# Android Wear アプリの作り方

hide92795
倉松つゆ

# はじめに

初めまして、サークル「hollyhock」の hide92795 と申します。今回はサークル「hollyhock」の本をご購入いただきありがとうございます。初めてのことばかりで色々と見づらいことがあるかと思いますが、ぜひ読んでいただければ幸いです。

# Android Wear とは

Android OS をベースにしたスマートウォッチ用の OS です。日本ではサムスンから Gear Live、LG から LG G Watch、ソニーから SmartWatch 3 などが発売 / 発売予定となっています。基本的な使い方としては、Bluetooth で連携した Android 端末とデータをやり取りし通知を表示させる、Android Wear 用にアプリを作成し、Android Wear 上で実行させるといったことが出来ます。ディスプレイの大きさが 1.5 インチ程度と非常に小さいため、一画面に表示できる情報量は少ないですが、わざわざスマホを開かずとも情報を見ることができます。今回は、Android Wear 用のアプリを作ってみて感じたことや、開発環境の事情などについて書いていこうと思います。

# Android Wear の開発環境

今まで、Android アプリを製作するときには、eclipse に ADT（Android Development Tools）をプラグインとしてインストールして使用する方法が一般的でした。しかし、Google が開発を進めている Android Studio が 2014/12/9 に正式版となり、eclipse ユーザーへの移行ガイドが用意されるなど Android Studio への移行が徐々に進められています。さらに、Android Wear 用のアプリを製作するためには Android Studio がほぼ必須と言っても過言ではない程依存するようになっています。


Android Studio 1.0 のスプラッシュ画面

eclipse から Android Studio に移行する上で一番戸惑うところはおそらくビルドツールの変更だと思います。eclipse では ant を用いてビルドや apk の作成を行うことが一般的でした。しかし、Android Studio では新しく Gradle によるビルドシステムがデフォルトで使用されることとなります。

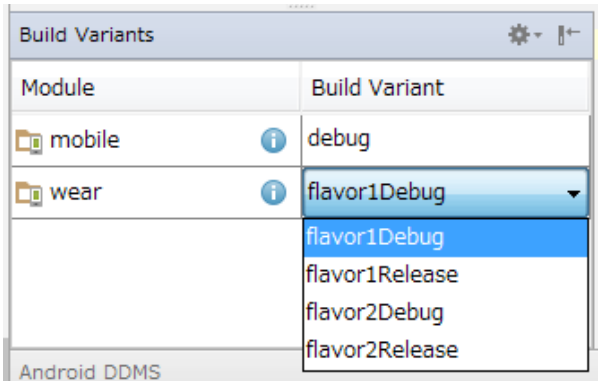
私も初めは Gradle への移行はあまり乗り気ではありませんでしたが、いざ使ってみると Android Studio ＋ Gradle という組み合わせがとても便利に感じました。Gradle でビルドスクリプトや依存関係などを記述するいわば ant の「build.xml」と同じようなものである「build.gradle」を見ながら、どのようなところで Gradle が便利になっているか見ていきましょう。

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 21
    buildToolsVersion "21.1.2"

    defaultConfig {
        applicationId "org.hide92795.android.wear.hoge"
        minSdkVersion 20
        targetSdkVersion 21
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.google.android.support:wearable:1.1.0'
    compile 'com.google.android.gms:play-services-wearable:6.5.87'
}
```

デフォルト状態で生成される build.gradle

１つ目はアプリの依存関係を楽に記述できるという点です。eclipse を使用していた時は、サポートライブラリを使用するのにサポートライブラリのプロジェクトをインポートして、アプリ内でリンクさせるといった作業が必要でした。さらに、このインポートしたサポートライブラリのプロジェクトがよくエラーを発生させ、アプリ内での謎のエラーの原因になることがしばしばありました。一方、Gradle では dependencies 内に指定の記述方法で書けば自動的にリンクを行ってくれますし、Maven Repository 内のライブラリもそこに記述を行うことで自動的にダウンロードやリンク作業を行ってくれます。画像内では、プロジェクト内の libs フォルダ以下にある jar ファイルと、WearableSupport ライブラリ、Wearable 用の Play Service ライブラリの３つを要求している事になります。Maven Repository からライブラリを参照するのであれば、それぞれのページに Gradle 記述用の文字列が掲載されているので、それをコピペするだけで取り込むことが出来ます。

２つ目は、Build Type と Flavor による使用リソースの切り替え機能です。build.gradle 内に記述を行うことで、複数種類のビルド環境を構築することが出来ます。例としては、デバッグ作業時にはデバッグ用の string.xml を使用する (res 以下の置き換え ) や、パッケージ名の最後に特定の文字を付加させる (AndroidManifest.xml の置き換え )、定数値の切り替え (src 以下の置き換え ) といった事ができるようになります。Build Type と Flavor の切り替えは Android Studio 上で簡単に切り替えることができます。



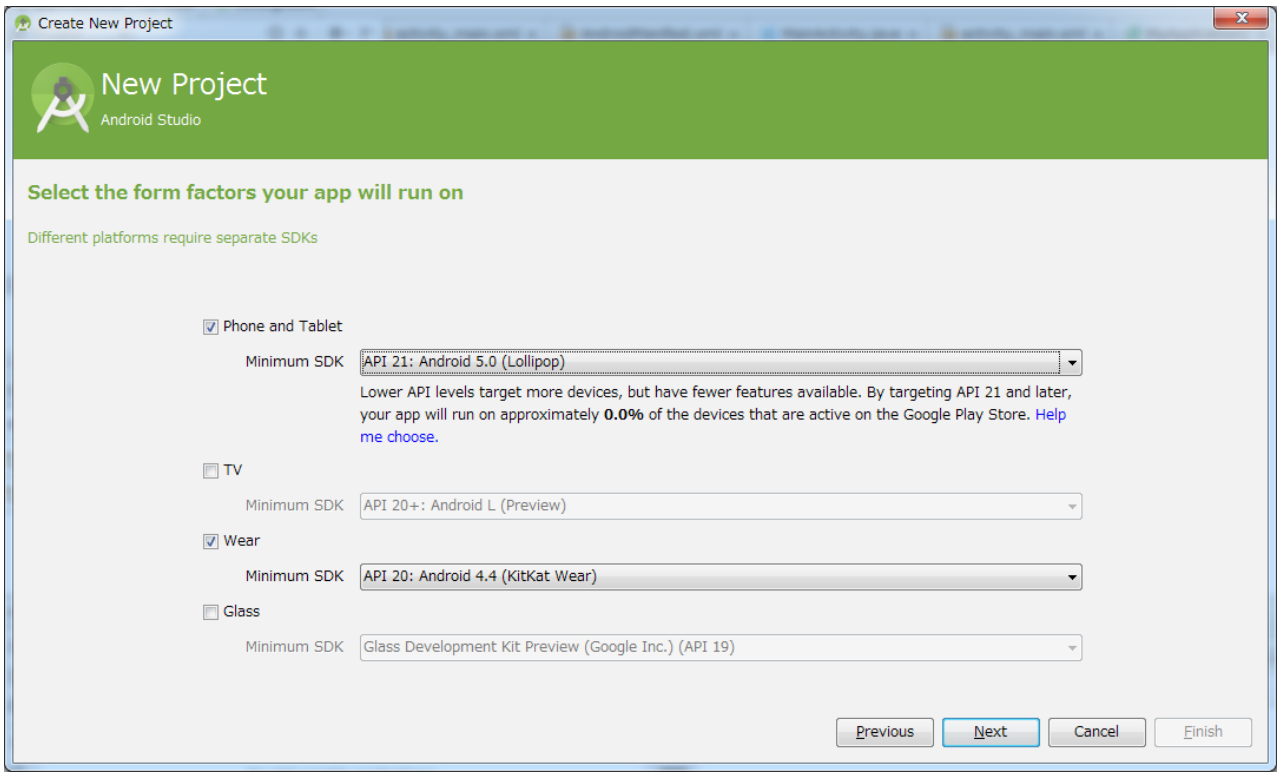
Flavor に「flavor1」と「flavor2」を追加した状態

このBuild TypeとFlavorによって、今まではデバッグする時のログの出力を調節するのにantで文字列置換を行う、もしくは手動で書き換えるといったことが必要だったものがBuild Typeで使用するソースコードを切り替えることができるのでとても便利になりました。

他にeclipseからAndroid Studioに乗り換えて便利になった点としては、レイアウトエディタの安定化という点があります。eclipseの時には5回に1回ほどの頻度でレイアウトエディタがぶっ壊れてしまい、再起動を余儀なくされていました。今のところAndroid Studioではレイアウトエディタに関する不満は時にありません。

# Android Wearでのアプリ開発

2014/12/25現在、Android Wearの最新バージョンは5.0.1になっています。Android Studioでは、プロジェクト作成時にAndroid Wearアプリを含めるかどうかを選ぶことができます。



プロジェクト作成画面

２つプロジェクトを作成し、手動でパッケージングすることもできますが、せっかくの機能なのでどんどん使っていきましょう。複数の同一プロジェクト内に２つの API レベルの違うアプリを組み込むことができるのが Android Studio の便利な点の 1 つです。

Phone and Tablet と Wear にチェックを入れた状態でプロジェクトを作成するとプロジェクトディレクトリ内に「mobile」ディレクトリと「wear」ディレクトリが生成されます。

それぞれのディレクトリ内にスマホ本体用のアプリと Android Wear 用のアプリの２種類を作成していくことになります。次に Wear 用アプリにデフォルトで作成された MainActivity.java の中身を見ていきましょう。

```
package org.hide92795.android.wear.hoge;

import android.app.Activity;
import android.os.Bundle;
import android.support.wearable.view.WatchViewStub;
import android.widget.TextView;

public class MainActivity extends Activity {
    private TextView mTextView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        fi nal WatchViewStub stub = (WatchViewStub) fi ndViewById(R.id.watch_view_stub);
        stub.setOnLayoutInfl atedListener(new WatchViewStub.OnLayoutInfl atedListener() {
            @Override
            public void onLayoutInfl ated(WatchViewStub stub) {
                mTextView = (TextView) stub.fi ndViewById(R.id.text);
            }
        });
    }
}
```

MainActivity.java

基本的な部分としては、スーパークラスが Activity であり、onCreate 内で setContentView をしているなど、普通の Android アプリと大差がありません。変わっているところは、onCreate 内で WatchViewStub に対して setOnLayoutInflatedListener を呼んでいるところでしょうか。Android Wear を搭載した端末には、LG G Watch などの四角いディスプレイを持つものと、Moto 360 のように丸型のディスプレイを持つものの２通りがあります。ディスプレイの形が変わるので、当然レイアウトもそれぞれの形に調節してあげる必要があります。この四角と丸型でレイアウトを切り替えるために使用するのが WatchViewStub です。ここで、MainActivity にセットされているレイアウトファイル「activity_main.xml」の中身を見てみましょう。

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.wearable.view.WatchViewStub
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/watch_view_stub"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:rectLayout="@layout/rect_activity_main"
    app:roundLayout="@layout/round_activity_main"
    tools:context=".MainActivity"
    tools:deviceIds="wear">
</android.support.wearable.view.WatchViewStub>
```

activity_main.xml

見て分かる通り、このレイアウトファイルには id が「watch_view_stub」の WatchViewStub が 1 つしか追加されていません。この WatchViewStub に付けられている

```
app:rectLayout="@layout/rect_activity_main"
```

と

```
app:roundLayout="@layout/round_activity_main"
```

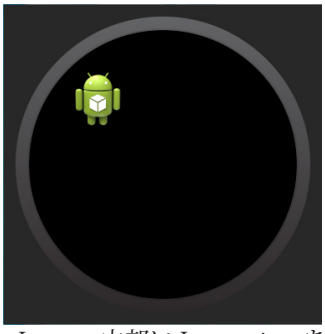
の２つの要素によってレイアウトを切り替えています。名前の通り、「app:rectLayout」が四角のディスプレイ用のレイアウト、「app:roundLayout」が丸型のディスプレイ用のレイアウトになります。この WatchViewStub ですが、判定は四角か丸かでしか判別ができないため、デバイスごとの細かな違い（例えば、Moto360 はセンサー類を搭載するために画面下が少し削られており、そこの部分には表示ができない）を考慮にいれるならば、ソースコードから切り替えを行うほうが実用的とも考えられます。

# Wearable UI Library

Android Wear ではディスプレイのサイズが小さくなったことにより、Google 側から幾つかの UI パーツ (Wearable UI Library) が提供されています。ここでは、それらのうちのいくつかについて紹介していきたいと思います。

## BoxInsetLayout

中に配置する View が、デバイスの形状によって隠れないように配置をしてくれるレイアウトです。丸型ディスプレイの時に内部の正方形状のエリアに View が収まるように配置を行います。



BoxInsetLayout 内部に Imageview を配置

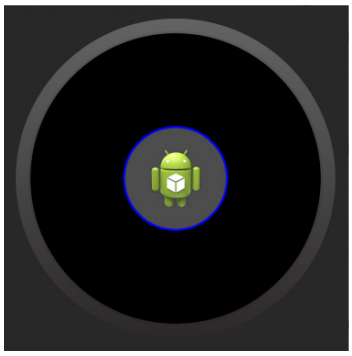
中に配置した View のどの面が画面外に出ないようにするかは、子 View の layout_box 要素に「left」「right」「top」「bottom」「all」で指定することが出来ます。画像のように左上側がはみ出ないようにするためには、「left|top」と指定する必要があります。

```
<android.support.wearable.view.BoxInsetLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_launcher"
        app:layout_box="left|top"/>

</android.support.wearable.view.BoxInsetLayout>
```

BoxInsetLayout 使用例

## CircledImageView

背景を円形に指定できる ImageView です。背景の円の色や大きさ、枠線を指定することができます。



```
<android.support.wearable.view.CircledImageView
  android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:id="@+id/view"
  android:layout_centerVertical="true"
  android:layout_centerHorizontal="true"
  android:src="@drawable/ic_launcher"
   app:circle_color="#404040"
   app:circle_border_color="#0000ff "
   app:circle_border_width="3dp"
   app:circle_radius="40dp"/>
```

CircledImageView 使用例

## ConfirmationActivity

ユーザーからの何らかのアクションに対して、アクションの実行の完了・失敗を表示する Activity です。表示するアニメーションは、この ConfirmationActivity を表示するための Intent に対してパラメータを設定するだけです。EXTRA_ANIMATION_TYPE に対しては SUCCESS_ANIMATION（成功のアニメーション）、FAILURE_ANIMATION（失敗のアニメーション）,OPEN_ON_PHONE_ANIMATION（携帯で開いたことを表すアニメーション）、EXTRA_MESSAGE に対して文字列を設定することでカスタムメッセージを表示できます。

Activity を表示する形になるので、AndroidManifest.xml に対して

```
<activity android:name="android.support.wearable.activity.Confi rmationActivity"/>
```

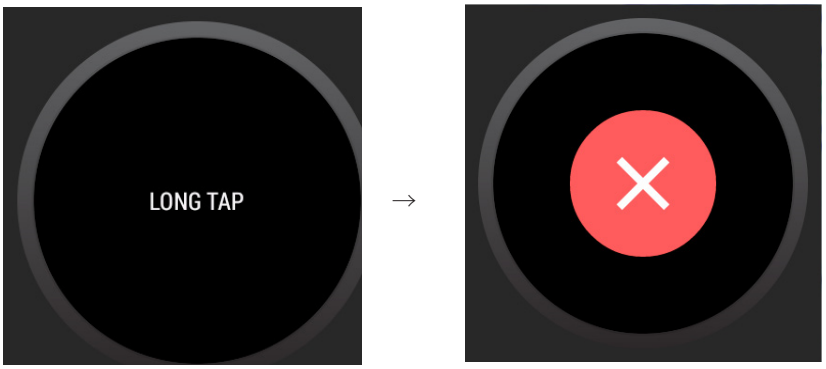
このように追加する必要があります。


```
Intent intent = new Intent(this, Confi rmationActivity.class);
intent.putExtra(Confi rmationActivity.EXTRA_ANIMATION_TYPE,
          Confi rmationActivity.OPEN_ON_PHONE_ANIMATION);
intent.putExtra(Confi rmationActivity.EXTRA_MESSAGE, "MESSAGE");
startActivity(intent);
```

ConfirmationActivity 使用例

## DismissOverlayView

long-press-to-dismiss を実装した View です。ロングタップを検知し、×ボタンを全画面に表示、もし×ボタンがタップされれば表示されていた Activity を終了します。DismissOverlayView は初めは透明な View のため、全画面を覆うようにレイアウト上に配置します。あくまで DismissOverlayView は×ボタンの表示と Activity の終了しか行わないので、DismissOverlayView の表示までは GestureDetector を使用する必要があります。



```
public class MainActivity extends Activity {
    private GestureDetectorCompat gesture_detector;
    private DismissOverlayView dismiss_overlay_view;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        fi nal WatchViewStub stub = (WatchViewStub) fi ndViewById(R.id.watch_view_stub);
        stub.setOnLayoutInfl atedListener(new WatchViewStub.OnLayoutInfl atedListener() {
            @Override
            public void onLayoutInfl ated(WatchViewStub stub) {
                gesture_detector = new GestureDetectorCompat(MainActivity.this, new LongPressListener());
                dismiss_overlay_view = (DismissOverlayView) fi ndViewById(R.id.dismiss_overlay_view);
            }
        });
    }

    @Override
    public boolean dispatchTouchEvent(MotionEvent event) {
        return gesture_detector.onTouchEvent(event) || super.dispatchTouchEvent(event);
    }

    private class LongPressListener extends GestureDetector.SimpleOnGestureListener {
        @Override
        public void onLongPress(MotionEvent event) {
            dismiss_overlay_view.show();
        }
    }
}
```

DismissOverlayView 使用例

DismissOverlayView を配置するときには、全画面かつ最前面に配置する必要があります。

# あとがき

ここまで読んでいただきありがとうございます。サークル「hollyhock」の hide92795 です。
初めてのコミケ、初めてのサークル活動、わからないことが多くありましたが、なんとかこの日まで持ってくることができました。
本当は本の形でコミケに臨みたかったのですが、まだわからないことが多すぎて外部に発注するのはやめ、自宅でも制作できる CD 収録という形になってしまいました。
また、内容的にももっと書きたいことがたくさんあったのですが、体調不良で筆が進まなかったこともあり、すべてを書き切ることは出来ませんでした。
今後の活動では、もっと様々なことをやってきたいと考えています。
また次の機会に出会えたら嬉しいです。

2014/12　hide92795

こんにちは！はじめまして、倉松つゆです。

初めてイラストを描かせていただきました！
本当はもう少しボリュームがあるはずだったのですが
勝手ながら私の体調の都合上、このような形になってしまいました…

新しいガジェットは面白いものですよね！
いつかだれしもウェアラブルデバイスを身に着ける時代が来るのでしょうかｖｖ

またお会いできることを願って☆

倉松つゆ 2014 年 12 月 ( 現在 9 度 8 分 )