選択肢が多いのも Android アプリの特徴です．

統合開発環境のインストール

Android アプリの開発では，無償で公開されている統合開発環境「Android Studio」を利用するのが一般的です．

Android Studio は，JetBrains 社が開発した非常に便利で高性能な Java 言語向けの統合開発環境「IntelliJ IDEA」を Android の開発に特化させたもので，Windows 版だけではなく mac OS 版，Linux 版も準備されています．

今回のチュートリアルでは，Windows 10 の環境下で解説を進めていきますが，ほかの OS でもほぼ同じ手順でインストールすることができますので安心してください．

まずは公式サイト[☆1]よりお使いのパソコンの OS にあったものダウンロードしてください．執筆時点での最新版は，バージョン 2.3.3 となっています．

ダウンロードしたファイルを実行し，ウィザードに従って初期設定のままインストールを行ってください．ダウンロード・インストールには少し時間がかかりますので注意してください．

インストールが完了すると，Android Studio の設定をインポートするかどうかのダイアログが表示されます．初めて Android Studio を使用される場合は，I do not have a previous version of Studio or I do not want to import my settings を選択してください．

次に Android アプリのコンパイルやビルドなどを行う Android SDK のインストールが始まります．Welcome 画面が表示されますので，Next で次へ進みます．

次に Android SDK のインストール方法が表示されます．Standard で推奨される Android SDK の環境をダウンロードします．

☆1 https://developer.android.com/studio/index.html

Android の SDK は，細かな単位でインストール制御をすることが可能です．それぞれの Android のバージョンのイメージ以外にも，Android の端末を開発機器として認識させる USB ドライバや，古いバージョンの端末に新しいバージョンの機能を追加するサポートライブラリなどもここからインストールすることができます．

ダウンロードするコンポーネントの一覧が表示されますので，Finish を押してダウンロードを開始します．

しばらくすると，インストールが完了し，Android のプロジェクトを作成できる状態となりました（図 -1）．

いよいよ次の章から，Android のプロジェクトを作成していきます．

プロジェクトの新規作成

Android のアプリは，プロジェクトの単位でアプリを開発します．Start a new Android Studio project より，新たなプロジェクトを作成します．

まずは以下の値を入力してください．

項目名	値
Application name	Android tutorial
Company domain	example.ipsj.or.jp

Application name には作成アプリの名前，Company domain には，アプリを開発する会社や組織のドメインを指定し，逆順に並べたものをそれぞれのアプリに Package name として固有の値を持たせます．Google play でアプリを公開する場合には，Package name が重複していると新規に登録することができませんので，慎重に設定しておく必要があります．

次に開発するアプリの対象を選択する画面となります．今回は何も変更せず，Phone and tablet の Minimum SDK を API 15（Android 4.0.3 IceCreamSandwich）とします．Android は，API Level として整数値がそれぞれのバージョンで割り振られており，その数値の大小で動作するバージョンかどうかを判定しています．今回の指定では，API 15（Android 4.0.3）以上の端末で動作することを意味しています．

Next を押すと，プロジェクトの雛形が自動で生成されます．

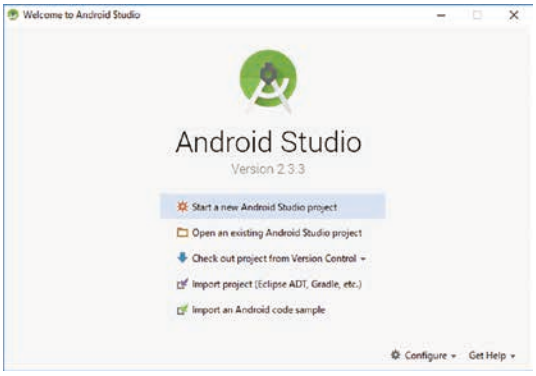
次の画面で，作成するアプリの画面パターンを選択する画面が表示されます．アプリで使用する画面と近いものを選択すると，Java のソースコードおよび XML（Extensible Markup Language）で記述されたレイアウトファイルが自動生成されますので非常に便利な機能です．

今回は Empty Activity からアプリを作成していきます（図 -2）．

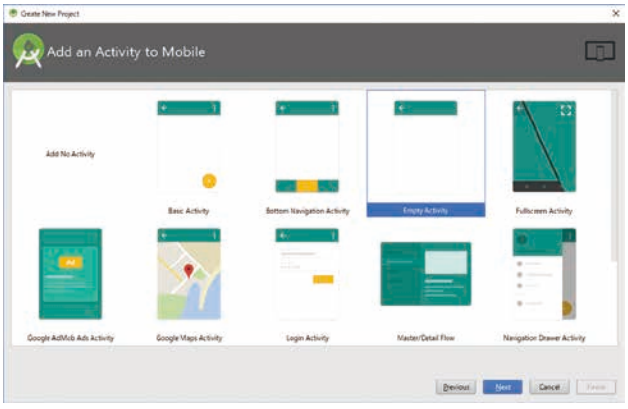
次のステップは Activity とレイアウトファイルの名称を入力する画面となります．

以下の値を入力し，Finish を押すとプロジェクトが生成されます．

Activity Name	MainActivity
Layout Name	activity_main



■図 -1　Android Studio スタート画面



■図 -2　Empty Activity

生成が終わると，Android 上で実行可能な Hello world が表示されるアプリとなっています．

プロジェクトの構造

Android Studio は，素早く開発を行うためにあらかじめ多くのディレクトリとファイルを自動的に生成します．実際のプロジェクト内のフォルダ構成は，細かく階層の深い構造をしていますが，Android Studio の左側にある Project ビューでは，シンプルでより扱いやすい構造で表示されます．

プロジェクトでは，モジュールという単位でアプリやライブラリが管理されており，自動で作成したプロジェクト内には，app というモジュールが作成されています．あらたにモジュールを追加することで，複数のアプリを作成することや，複数のアプリから参照される機能をまとめたライブラリを追加することも可能です．

名称	概要
app/manifests	アプリの内部の構造や，使用する機能など，基本的な情報を定義するファイルです．
app/java	Java のソースコードを格納するディレクトリです．アプリの処理はツリーでは表示がない main，ユニットテストは androidTest や test にソースコードを追加していきます．
app/res	アプリで使用する画像や文字を格納するディレクトリです．リソースの種類によって格納するディレクトリが決まっています．
app/res/drawable	アプリで使用する画像を格納します．
app/res/layout	画面のレイアウトを定義する XML ファイルを格納します．
app/res/mipmap	Android4.2 から利用できる新しい画像の表示方法です．拡大縮小のアニメーション時にも最適なビットマップを使用してくれる機構が備わっています．
app/res/values	アプリで使用する文字列，サイズ，色などを XML で定義するディレクトリです．
Gradle Scripts	アプリをビルドするための各種設定が定義されています．build.gradle はプロジェクト全体とモジュールに用意されており，モジュールの build.gradle にてバージョン情報や依存するライブラリなどを定義します．

エミュレータの作成

Android の SDK には，開発するパソコン内に仮想の Android 端末を作成するエミュレータの機能があります．Android の端末を準備できない場合には，エミュレータでアプリを動作させ，実機と同様にアプリの検証することができます．

まずは Android Studio の上部ツールバーにある，AVD Manager を起動します．左下の Create Virtual Device ボタンを押すと，さまざまな機種の画面サイズが表示されますので，任意のサイズを選択してください．

次に，Android の OS バージョンを選択する画面が表示されます．今回のプロジェクトでは API 15 が最小のバージョンとなっていますので，そのバージョン以上の OS を選択します．

最後に，作成したエミュレータに対して任意の名称をつけます．これで Android のエミュレータが作成されました．

アプリの実行

Android では Gradle と呼ばれるビルドシステムを使用して，アプリのビルドを行います．

アプリの実行は，コンパイルのエラーが出ていない状態で，画面上部のツールバーにある三角形の実行ボタンを押すと，Android のエミュレータや開発用のパソコンに接続された Android の一覧が表示され，どのデバイスで実行するかを選択すると，Gradle によるビルドが始まり，その後デバイスに転送されて画面上に Hello World! が表示されます．

サンプルアプリの作成

ここからは，先ほど作成したプロジェクトに処理を追加して，数字をカウントしていくアプリを作成していきます．

画面上に現在のカウント値を表示し，+1 ボタンを押すとカウント値をインクリメント，リセットボタンを押すと 0 に戻る機能を実装してみましょう（図 -3）．

レイアウトファイルの作成

Android では，XML を用いて View と呼ばれる画面の部品を並べて画面のレイアウトを作成します．Java のプログラムコードでも View を作成していくことはできますが，複雑な GUI（Graphical User Interface）を構成するとなるとコード量が増え，保守にも大変時間がかかりますので XML で作成する方法がおすすめです．

さまざまな画面の解像度が存在する Android では，座標の指定をした絶対値での View の配置や，フォントサイズを指定することは推奨されていません．相対的な値や指定方法を用いて柔軟なレイアウトを作成する必要があります．

まずは左側の Project ビューより，res/layout/main.xml を開いてみましょう．View には，ボタンやテキストを入力する EditText，ドロップダウンのリストを表示する Spinner，など一般的なソフトウェアで使用される部品が用意されています．それらを Palette よりドラッグ・アンド・ドロップして画面上に配置する Design での編集モードか，XML を直接入力して編集する Text での編集モードを任意で選ぶことができます．

今回は，プロジェクト作成時に生成されたレイアウトファイルを編集して次のようなレイアウトファイルにします．

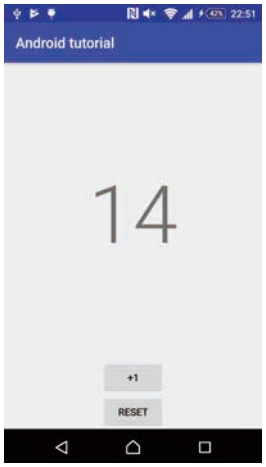
■ res/layout/activity_main.xml（参照 -1）

LinearLayout は，中に含まれる View を縦または横に一直線に並べるレイアウトです．orientation の属性を vertical にすると縦，horizontal にすると横に View を並べることができます．

今回のチュートリアルでのレイアウトは，LinearLayout の中に TextView と呼ばれるテキストを表示する View と，その名の通りのボタンを表示する Button とを縦に並べています．

ボタンや，カウント値を表示する TextView など，ソースコードから参照するものは，id の属性で View に ID をつけ，ソースコード上から参照するします．具体的な参照方法は，次節で詳しく説明します．

それぞれの View で指定されている layout_width および layout_height は，View の高さを指定する



■図 -3　サンプルアプリ

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    tools:context="jp.or.ipsj.example.MainActivity">

    <TextView
        android:id="@+id/text_count"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="center"
        android:textAppearance="@style/TextAppearance.AppCompat.Display4" />
    <Button
        android:id="@+id/button_increment"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/plus_one" />
    <Button
        android:id="@+id/button_reset"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/reset" />
</LinearLayout>
```

■参照 -1　res/layout/activity_main.xml

属性です．wrap_content を指定すると中の要素に合わせて最小のサイズ，match_parent を指定すると画面いっぱいに引き伸ばすことができます．しかし match_parent を指定してしまうと，定義した View が画面いっぱいに引き伸ばされ，以後の View が画面に入りきらない画面となってしまいます．そういった際には，layout_weight というパラメータを用いて，比率でレイアウトの幅や高さを指定することができます．

gravity の属性は，center/left/top/right/bottom などの値を指定して，中に含まれる View をどの位置に寄せるかを指定することができるものです．bottom|right のように複数の値をバーティカルバーで区切って指定をすることもできます．今回のアプリでは，center を指定して中の View を中央に寄せています．

textAppearance の属性は，あらかじめ用意された文字のスタイルを適応することで，アプリで使用されているフォントの統一感を出すことができます．

TextView や Button に設定されている text 属性は，それぞれの View に表示する文字列を指定するためのものです．直接文字列を指定することも可能ですが，res/values/strings.xml ファイルに文字列を定義し，それを参照する実装方法が一般的です．外部に文字列を集約して定義することにより，多くの言語に対応するアプリの作成を容易に実現することができます．

■ res/values/strings.xml（参照 -2）

ボタンのイベント処理を追加

Android では，Activity というクラスを継承することでアプリの画面を作成することができます．Activity は AndroidManifest と呼ばれるファイルにクラス名と起動条件を定義することで，Android のシステム上より呼び出しされるようになります．iOS のアプリとは異なり，複数のエンドポイントを持つアプリも作成することができます．

ここからはその Activity に処理を実装し，ボタンを押したときのイベントを実装していきます．

■ MainActivity.java（参照 -3）

Activity を継承したクラスを作成し，onCreate と呼ばれるメソッドをオーバライドすると，アプリの起動時にこのメソッドが呼び出されます．その時にどのような画面を表示するかを setContentView メソッドを使って指定します．

res ディレクトリに，レイアウトファイルや画像などを格納すると，Android のビルドシステムが R クラスというものを生成し，ファイル名やレイアウトで定義した ID の名称で定数が自動的に作成されますので，R. layout. activity_main のように，どのレイアウトを使うかを指定することができます．

ほかにも Activity には，onResume，onStart，onStop，onDestroy など，アプリの起動時や再開時，終了時などさまざまなタイミングで呼ばれるメソッドが準備されており，そのタイミングに処理が必要であれば，オーバライドして処理を実装することができるように設計されています．

レイアウトファイルで定義した View は，findViewById メソッドで ID を指定すると，View 型のオブジェクトとしてソースコード上に読み込むことができます．それを Button 型や TextView 型にキャストして，アプリで必要な処理を実装します．

今回は，このボタンが押されたときに処理を実装しますので，setOnClickListener メソッドで OnClickListener というインタフェースをセットします．OnClickListener をセットしておくと，ボタンが押され

```
<resources>
    <string name="app_name">Android tutorial</string>
    <string name="plus_one">+1</string>
    <string name="reset">Reset</string>
</resources>
```

■参照 -2　res/values/strings.xml

たときのイベントがonClickメソッドにコールバックされますので，このタイミングでボタンをカウントする処理やカウントをリセットする処理を実装します．

このように少しのソースコードを実装するだけで，簡単にアプリが開発できてしまうのがAndroidです．さらなる勉強をして，Androidのアプリを公開してみませんか？

(2017年10月29日受付)

石丸宗平　roborovskii.so@gmail.com
システムエンジニアとしてアプリの開発やシステムの開発に従事．また Code for などのシビックテックの活動にも積極的に参加している．

```
package jp.or.ipsj.example.androidtutorial;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {
   /** カウント値を表示するView */
   private TextView mCountView;
   /** カウント値 */
   private int mCount;

   /**
   * Activityが生成されるときに呼ばれるメソッド
    */
   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       /* XMLのレイアウトファイルを表示する */
       setContentView(R.layout.activity_main);

       /* レイアウトファイルのViewを参照する */
       mCountView = (TextView) findViewById(R.id.text_count);
       /* クリックされたときのコールバックを設定する */
       Button incrementButton= (Button) findViewById(R.id.button_increment);
       incrementButton. setOnClickListener(this);
       Button resetButton = (Button) findViewById(R.id.button_reset);
       resetButton.setOnClickListener(this);
   }

   /**
   * Viewがクリックされたときに呼ばれるメソッド
    */
   @Override
   public void onClick(View v) {
       int id = v.getId();
       switch (id) {
           case R.id.button_reset:
               mCount = 0;
               break;
           case R.id.button_increment:
               mCount++;
                break;
       }
       mCountView.setText(String.valueOf(mCount));
   }
}
```

■参照-3　MainActivity.java