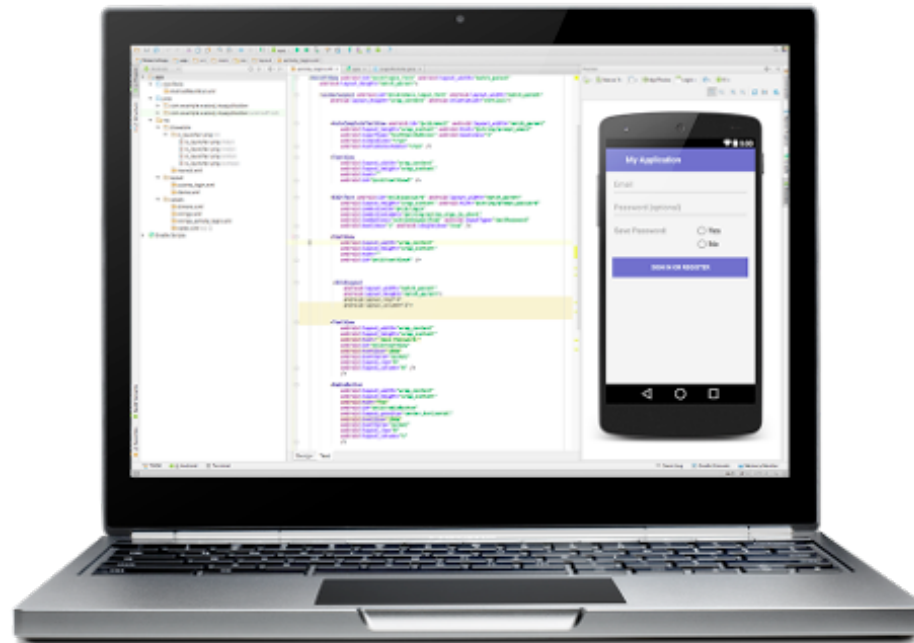


Android入門講座

はじめてのアプリ作成

2018年8月25日



アプリ作成に必要なファイルの準備

DVDディスク、またはUSBメモリから取得したファイルを、Windows PC（または Mac）の任意の場所（たとえば、デスクトップに作成した「work」フォルダ）に配置します。

今回利用するファイルは以下のとおりです。

- ・ **android-studio-ide-173.4907809-windows.exe**

Android アプリを開発するためのツールである、Android Studio のインストーラファイル（Windows 64bit 用）

- ・ **android-studio-ide-173.4907809-windows32.zip**

Android Studio（Windows 32bit 用）

- ・ **android-studio-ide-173.4907809-mac.dmg**

Android Studio のインストーラファイル（Mac 用）

- ・ **UniversalAdbDriverSetup.msi**

Windows PC で Android 実機を接続し、認識できるようにするためのツール

- ・ **lesson**

当講習会で開発するアプリのソースコードを格納しているフォルダ

- ・ **Android入門講座（はじめてのアプリ作成）.pdf**

講習会前半で利用する資料（当資料）

- ・ **Android入門講座（センサーを活用したアプリ作成）.pdf**

講習会前半で利用する資料

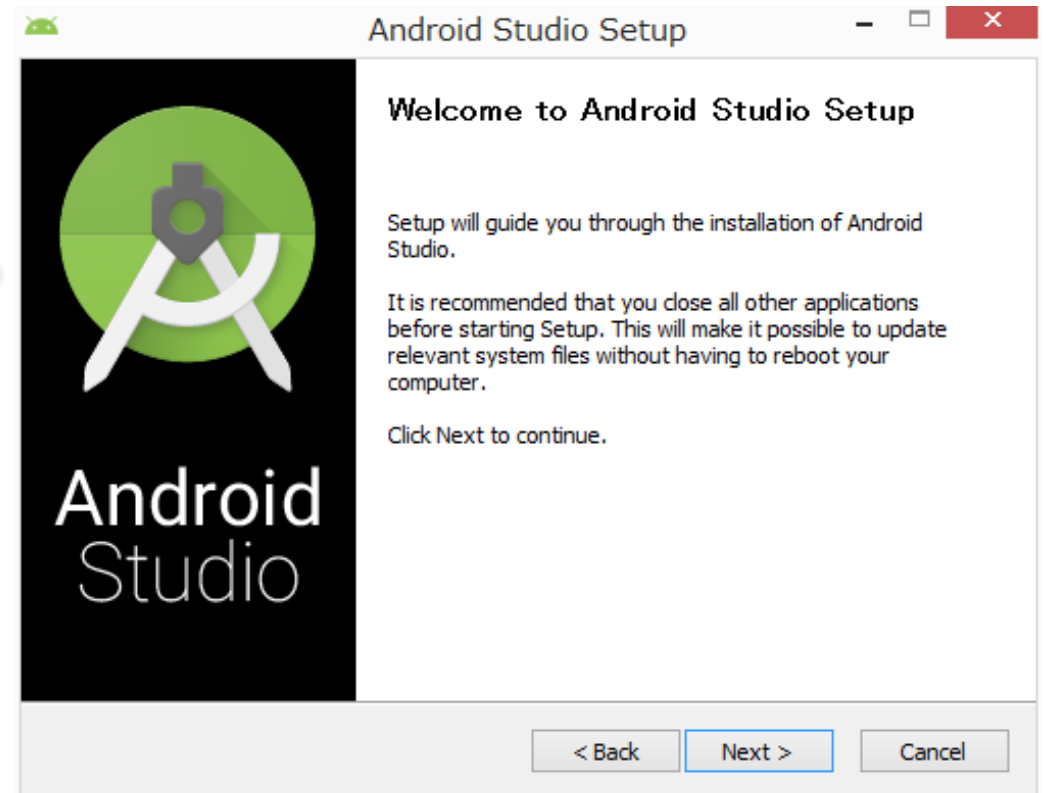
Android Studio のインストール (Windows 64bit版の場合。その1)

「android-studio-ide-173.4907809-windows.exe」

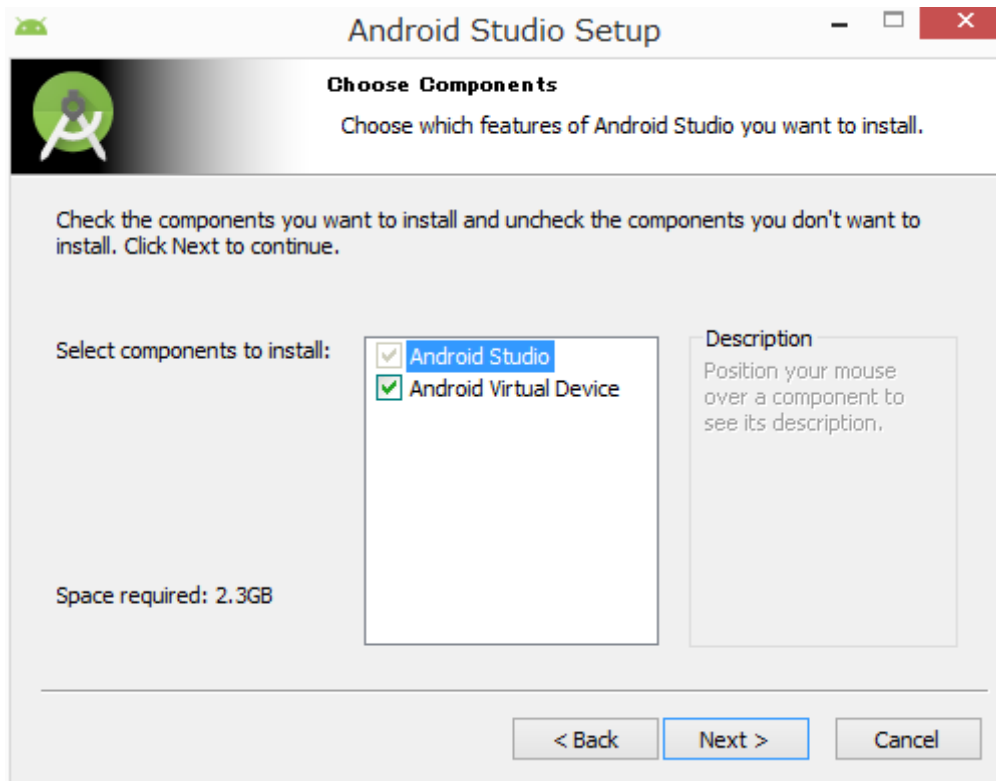
をダブルクリックします。

インストールには管理者権限が必要となります。

そのまま「Next」をクリックします。

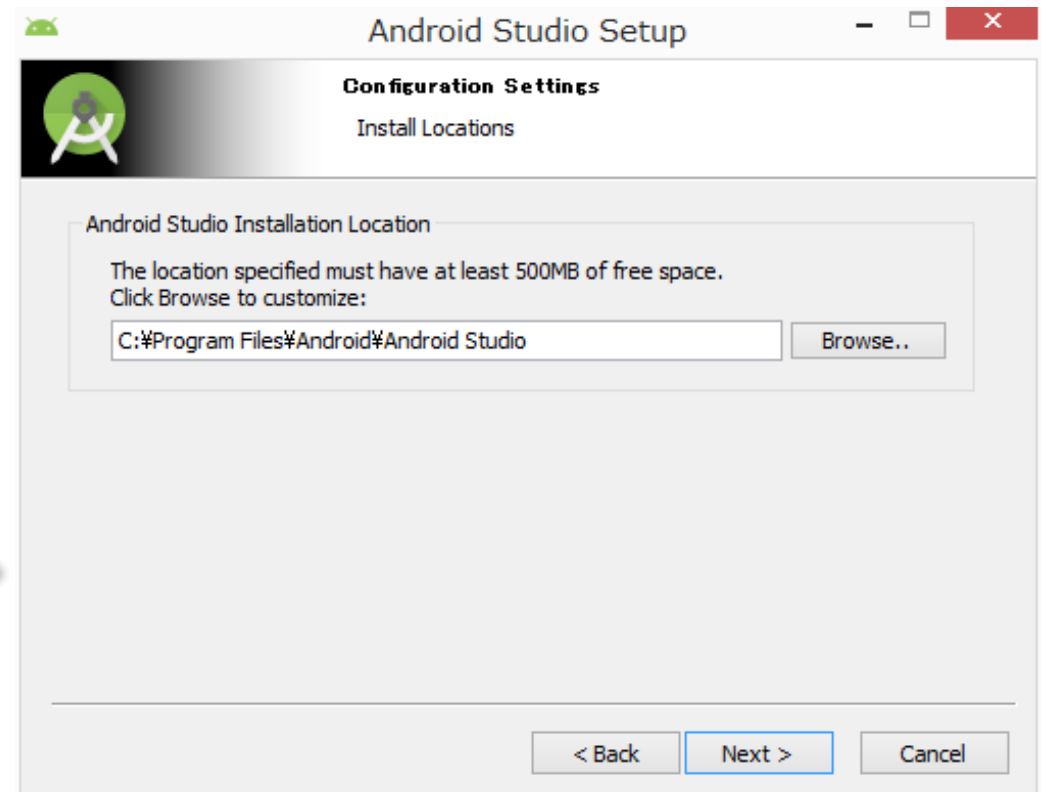


Android Studio のインストール (Windows 64bit版の場合。その2)

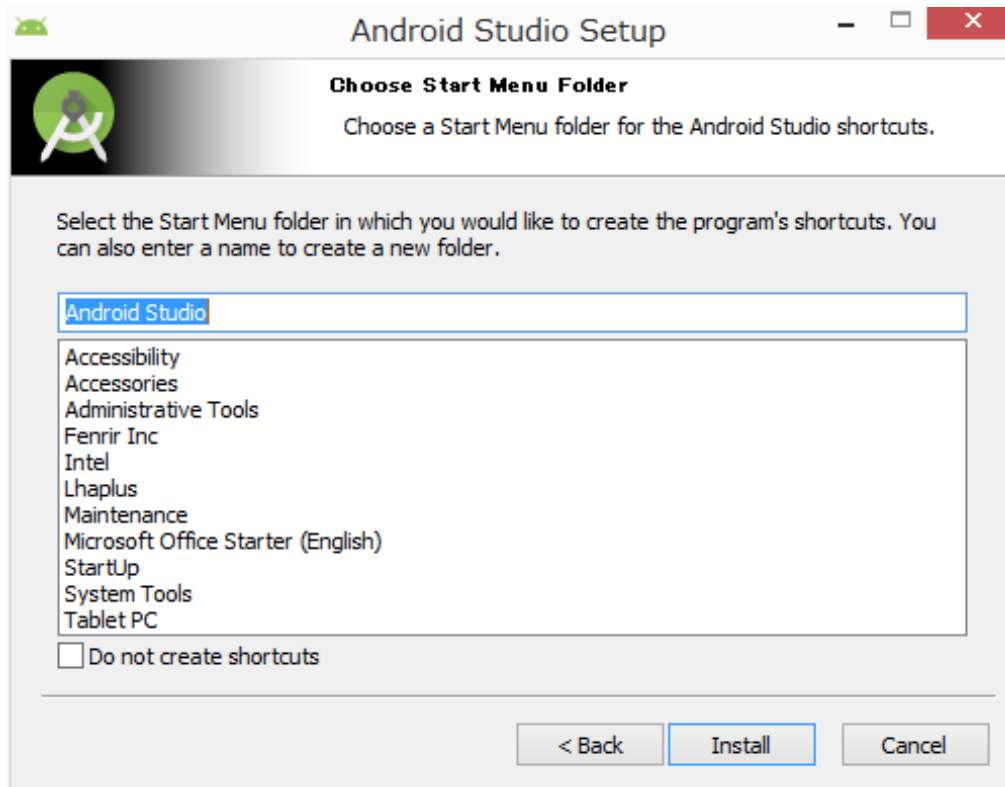


「Android Virtual Device」は、今回は利用しないのでチェックをはずしてもよいでしょう。その後、「Next」をクリックします。

そのまま「Next」をクリックします。

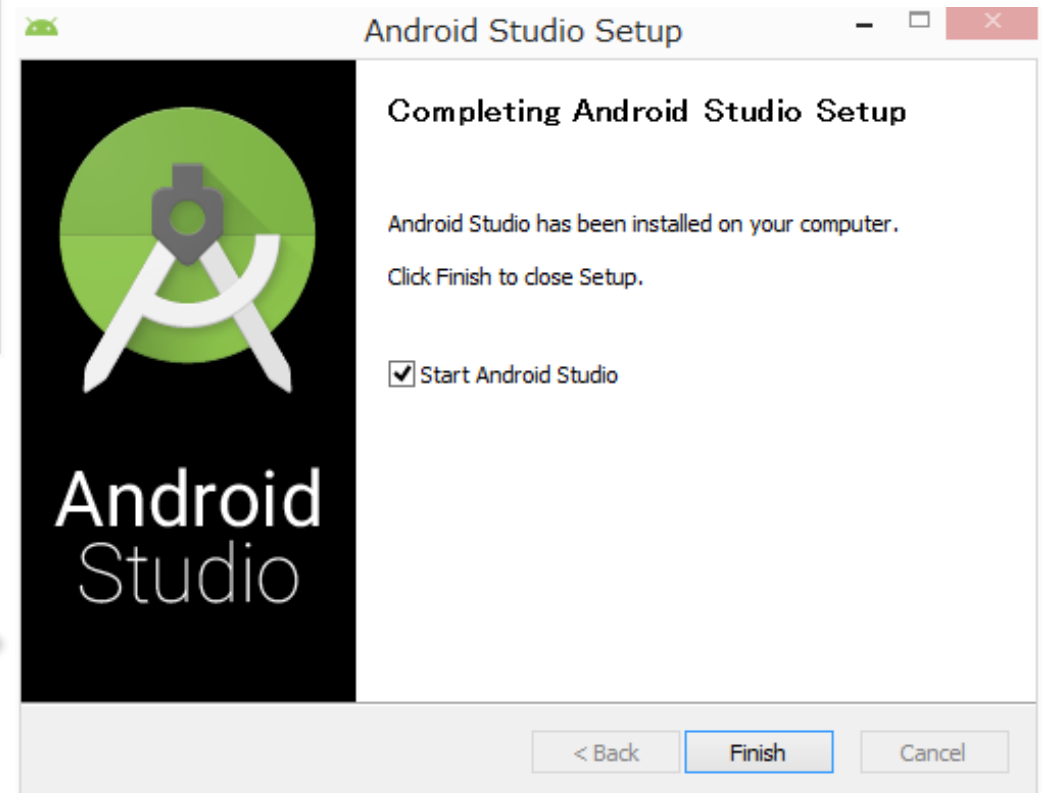


Android Studio のインストール (Windows 64bit版の場合。その3)



そのまま「Install」をクリックします。

しばらくしてインストールが完了したのち、
「Finish」をクリックして、Android Studio を起動します。

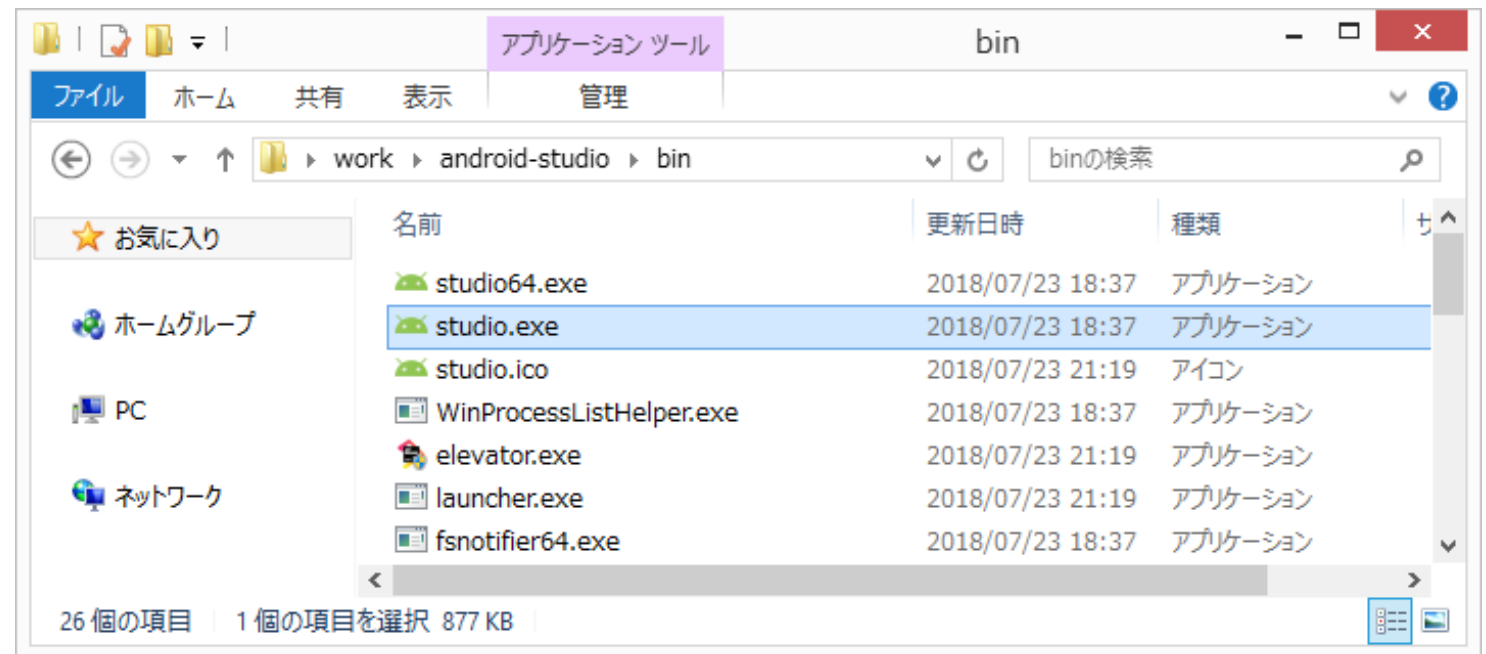


Android Studio のインストール (Windows 32bit版の場合。)

「android-studio-ide-173.4907809-windows32.zip」

を任意の場所（デスクトップの「work」フォルダ内など）に展開します。

展開されたフォルダ内の「**android-studio¥bin¥studio.exe**」をクリックして、Android Studio を起動します。



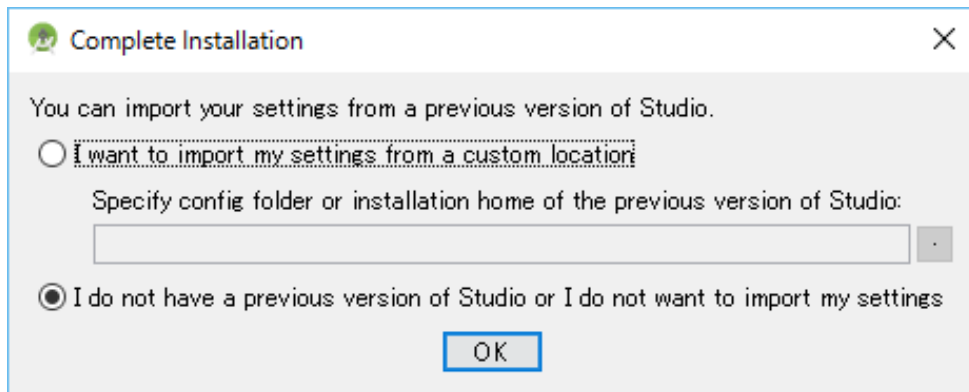
Android Studio のインストール (Mac の場合。)

「**android-studio-ide-173.4907809-mac.dmg**」

をダブルクリックします。

インストールの手順は、Windows 64bit 版の場合と同様です。

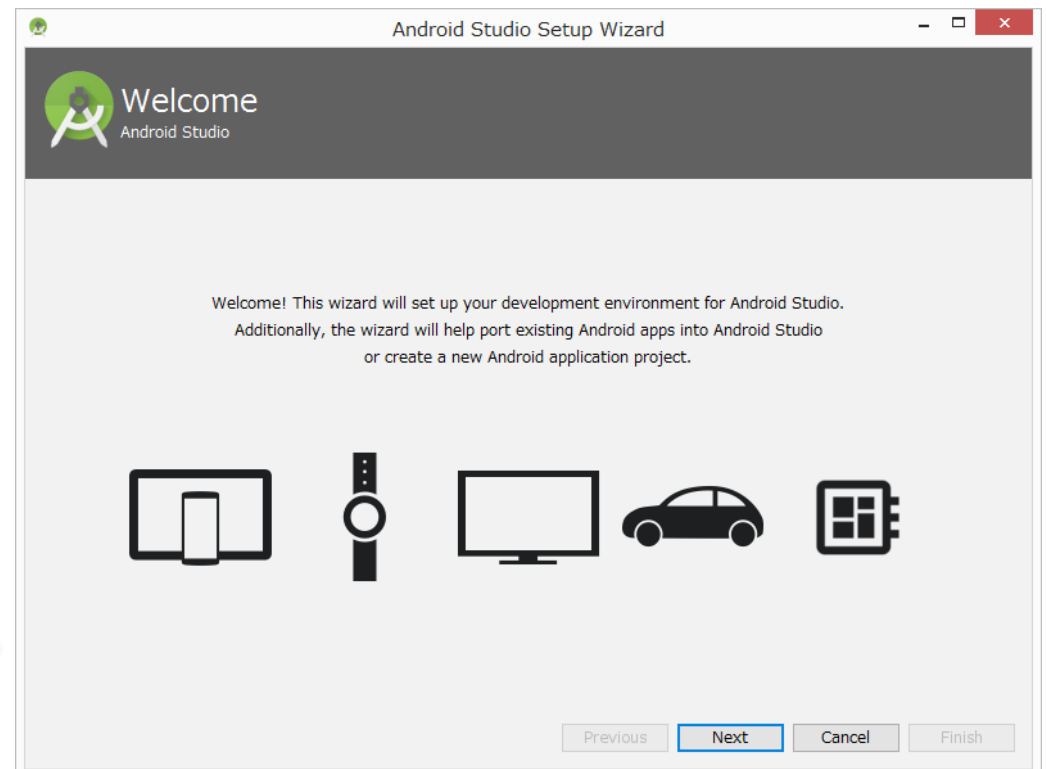
Android Studio の起動（その1）



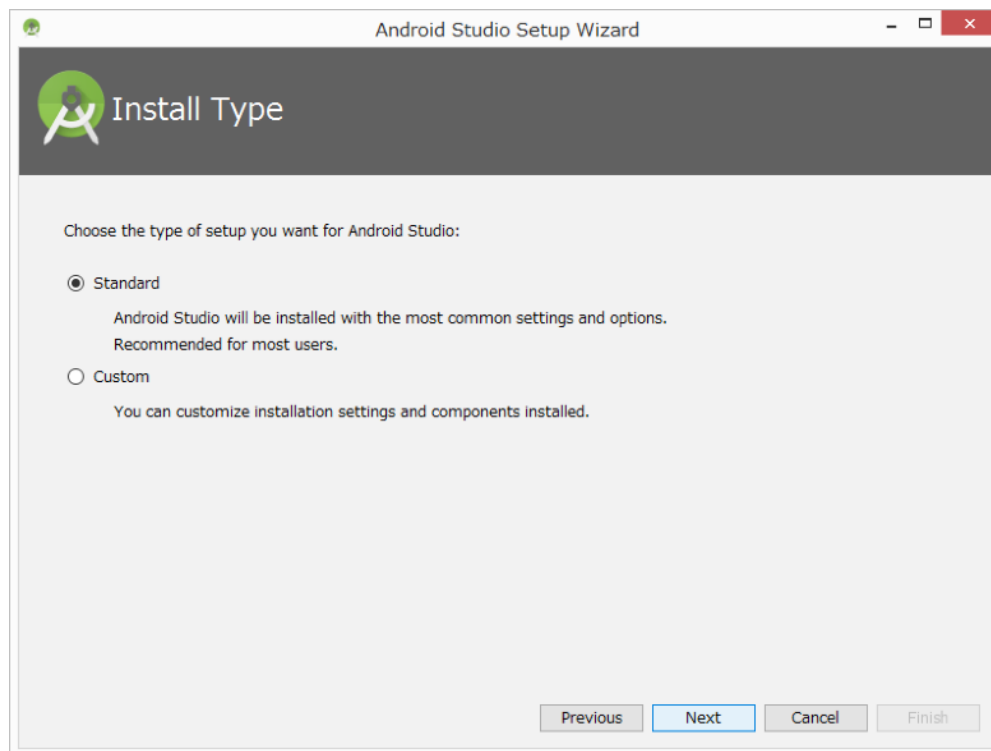
Android Studio の初回起動時に、このようなダイアログボックスが表示されましたら、今回は下側の

「I do not have a previous . . .」
をチェックして「OK」ボタンをクリックします。

そのまま「Next」をクリックします。

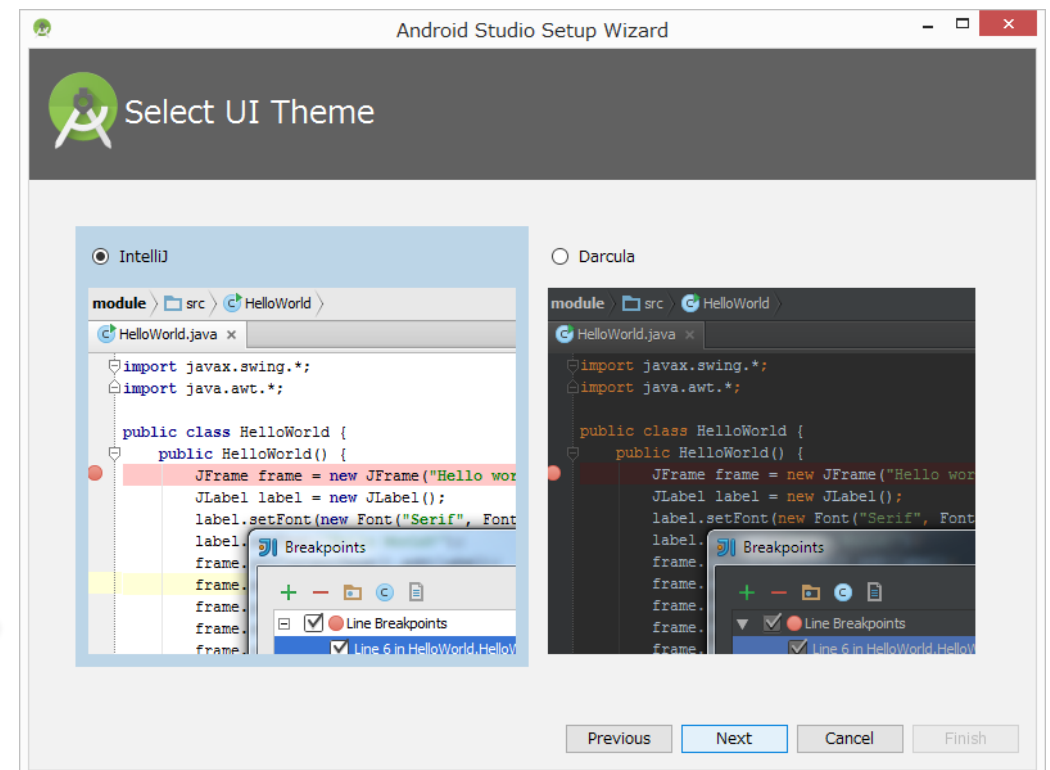


Android Studio の起動 (その2)

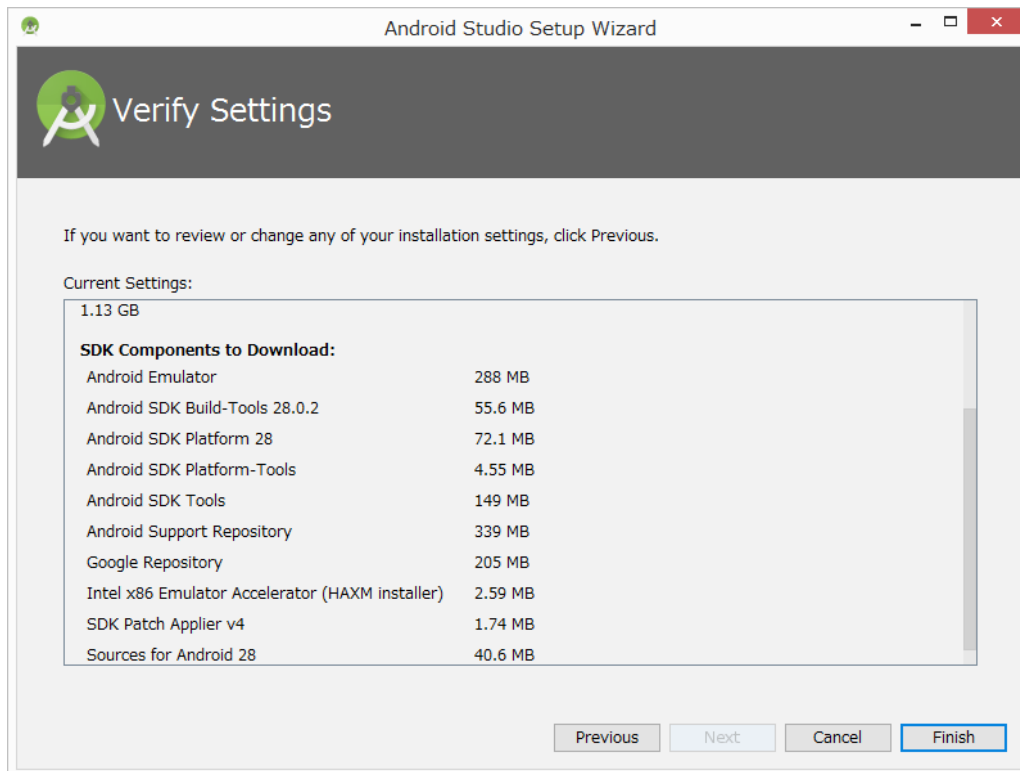


そのまま「Next」をクリックします。

UI テーマはお好みで選択して、
「Next」をクリックします。

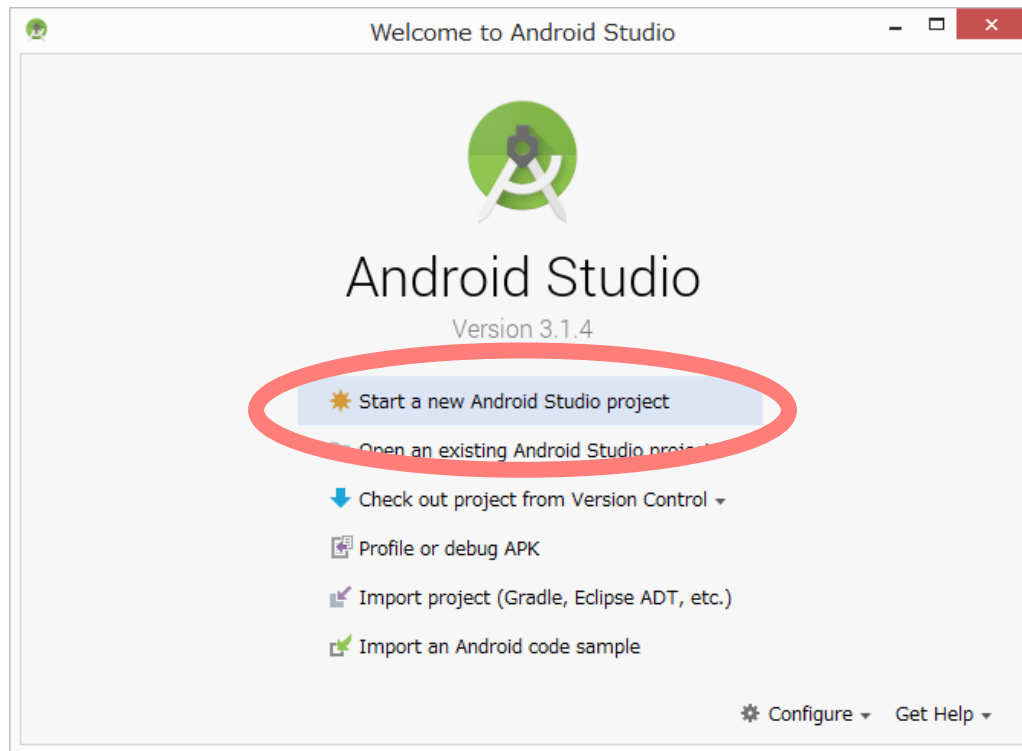


Android Studio の起動 (その3)



「Finish」をクリックして、初期設定処理を完了します。

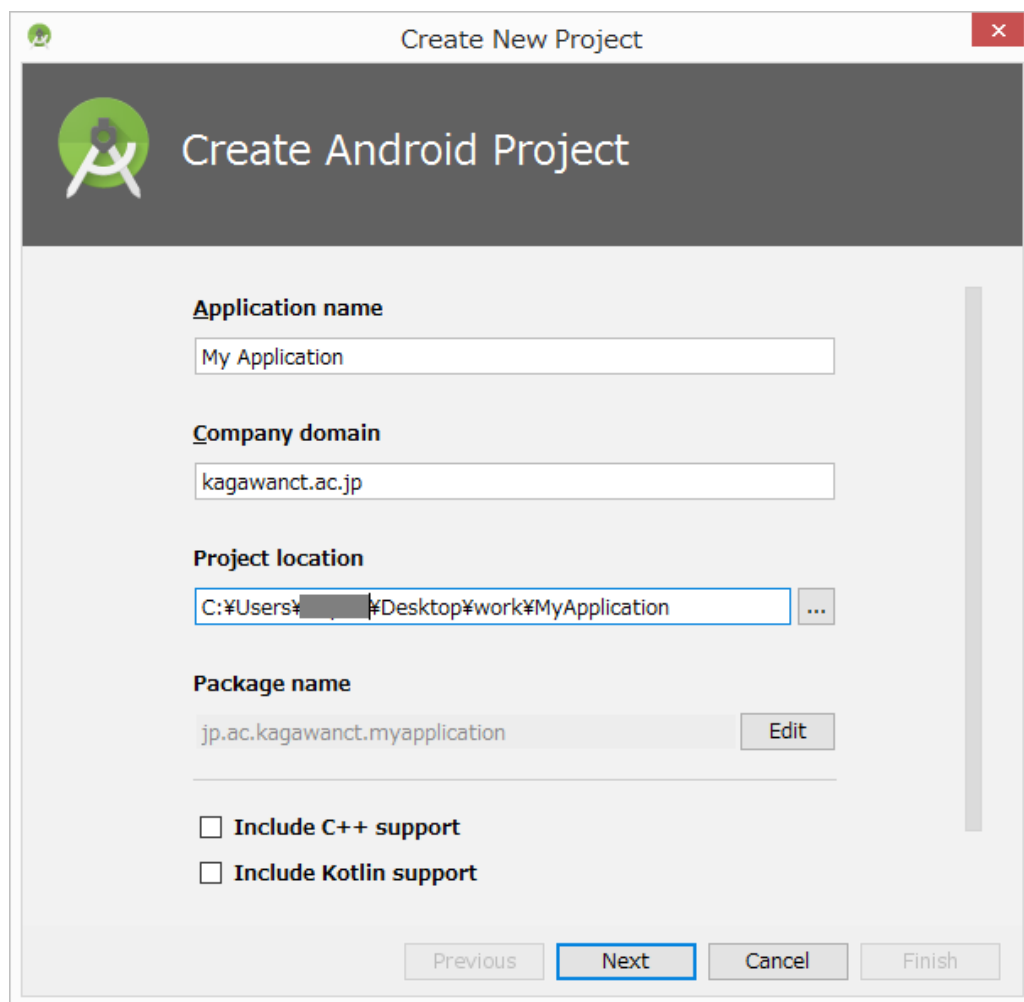
はじめてのアプリ作成（その1）



いちばん上の

「**Start a new Android Studio project**」
をクリックします。

はじめてのアプリ作成（その2）



The screenshot shows the 'Create New Project' dialog box in Android Studio. The dialog has a title bar with a close button. The main area has a dark header with the Android logo and the text 'Create Android Project'. Below the header, there are four sections with labels and input fields:

- Application name**: Input field containing 'My Application'.
- Company domain**: Input field containing 'kagawanct.ac.jp'.
- Project location**: Input field containing 'C:\Users\...\Desktop\work\MyApplication' with a browse button ('...').
- Package name**: Input field containing 'jp.ac.kagawanct.myapplication' with an 'Edit' button.

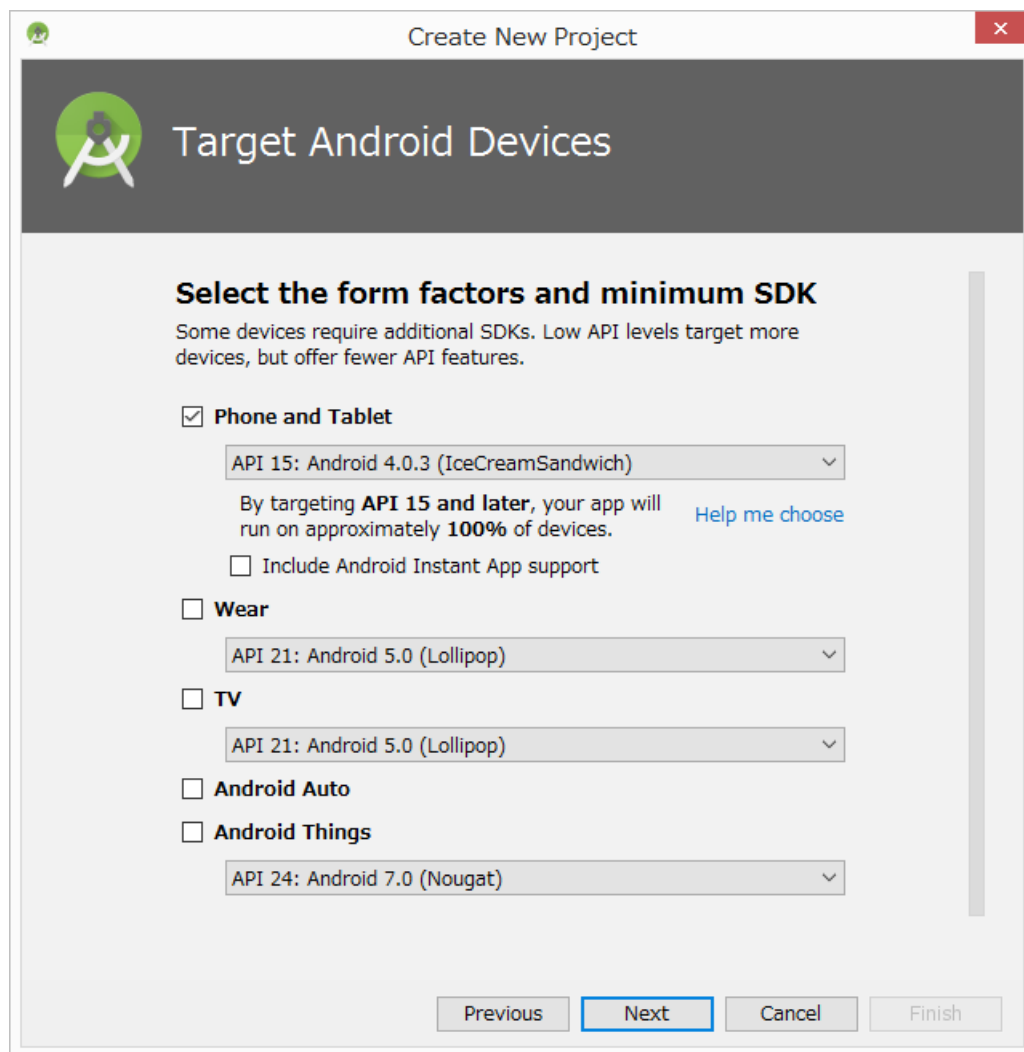
Below these fields, there are two checkboxes:

- ☐ Include C++ support
- ☐ Include Kotlin support

At the bottom, there are four buttons: 'Previous', 'Next' (highlighted with a blue border), 'Cancel', and 'Finish'.

「Application name」に「My Application」、
「Company Domain」に、たとえば「kagawanct.ac.jp」、
「Project location」に、たとえばデスクトップの
「work¥MyApplication」フォルダを指定して、
「Next」をクリックします。

はじめてのアプリ作成（その3）



Create New Project

Target Android Devices

Select the form factors and minimum SDK
Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

☒ **Phone and Tablet**
API 15: Android 4.0.3 (IceCreamSandwich) ▼
By targeting **API 15 and later**, your app will run on approximately **100%** of devices. [Help me choose](#)
☐ Include Android Instant App support

☐ **Wear**
API 21: Android 5.0 (Lollipop) ▼

☐ **TV**
API 21: Android 5.0 (Lollipop) ▼

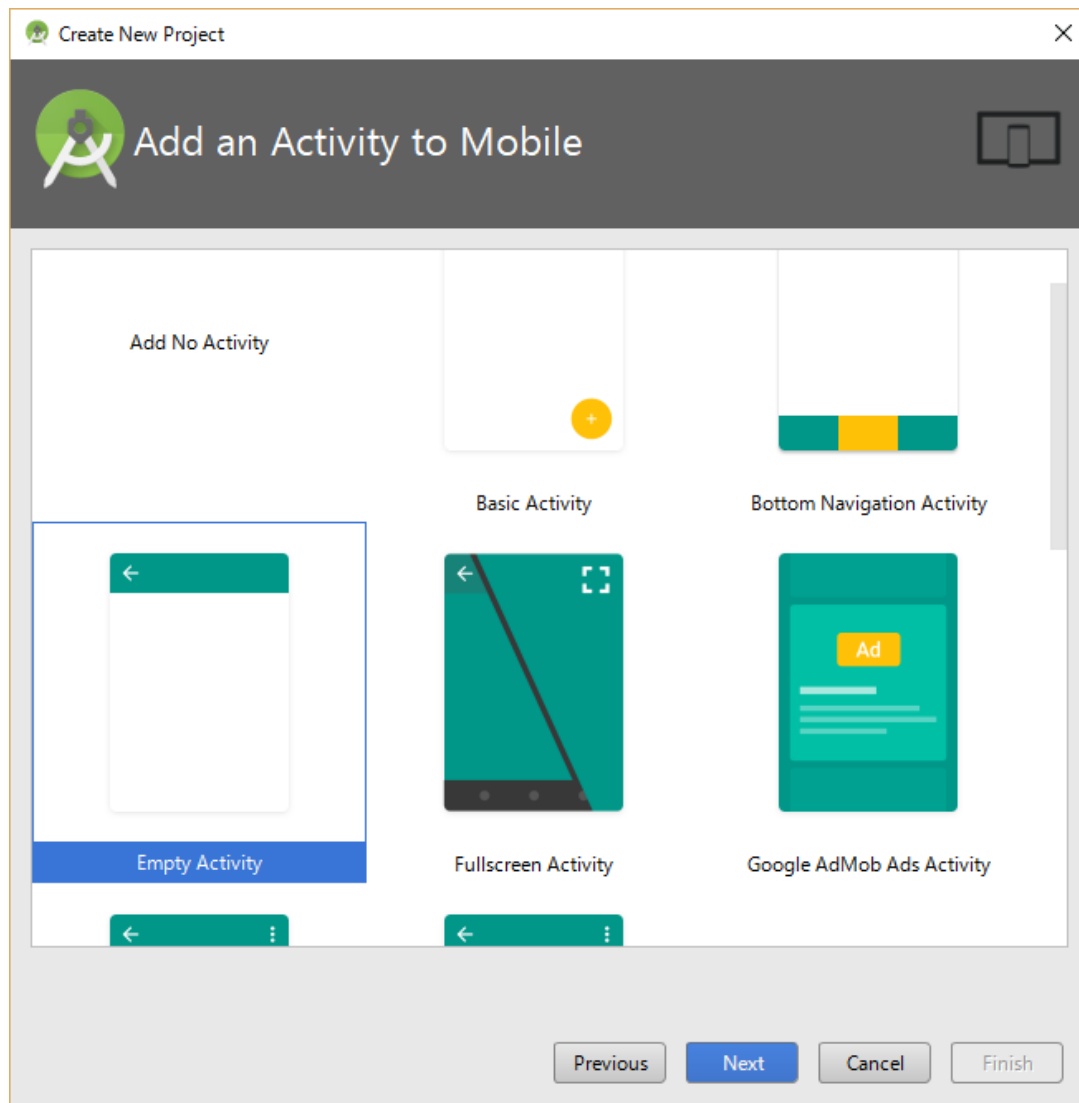
☐ **Android Auto**

☐ **Android Things**
API 24: Android 7.0 (Nougat) ▼

Previous Next Cancel Finish

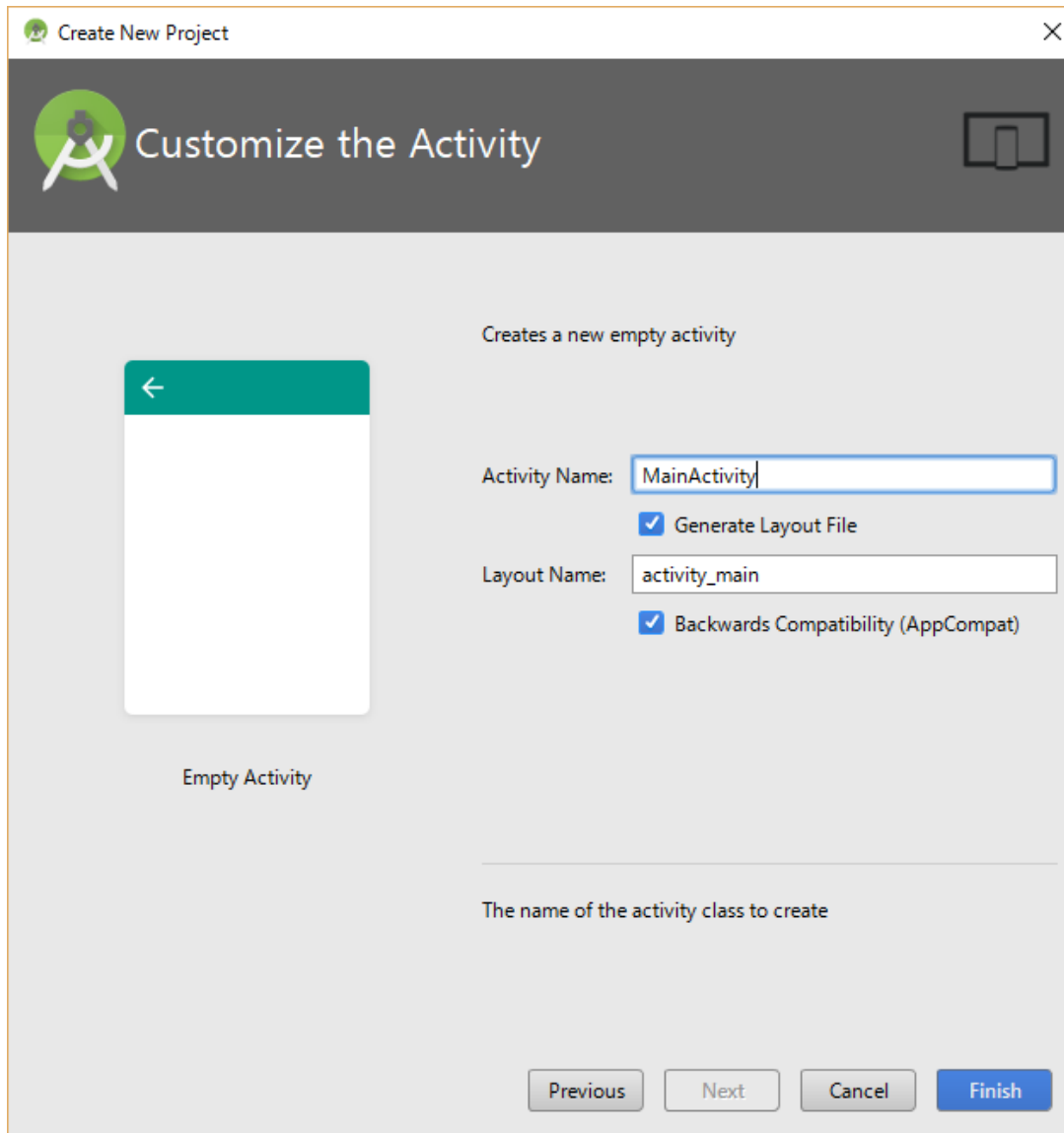
「Phone and Tablet」にチェックがついていることを確認して、そのまま「Next」をクリックします。

はじめてのアプリ作成（その4）



「Empty Activity」を選択して「Next」をクリックします。

はじめてのアプリ作成（その5）



Create New Project

Customize the Activity

Creates a new empty activity

Activity Name: MainActivity

☒ Generate Layout File

Layout Name: activity_main

☒ Backwards Compatibility (AppCompat)

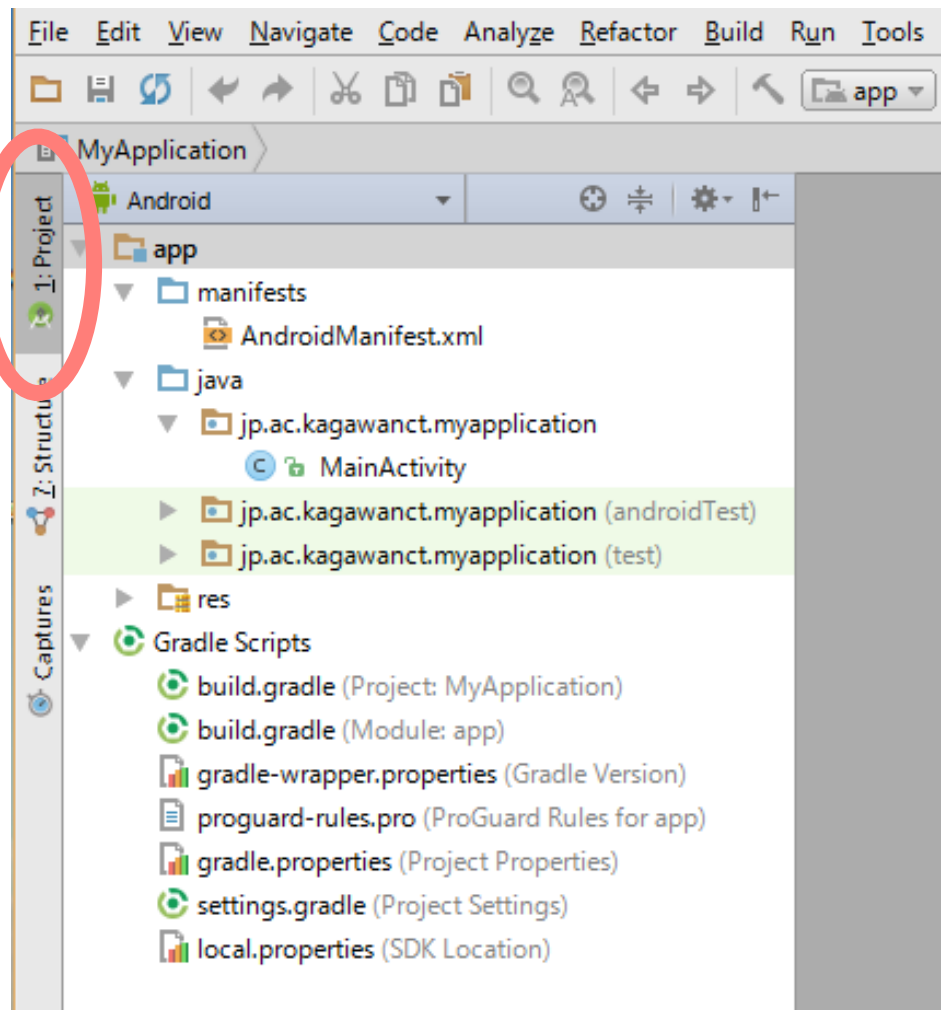
Empty Activity

The name of the activity class to create

Previous Next Cancel Finish

そのまま「**Finish**」をクリックします。

はじめてのアプリ作成（その6）



画面左端の「**Project**」タブをクリックすると、Android Studio が自動生成したファイルを確認することができます。

はじめてのアプリ 実行（その1。スマートフォンの設定）

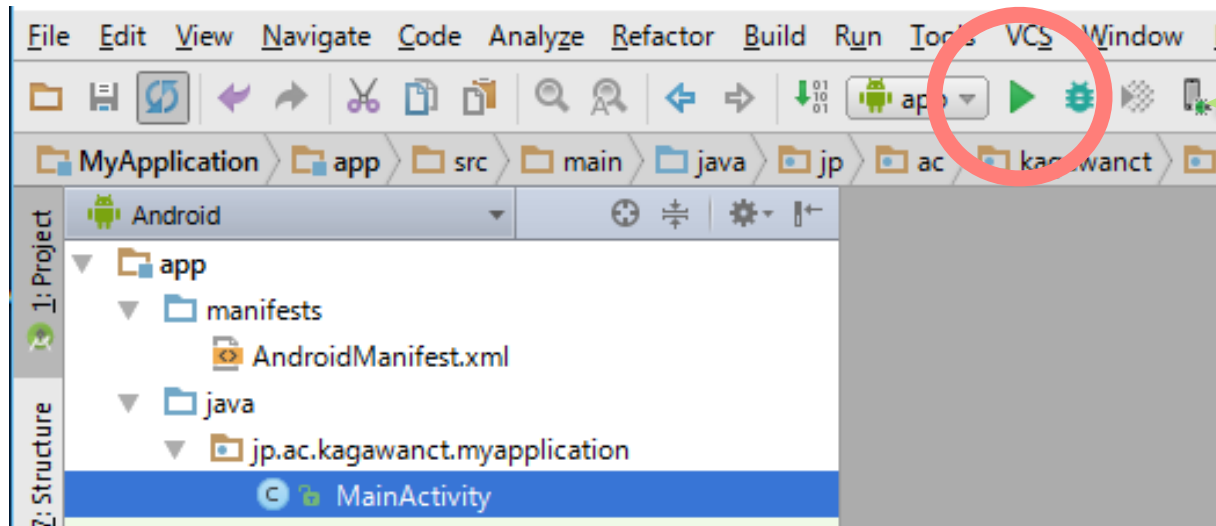


スマートフォンの「設定」アプリを起動して、「端末情報」をタップします。
表示された項目の中の「ビルド番号」を7回連続してタップして、デベロッパー（開発者）モードに設定します。

「設定」アプリのトップ画面に戻って、「開発者向けオプション」が表示されていれば、デベロッパーモードの設定完了です。



はじめてのアプリ 実行（その2。アプリケーションの実行）

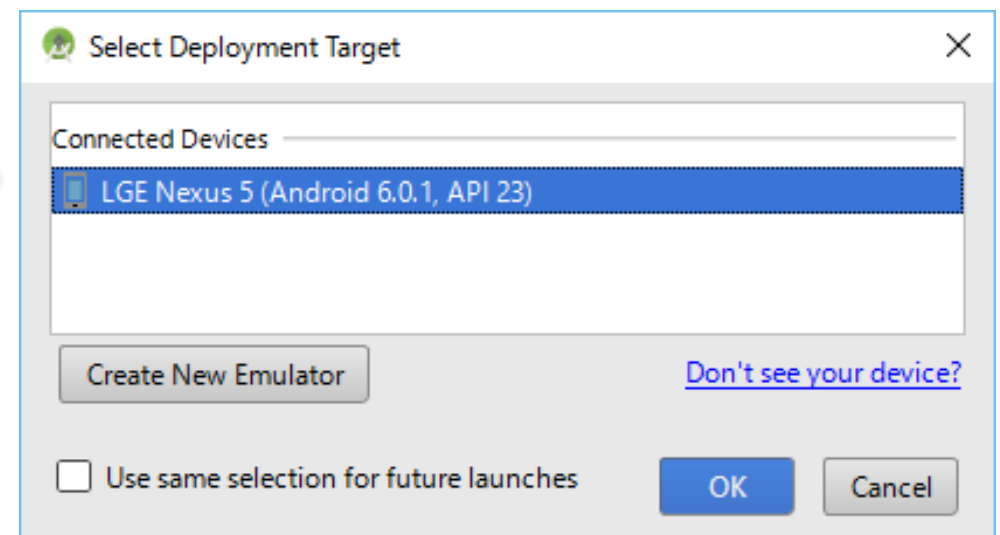


スマートフォンを PC に USB ケーブルで接続します。
その状態で「Run」ボタン（緑色の三角形のボタン）をクリックします。

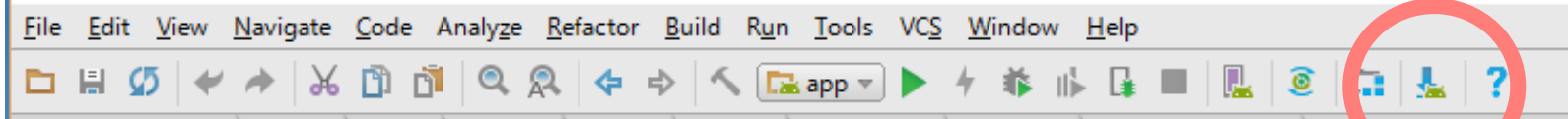
スマートフォンが正しく接続されていると、このように、ダイアログボックスにそのスマートフォンのモデル名が表示されます。

（スマートフォン側で「このコンピュータを信頼しますか」のようなダイアログボックスが表示された場合は「信頼」を選択します。）

「OK」ボタンをクリックして、アプリを実行します。

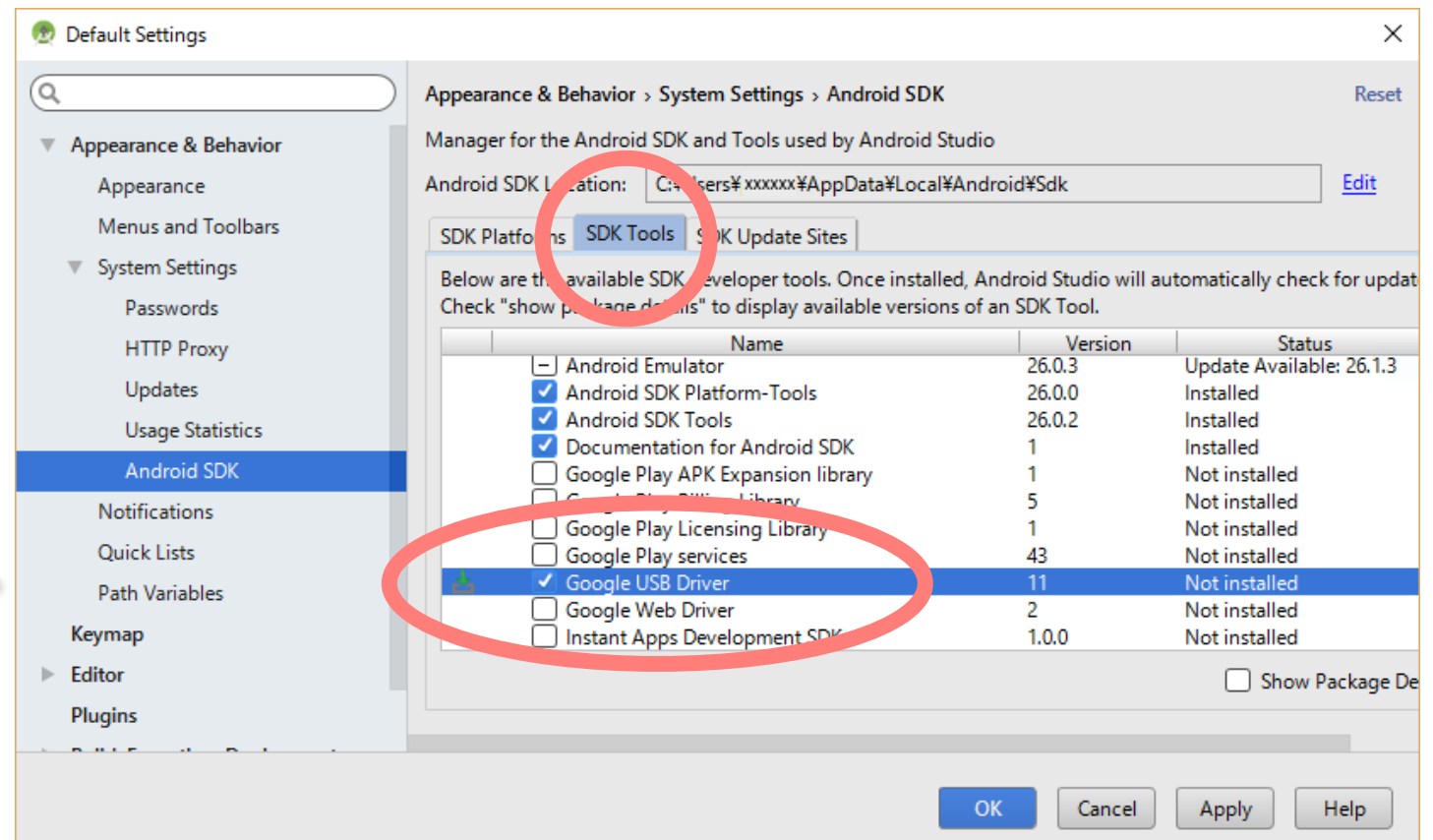


はじめてのアプリ 実行（その3。Google USB Driverインストール（Windowsのみ））

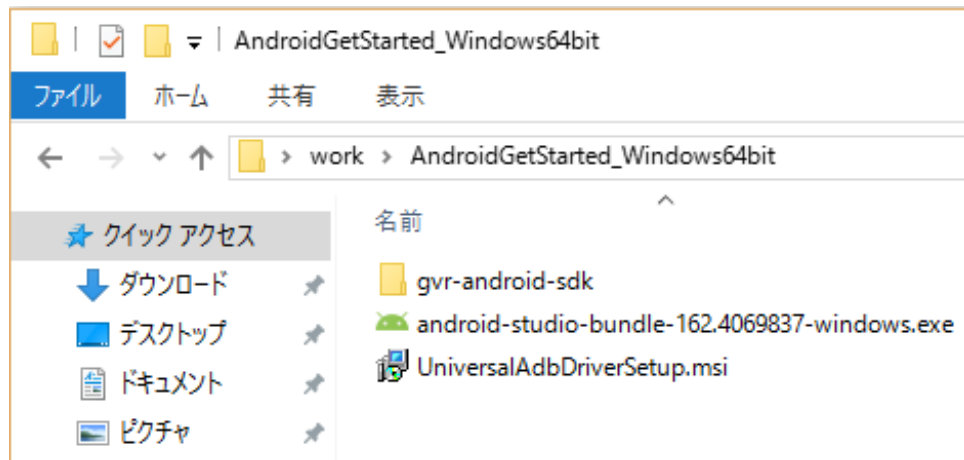


Windows 環境でスマートフォンが認識できない場合、
まずは「Google USB Driver」のインストールを試してみます。
メニューバーから「SDK Manager」のボタンをクリックします。

「SDK Tools」タブを選択して、
その下側に表示される一覧から
「Google USB Driver」
にチェックをつけて、
「OK」ボタンをクリックします。



はじめてのアプリ 実行（その4。Universal ADB Driverインストール（Windowsのみ））



Windows 環境で、「Google USB Driver」をインストールしても、まだスマートフォンが認識されない場合、

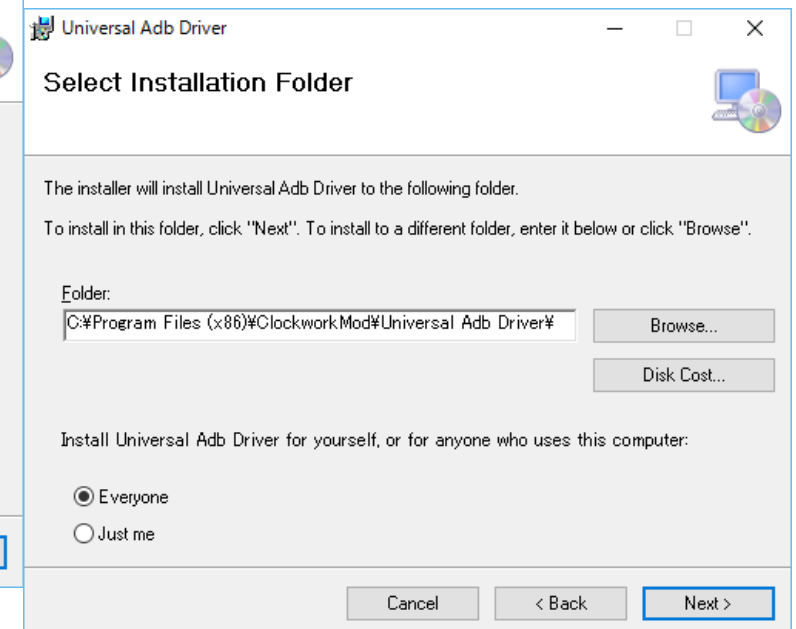
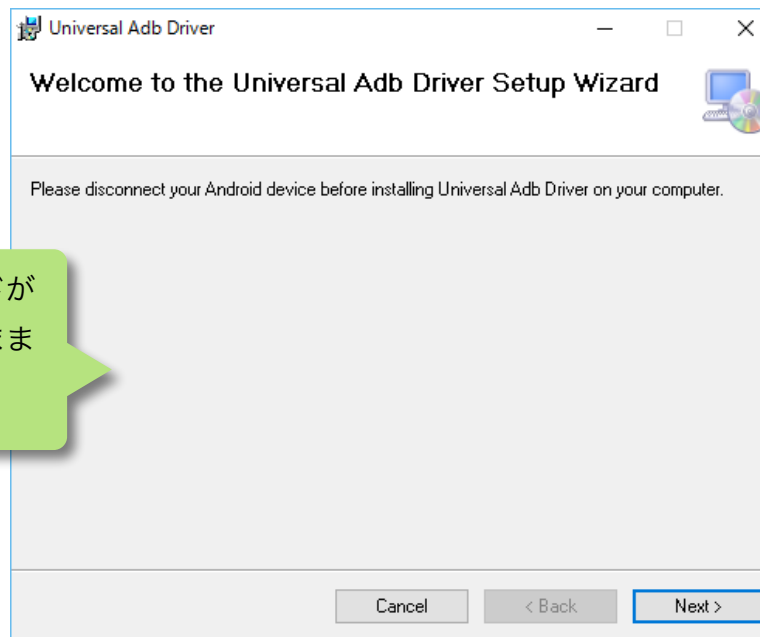
「Universal ADB Driver」のインストールを試してみます。

「AndroidGetStarted_Windows64bit」フォルダの中にある、

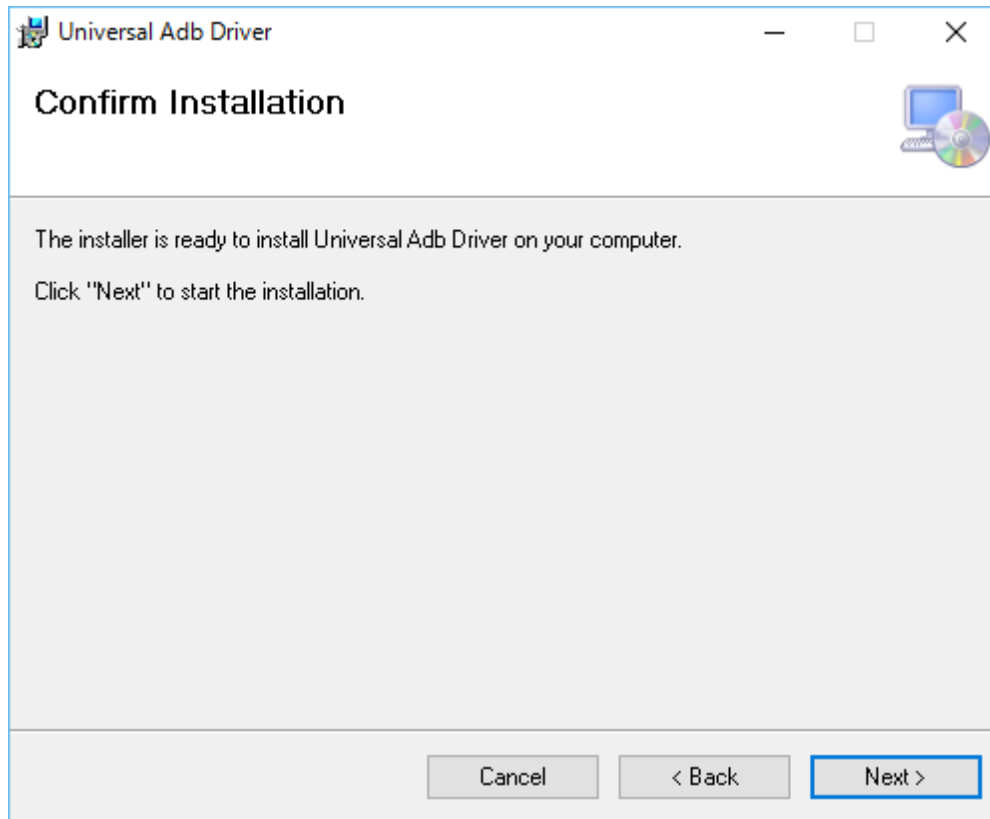
「UniversalAdbDriverSetup.msi」

をダブルクリックして実行します。（管理者権限での操作が必要です。）

このようなインストールウィザードが表示されますので、いずれもそのまま「Next」をクリックします。

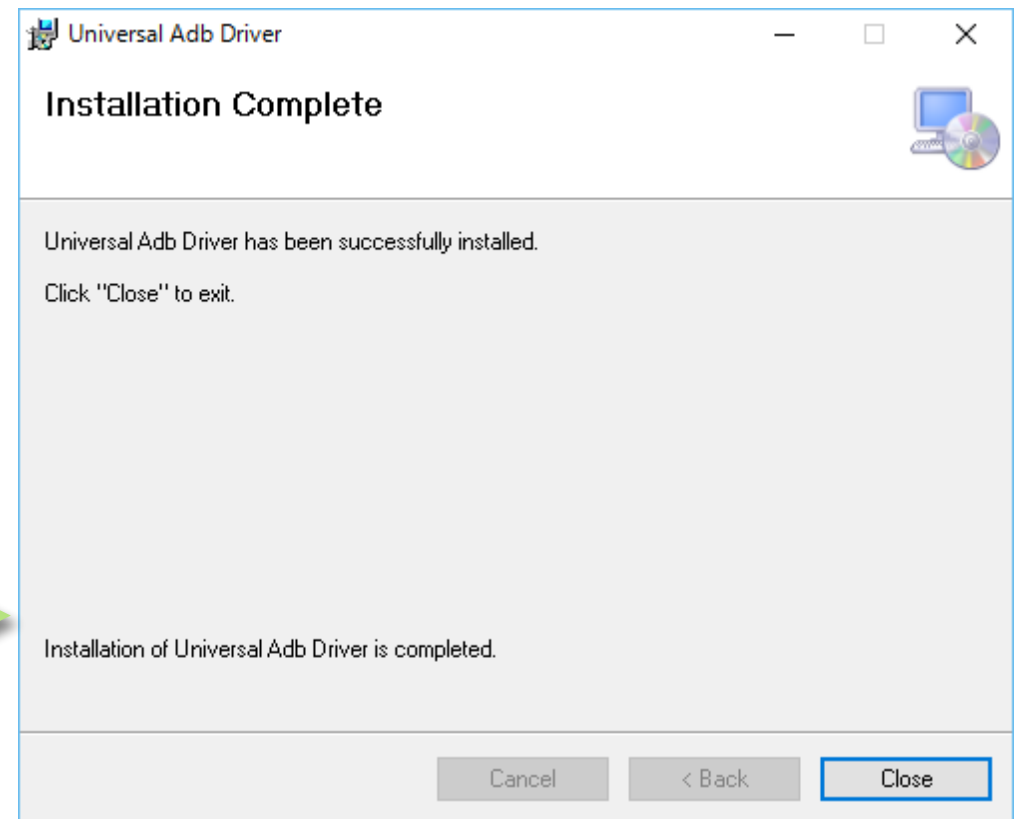


はじめてのアプリ 実行（その5。Universal Adb Driverインストール（Windowsのみ））

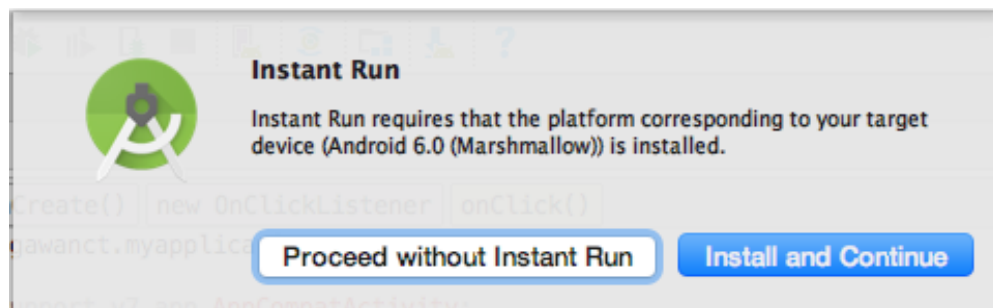


そのまま「Next」をクリックします。

インストール完了後、「Close」をクリックします。

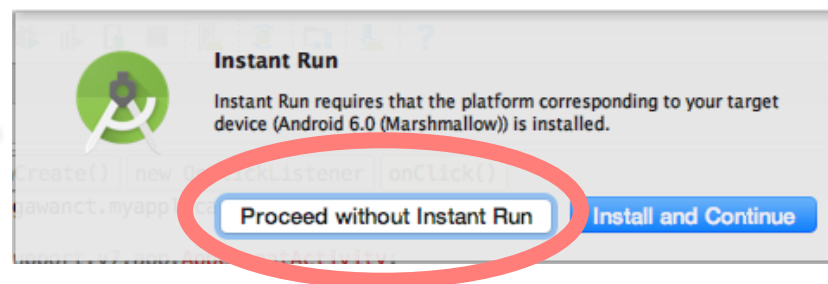


はじめてのアプリ 実行（その6。Instant Run の設定）

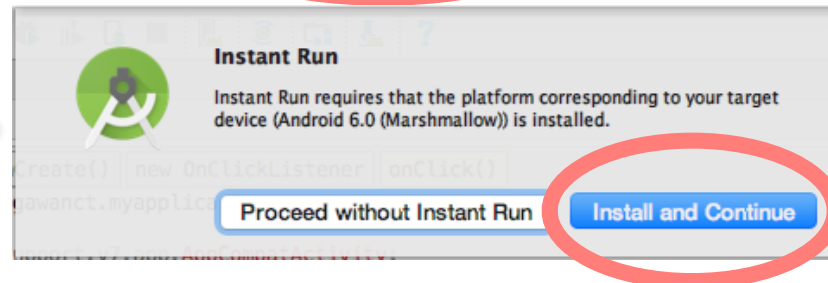


アプリ実行時にこのようなメッセージが表示される場合は、以下いずれかの対応を行います。

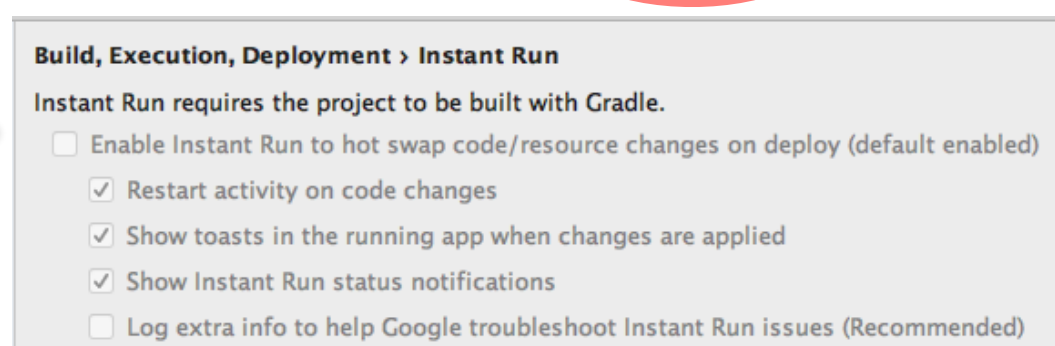
(A) 「Proceed without Instant Run」をクリックして、処理を続行する。（毎回、同手順を実施する必要があります。）



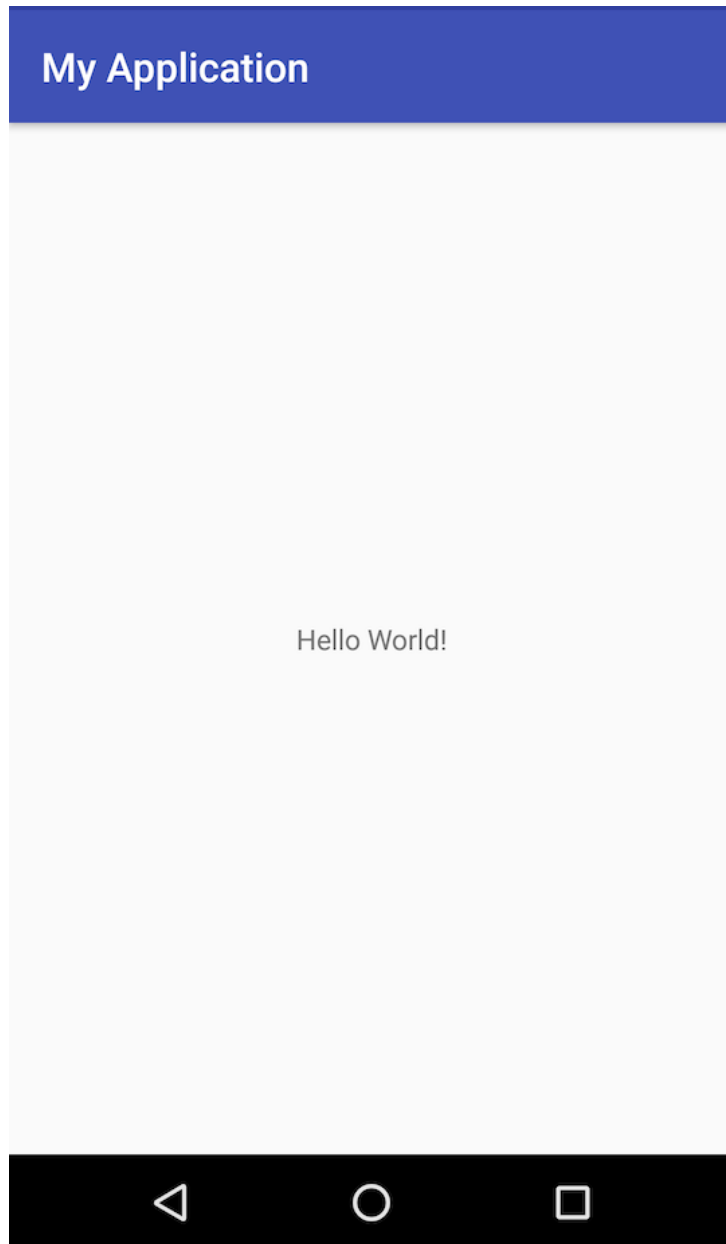
(B) 「Install and Continue」をクリックして、必要なプラットフォームSDKをインストールする。



(C) Android Studio の設定画面から「Build, Execution, Deployment」->「Instant Run」を選択して、Instant Run を無効にする。

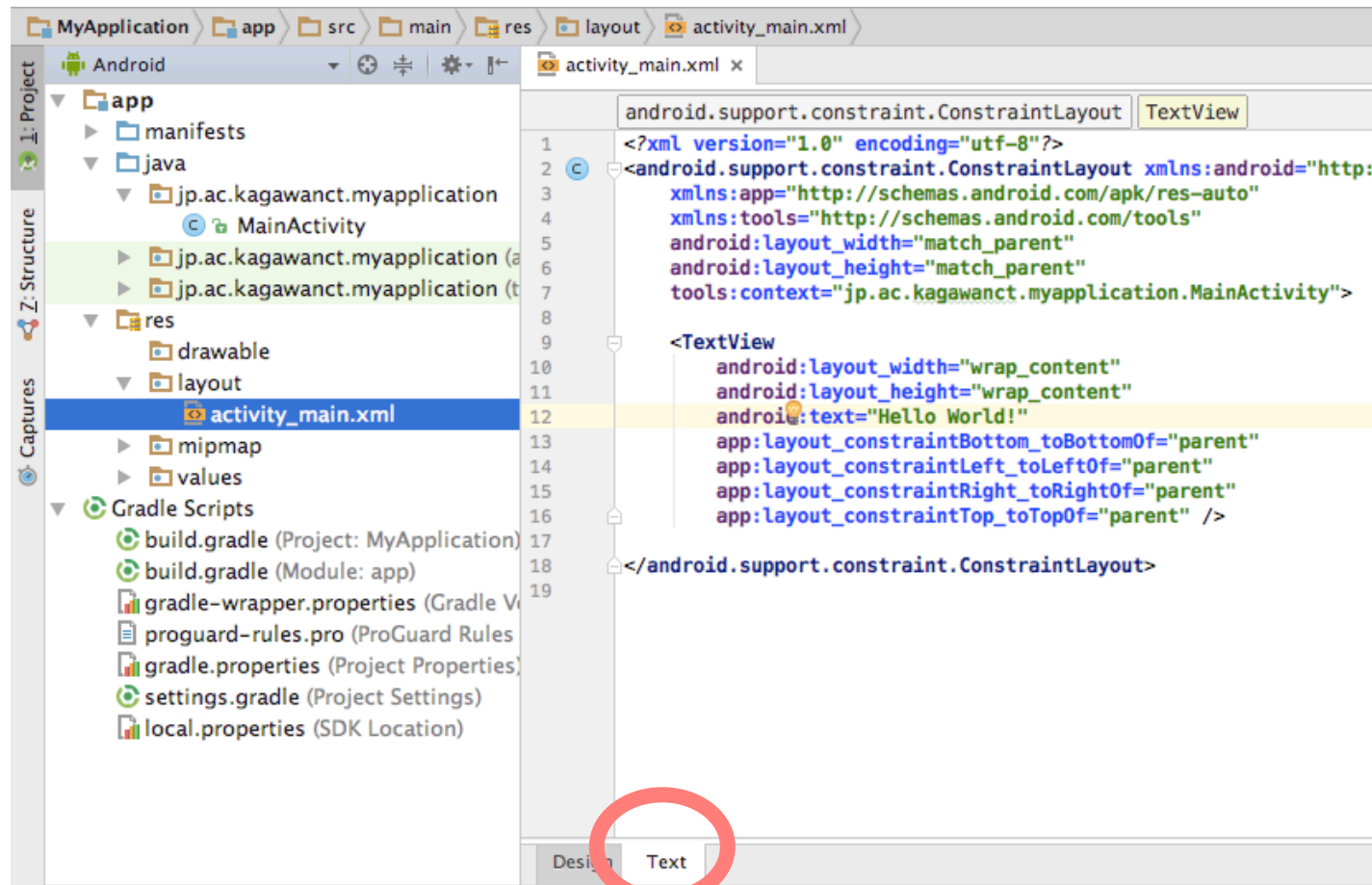


はじめてのアプリ 実行 (その7)



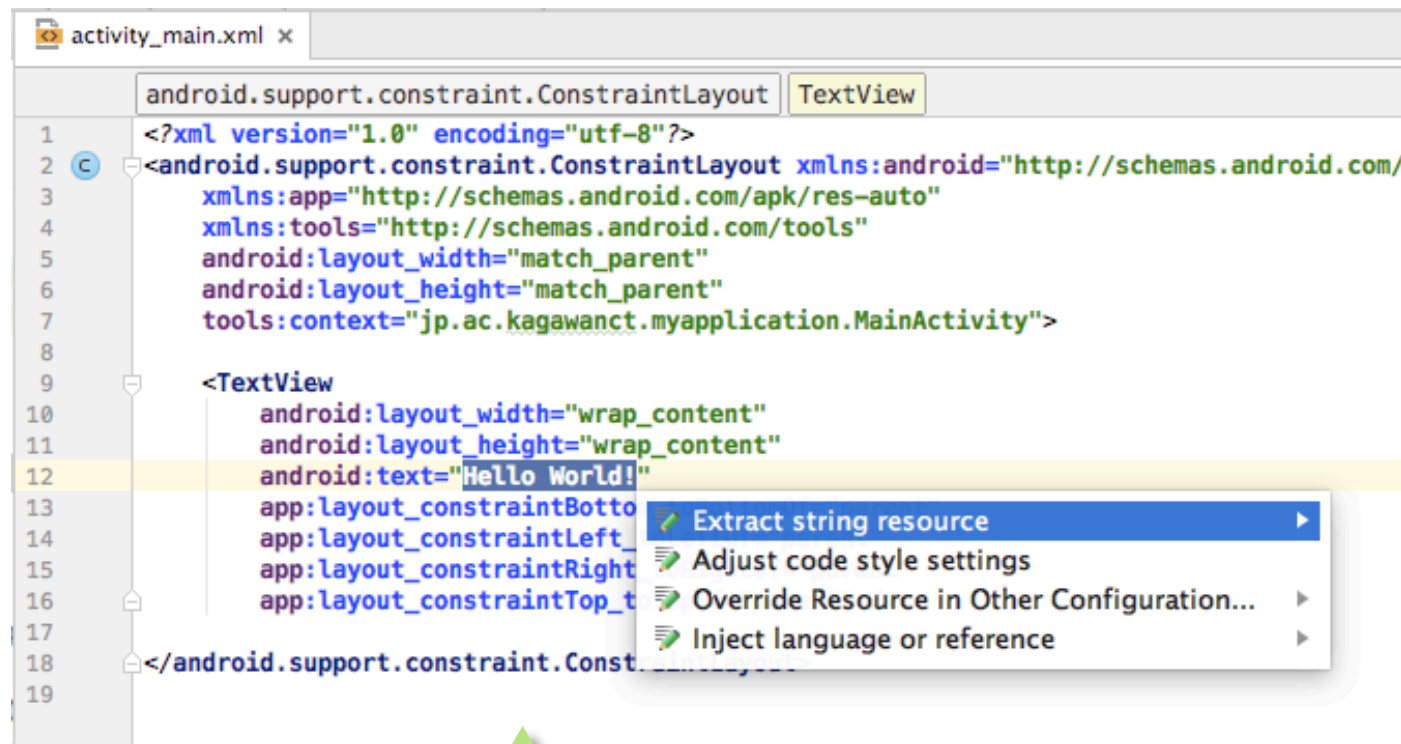
正しくアプリが実行できた場合、このように、
「My Application」というタイトルと、
「Hello World!」というメッセージのある画面が表示されます。

画面表示メッセージの変更（その1）



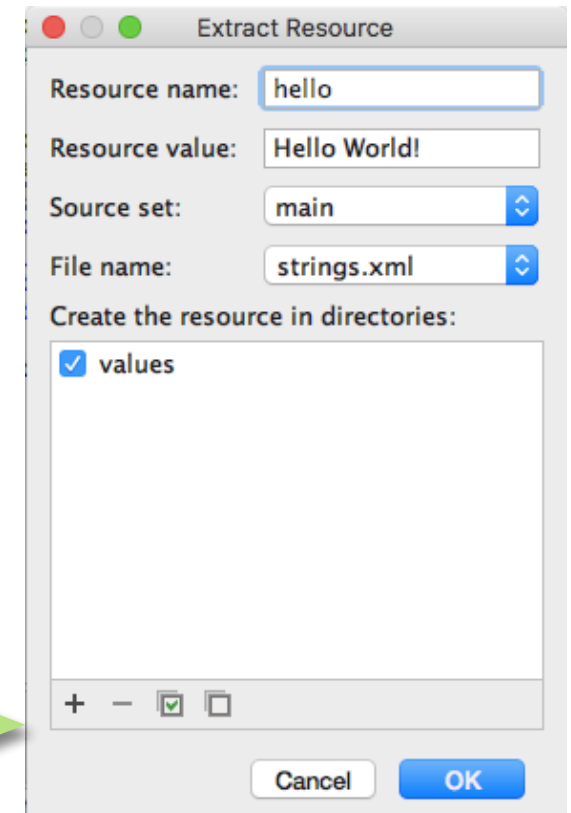
「Hello World!」というメッセージが定義されているのは、
「app」 → 「res」 → 「layout」 フォルダの、「**activity_main.xml**」ファイルです。
Android Studio ツール左側の「Project」欄から、この「**activity_main.xml**」をダブルクリックすると、
右側にその内容が表示されます。
このとき、下側のタブは、最初は「Design」が選択されていますので、これを「Text」に切り替えます。

画面表示メッセージの変更（その2）



「**android:text=**」のあとの、ダブルクォーテーション（**"**）で囲われた部分に記入されている「**Hello World!**」の部分にカーソルを合わせて、
「**Alt + Enter**」キー（Mac の場合は「**option + enter**」）を押します。
ポップアップ表示されたダイアログボックスの中から「**Extract string resource**」を選択します。

このような画面が表示されますので、いちばん上の「**Resource name**」に「**hello**」と入力して「**OK**」をクリックします。



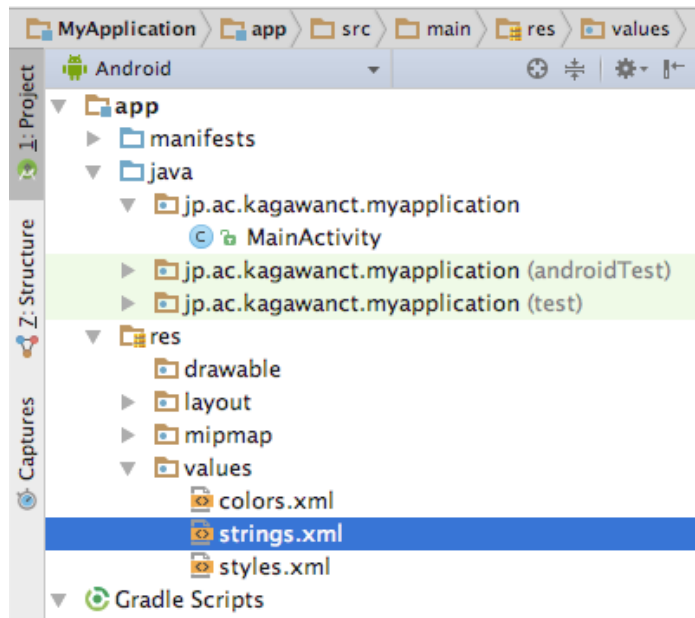
画面表示メッセージの変更（その3）



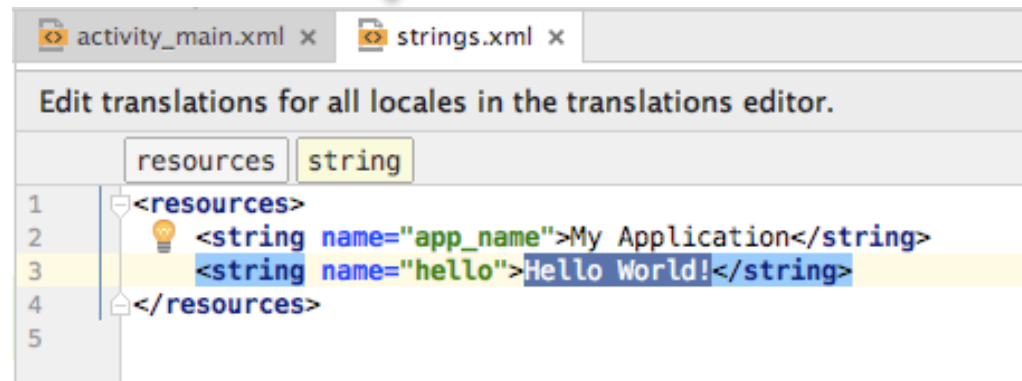
```
activity_main.xml x
android.support.constraint.ConstraintLayout TextView
1 <?xml version="1.0" encoding="utf-8"?>
2 <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
3   xmlns:app="http://schemas.android.com/apk/res-auto"
4   xmlns:tools="http://schemas.android.com/tools"
5   android:layout_width="match_parent"
6   android:layout_height="match_parent"
7   tools:context="jp.ac.kagawa.ct.myapplication.MainActivity">
8
9   <TextView
10     android:layout_width="wrap_content"
11     android:layout_height="wrap_content"
12     android:text="@string/hello"
13     app:layout_constraintBottom_toBottomOf="parent"
14     app:layout_constraintLeft_toLeftOf="parent"
15     app:layout_constraintRight_toRightOf="parent"
16     app:layout_constraintTop_toTopOf="parent" />
17
18 </android.support.constraint.ConstraintLayout>
19
```

「Hello World!」と表記されていた部分には、代わりに「@string/hello」と表示されます。

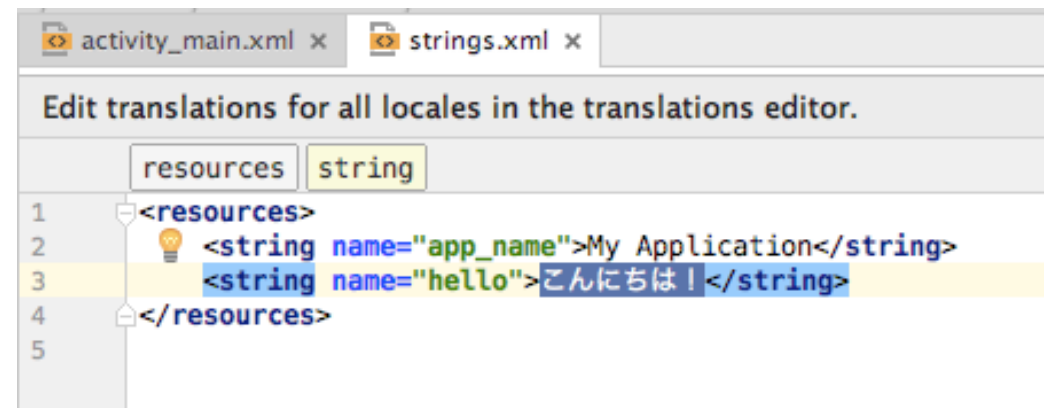
画面表示メッセージの変更（その4）



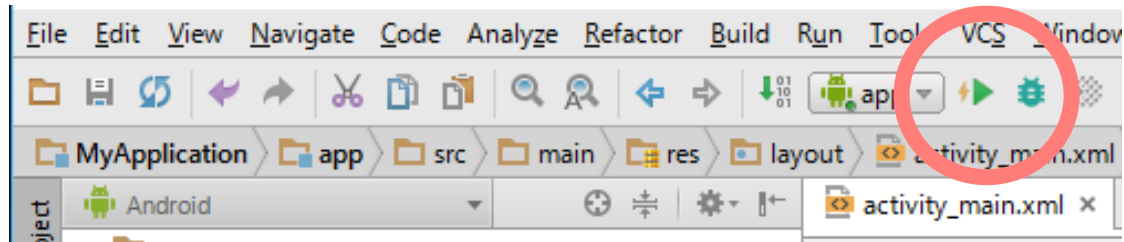
「Hello World!」というメッセージは、
「res」 → 「values」 フォルダの「strings.xml」のほうで、
「<string name="hello">」タグの要素として登録されています。



この「strings.xml」の中で登録されている
「Hello World!」という部分を、今回は
「こんにちは!」というメッセージに変更してみます。

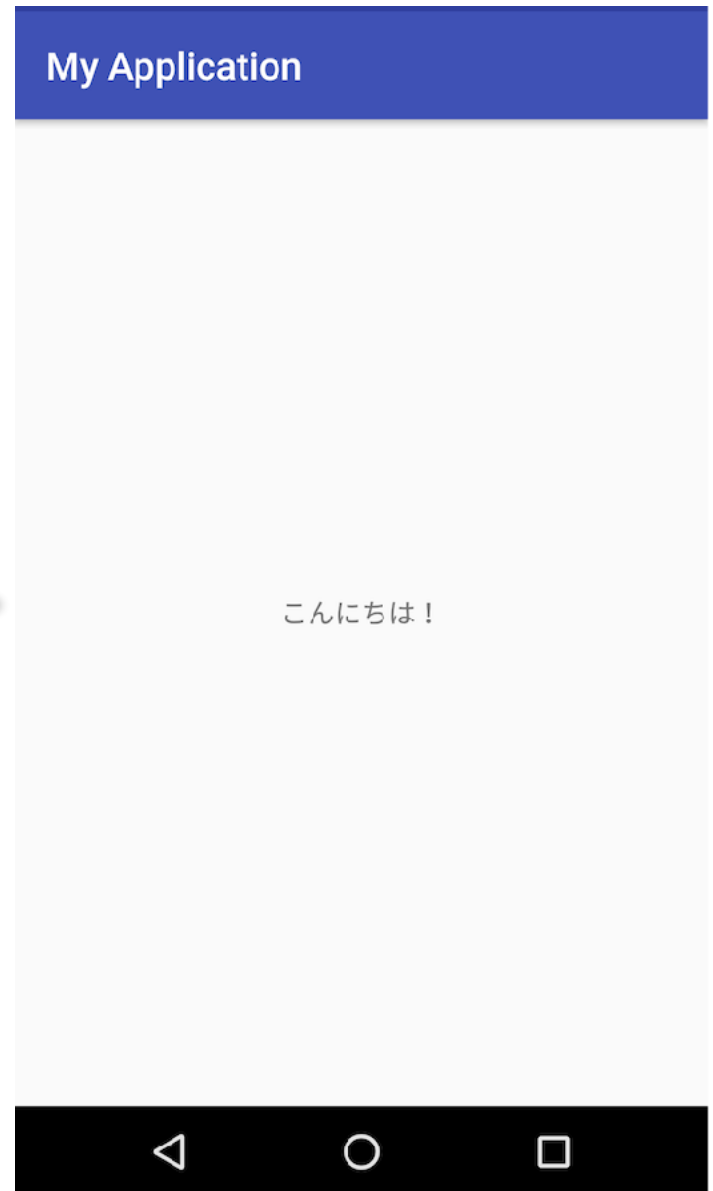


画面表示メッセージの変更（その4）



再度、「Run」ボタン（緑色の三角形のボタン）をクリックして、アプリを実行します。

スマートフォン側で、アプリが再起動し、設定したメッセージが画面上に表示されます。



ボタンの追加（その1）

```
activity_main.xml x
android.support.constraint.ConstraintLayout
2  <android.support.constraint.ConstraintLayout xmlns:android="http://
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context="jp.ac.kagawanct.myapplication.MainActivity">
8
9      <TextView
10          android:layout_width="wrap_content"
11          android:layout_height="wrap_content"
12          android:text="@string/hello"
13          app:layout_constraintBottom_toBottomOf="parent"
14          app:layout_constraintLeft_toLeftOf="parent"
15          app:layout_constraintRight_toRightOf="parent"
16          app:layout_constraintTop_toTopOf="parent"
17          android:id="@+id/text_view" />
18
19      <Button
20          android:id="@+id/button_message"
21          android:layout_width="wrap_content"
22          android:layout_height="wrap_content"
23          android:text="メッセージを表示します"
24          app:layout_constraintTop_toBottomOf="@+id/text_view"
25          app:layout_constraintLeft_toLeftOf="parent"
26          app:layout_constraintRight_toRightOf="parent" />
27
28  </android.support.constraint.ConstraintLayout>
29
```

先ほどと同じ「activity_main.xml」ファイルに、
以下のように入力します。

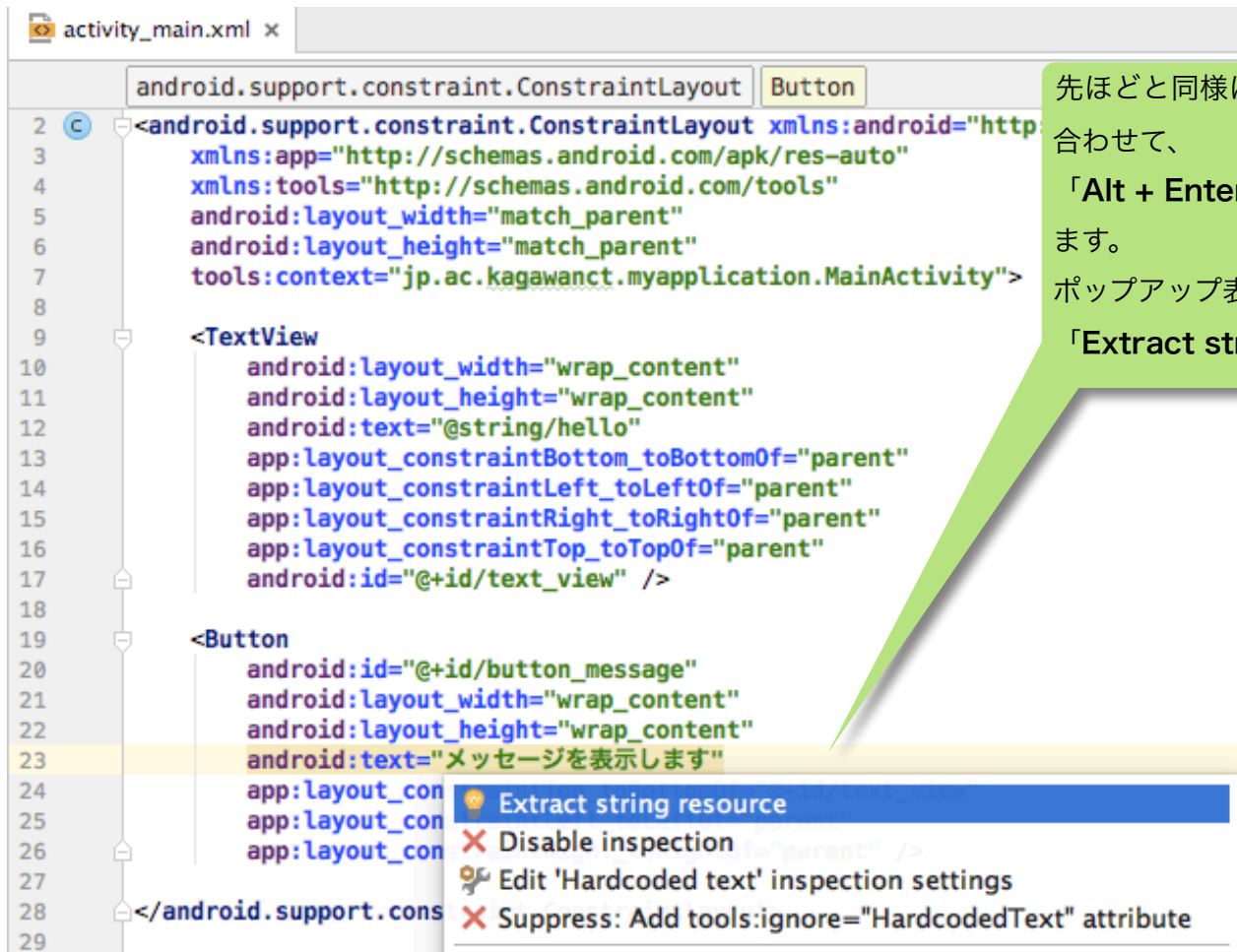
（赤色が、新しく入力する部分です。）

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:id="@+id/text_view" />

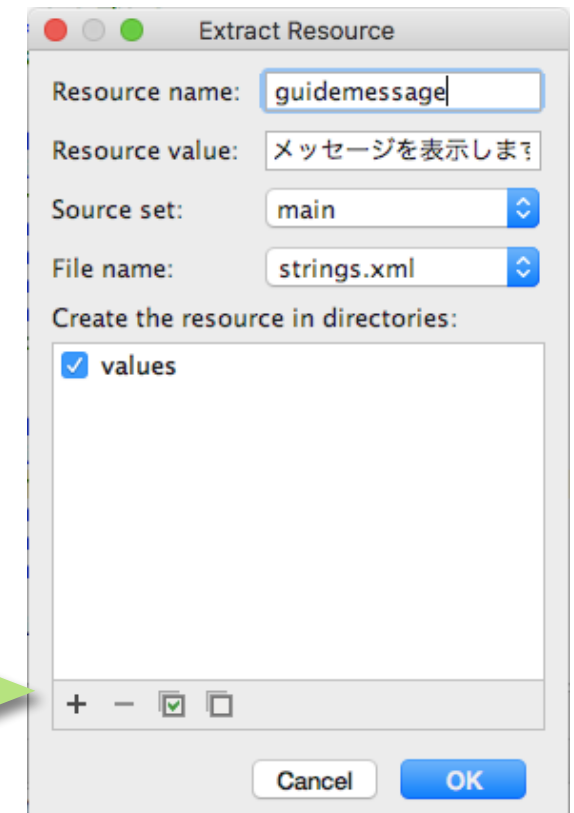
<Button
    android:id="@+id/button_message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="メッセージを表示します"
    app:layout_constraintTop_toBottomOf="@+id/text_view"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent" />

</android.support.constraint.ConstraintLayout>
```

ボタンの追加 (その2)



先ほどと同様に、「メッセージを表示します」の部分にカーソルを合わせて、
「Alt + Enter」キー（Mac の場合は「option + enter」）を押します。
ポップアップ表示されたダイアログボックスの中から
「Extract string resource」を選択します。



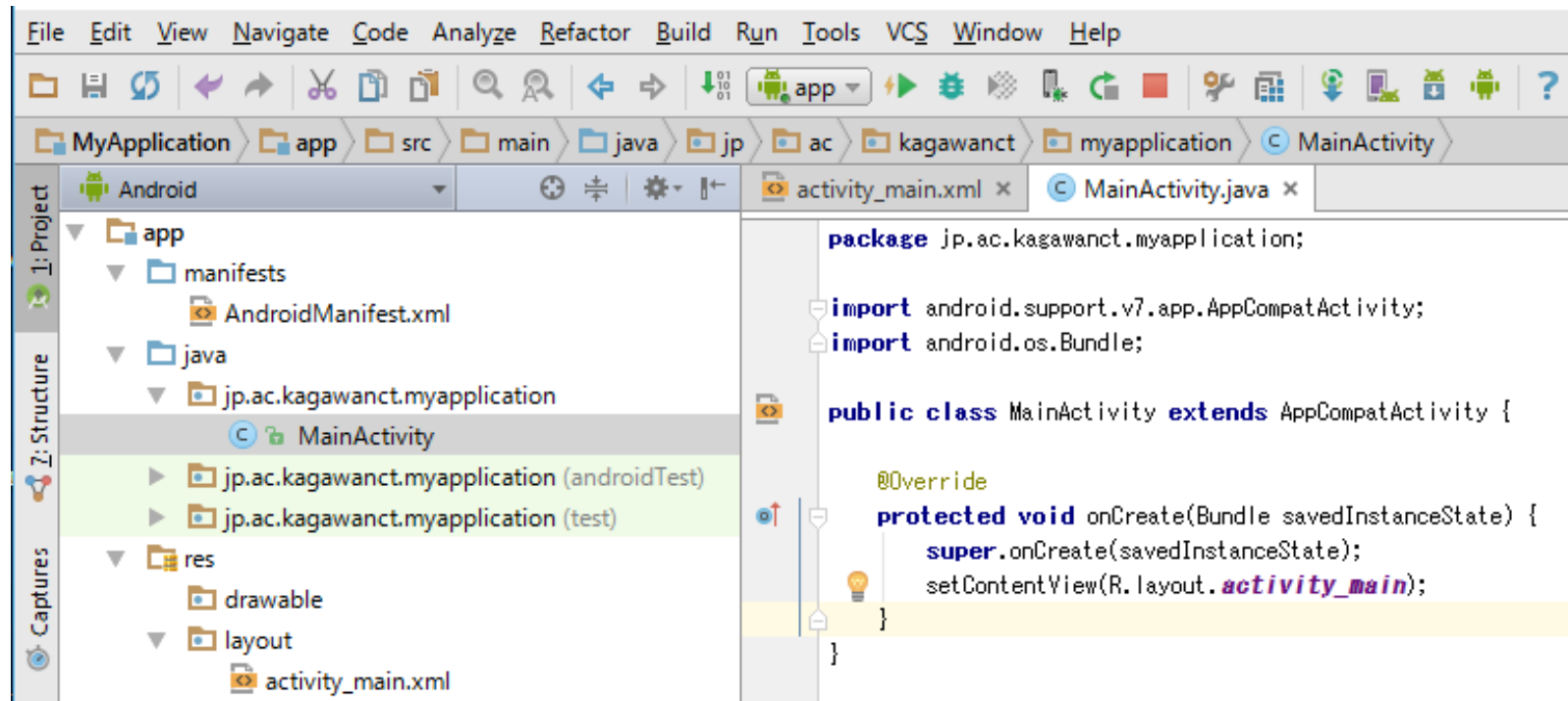
このような画面が表示されますので、いちばん上の「Resource name」に「guidemessage」と入力して「OK」をクリックします。

ボタンの追加（その3）



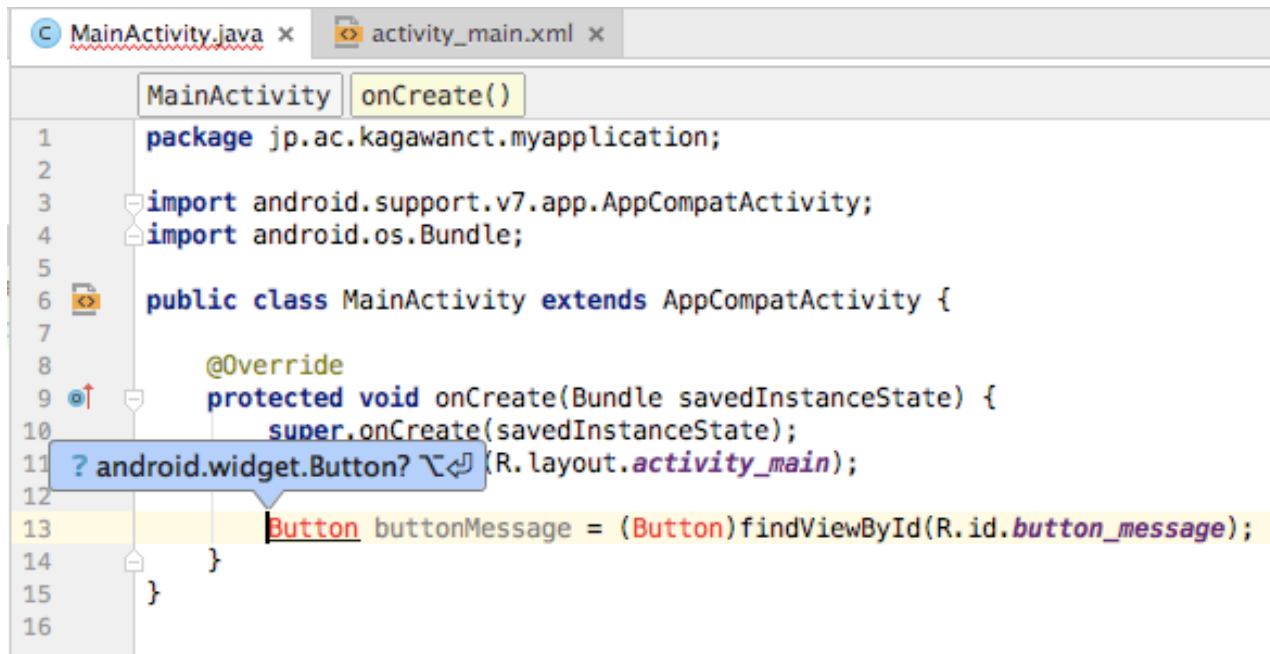
アプリを再実行すると、
メッセージの下に、ボタンが表示されます。

ボタンタップ時にメッセージを表示する（その1）



Android Studio ツール左側の「Project」欄から、
「app」 → 「java」 → 「～.myapplication」 → 「**MainActivity**」を選択してダブルクリックします。
右側に、この「**MainActivity**クラス（**MainActivity.java**）」の内容が表示されます。

ボタンタップ時にメッセージを表示する（その2）



```
1 package jp.ac.kagawanct.myapplication;
2
3 import android.support.v7.app.AppCompatActivity;
4 import android.os.Bundle;
5
6 public class MainActivity extends AppCompatActivity {
7
8     @Override
9     protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         ? android.widget.Button? (R.layout.activity_main);
12
13         Button buttonMessage = (Button)findViewById(R.id.button_message);
14     }
15 }
16
```

MainActivityクラスの「onCreate」メソッド内に、このようにコードを追記します。
(赤色が、新しく入力する部分です。)

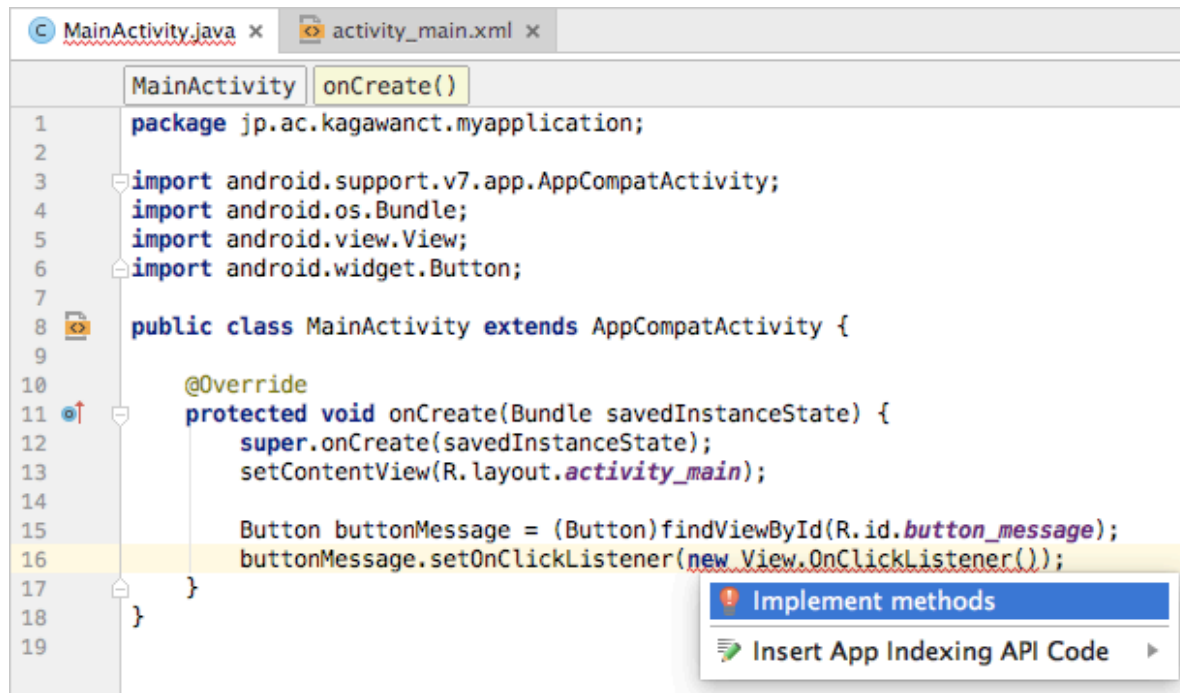
コード追記後、「Button」部分にカーソルを合わせて、「Alt + Enter」キー
(Macの場合は「option + enter」)を押します。
こうすることで「Button」クラスのimport文が自動的に追加されます。

```
public class MainActivity extends AppCompatActivity {

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

        Button buttonMessage = (Button)findViewById(R.id.button_message);
    }
}
```

ボタンタップ時にメッセージを表示する（その3）

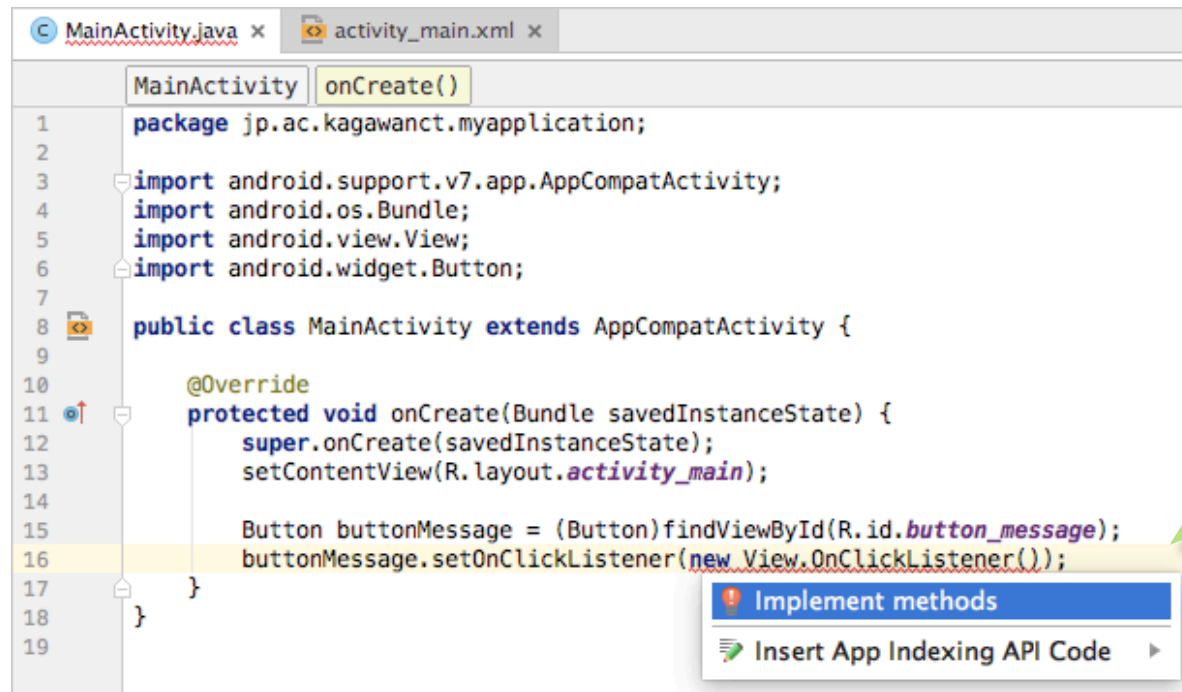


```
1 package jp.ac.kagawanct.myapplication;
2
3 import android.support.v7.app.AppCompatActivity;
4 import android.os.Bundle;
5 import android.view.View;
6 import android.widget.Button;
7
8 public class MainActivity extends AppCompatActivity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_main);
14
15         Button buttonMessage = (Button)findViewById(R.id.button_message);
16         buttonMessage.setOnClickListener(new View.OnClickListener());
17     }
18 }
19
```

先ほど入力したコードのすぐ下に、以下のコードを追記します。
(赤色が、新しく入力する部分です。)

```
...
    Button buttonMessage = (Button)findViewById(R.id.button_message);
    buttonMessage.setOnClickListener(new View.OnClickListener());
}
}
```

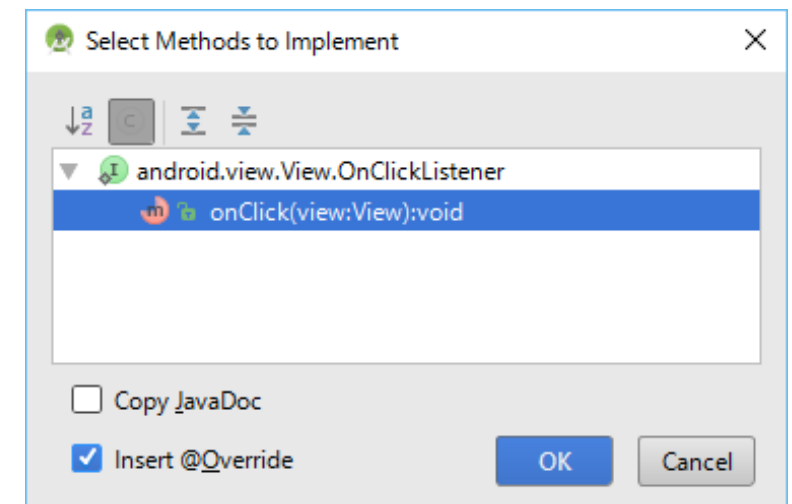
ボタンタップ時にメッセージを表示する（その4）



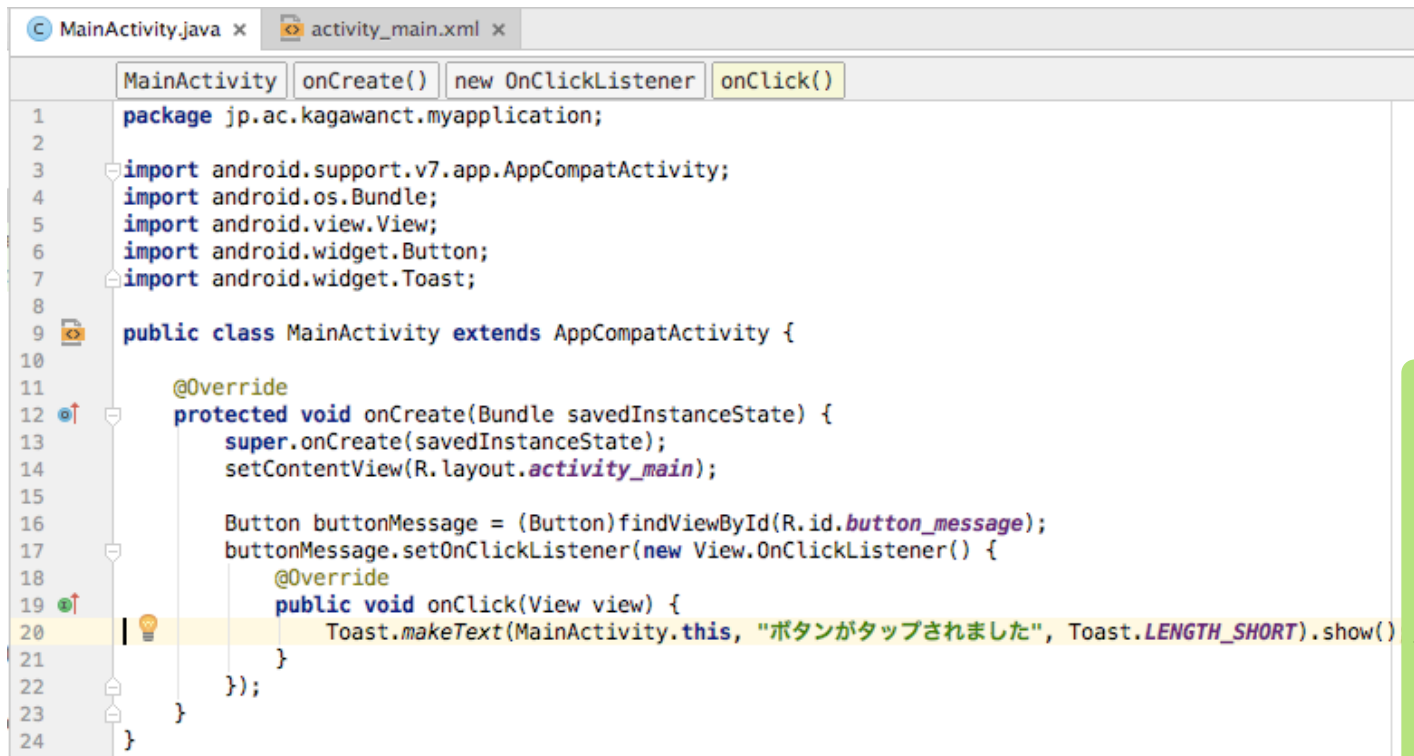
コード追記後、

- 1) 「**View**」部分にカーソルを合わせて「**Alt + Enter**」(Macの場合は「**option + enter**」)。Viewクラスのimport文が自動挿入されます。
- 2) 「**new View.OnClickListener()**」部分にカーソルを合わせて、「**Alt + Enter**」キー(Macの場合は「**option + enter**」)を押します。

ポップアップ表示されたダイアログボックスの中から「**Implement methods**」を選択して、「**onClick**」メソッドを追加します。



ボタンタップ時にメッセージを表示する（その5）



```
1 package jp.ac.kagawanct.myapplication;
2
3 import android.support.v7.app.AppCompatActivity;
4 import android.os.Bundle;
5 import android.view.View;
6 import android.widget.Button;
7 import android.widget.Toast;
8
9 public class MainActivity extends AppCompatActivity {
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.activity_main);
15
16         Button buttonMessage = (Button)findViewById(R.id.button_message);
17         buttonMessage.setOnClickListener(new View.OnClickListener() {
18             @Override
19             public void onClick(View view) {
20                 Toast.makeText(MainActivity.this, "ボタンがタップされました", Toast.LENGTH_SHORT).show();
21             }
22         });
23     }
24 }
```

前の手順で自動的に追加された「onClick」メソッド内に、以下のコードを追記します。（赤色が、新しく入力する部分です。）

コード追加後、「Toast」部分にカーソルを合わせて、「Alt + Enter」（Mac の場合は「option + enter」）して、Toast クラスのimport 文を自動挿入します。

```
...
buttonMessage.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Toast.makeText(MainActivity.this, "ボタンがタップされました", Toast.LENGTH_SHORT).show();
    }
});
...
```

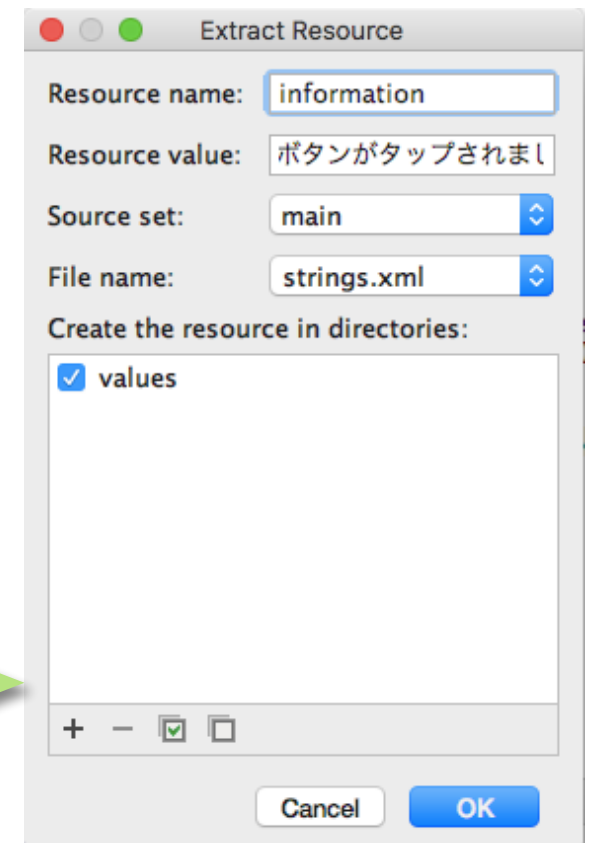
ボタンタップ時にメッセージを表示する（その6）

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        Button buttonMessage = (Button)findViewById(R.id.button_message);  
        buttonMessage.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View view) {  
                Toast.makeText(MainActivity.this, "ボタンがタップされました", Toast.LENGTH_SHORT).show();  
            }  
        });  
    }  
}
```

- ✓ Extract string resource ▶
- Convert to Basic Latin ▶
- Insert App Indexing API Code ▶
- Inject language or reference ▶

「ボタンがタップされました」の部分にカーソルを合わせて、
「Alt + Enter」キー（Macの場合は「option + enter」）
を押します。
ポップアップ表示されたダイアログボックスの中から
「Extract string resource」を選択します。

このような画面が表示されますので、いちばん上の「Resource name」
に「information」と入力して「OK」をクリックします。



ボタンタップ時にメッセージを表示する（その7）

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        Button buttonMessage = (Button)findViewById(R.id.button_message);  
        buttonMessage.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View view) {  
                Toast.makeText(MainActivity.this, R.string.information, Toast.LENGTH_SHORT).show();  
            }  
        });  
    }  
}
```

「ボタンがタップされました」と入力していた部分には、
「R.string.information」と表示されます。

ボタンタップ時にメッセージを表示する（その8）



「メッセージを表示します」ボタンをタップすると、画面下部にメッセージが表示されます。