SECTION 02

Androidアプリ開発に必要なものを揃えよう

このセクションでは、Androidアプリを作り始めるためのツールを揃えます。いくつかの手順は踏みますが、インターネットからダウンロードしてきて、少しだけ操作をして、あとはパソコンが設定が終えてくれるまでの長い待ち時間を過ごせばよいものばかりです。まだスタートラインの少し手前です。肩の力を抜いて、コーヒーでも飲みながら操作してください。

Androidアプリ開発に必要なもの

Androidアプリ開発に必要なものを確認します。まず、手元に必要なのが、インターネットに繋がったパソコンです。これがないと始まりませんね。そして、パソコンの用意ができたら、そこにAndroid StudioとAndroid SDKを導入します。それぞれについて、詳しく解説します。

インターネットに繋がったパソコンを用意する

何はともあれ、アプリの開発にはパソコンが必要です。特にAndroidアプリの開発を行う場合には、インターネットに繋がっているパソコンが必要になります。作ろうとしているアプリがインターネットとの通信を必要としていない場合でも、アプリの開発にはインターネット環境が必要です。

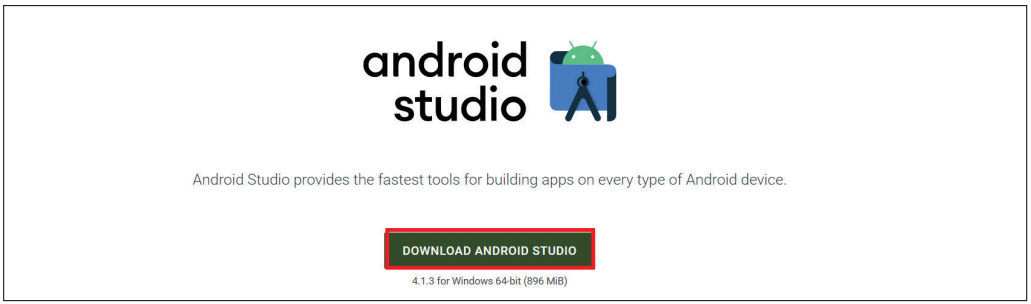
パソコンのOSについては次の環境で動かすことができます。

- Microsoft Windows 8/10のいずれか（64bit版のみ）
- macOS 10.14（macOS Mojave）以上
- GNOME、KDEまたはUnity DEによるデスクトップ環境を持つLinux（64bit版のみ）
- Chrome OS

POINT

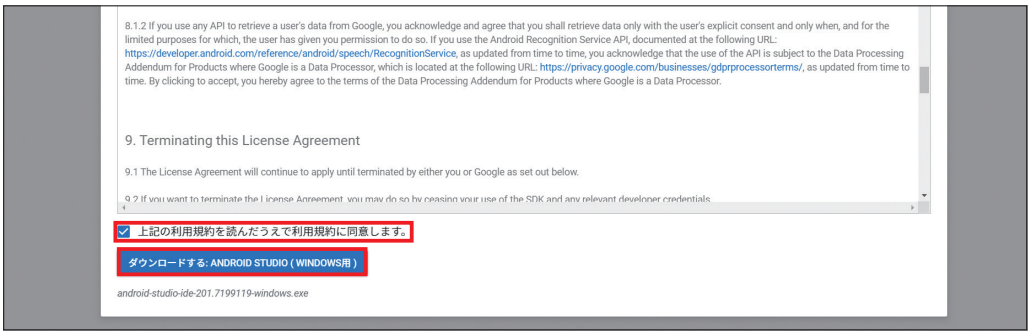
macOSの場合は、ショートカットキーや、改行文字の文字コードなどがWindowsと異なるので注意してください。本書ではできるかぎりmacOSの場合についても言及しています。

図1-4 Android Studioのダウンロードサイト



利用規約を確認し、利用規約に同意できたら、＜上記の利用規約を読んだうえで利用規約に同意します。＞をクリックしてチェックを付けます。すると、ダウンロードするためのボタンが有効になるので、＜ダウンロードする：ANDROID STUDIO（WINDOWS用）＞のボタンをクリックします（図1-5）。Macの場合は＜ダウンロードする：ANDROID STUDIO（Mac用）＞のボタンをクリックします。

図1-5 利用規約を確認するとダウンロードボタンが有効になる



すると、画面左下にダウンロード中の表示が現れます。完了するまで長い時間がかかるので、しばらく待ちます（図1-6）。

図1-6 ダウンロードの完了を待つ

ダウンロード完了後の流れは、WindowsとmacOSで少し異なります。

macOSの場合は、ダウンロードしたファイルをダブルクリックします。すると、以下のようなウィンドウが現れます（図1-7）。

SECTION 02

最初のプロジェクトを作成しよう

作りたいアプリの輪郭が見えてきたので、そろそろアプリの作り方を学んでいきましょう。Android Studioを操作することで、プログラムや素材を配置するためのフォルダやファイルのひな型が作られます。それぞれのフォルダやファイルがどのような役割を担っているのかを解説します。

Android Studioでプロジェクトを作成する

それでは、実際にAndroid Studioを使い始めます。まずは、**プロジェクト**を作りましょう。プロジェクトとは、アプリを作るために必要なファイルやフォルダをまとめて管理するためのフォルダのことです。Android Studioを操作していくことで、簡単に作成できます。Android Studioを起動すると、「Android Studioへようこそ」の画面が表示されますので、＜Create New Project＞をクリックします（図2-2）。

図2-2 Android Studioへようこそ

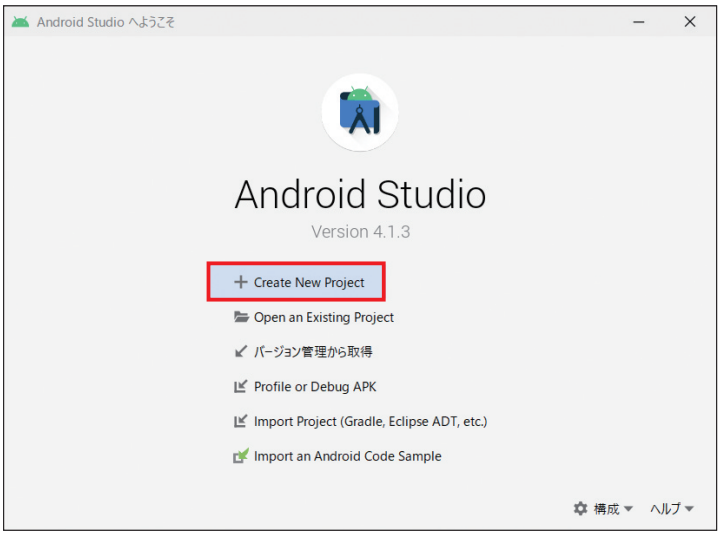
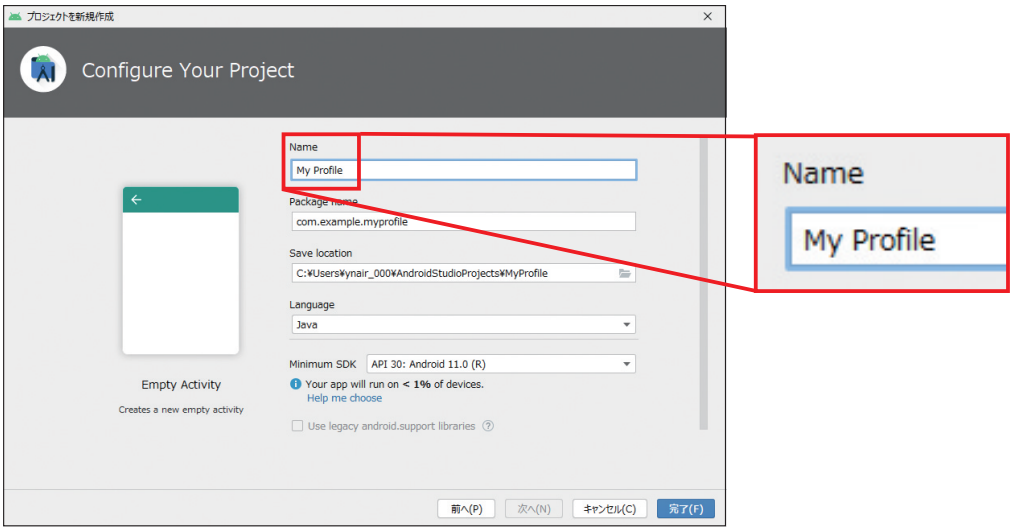


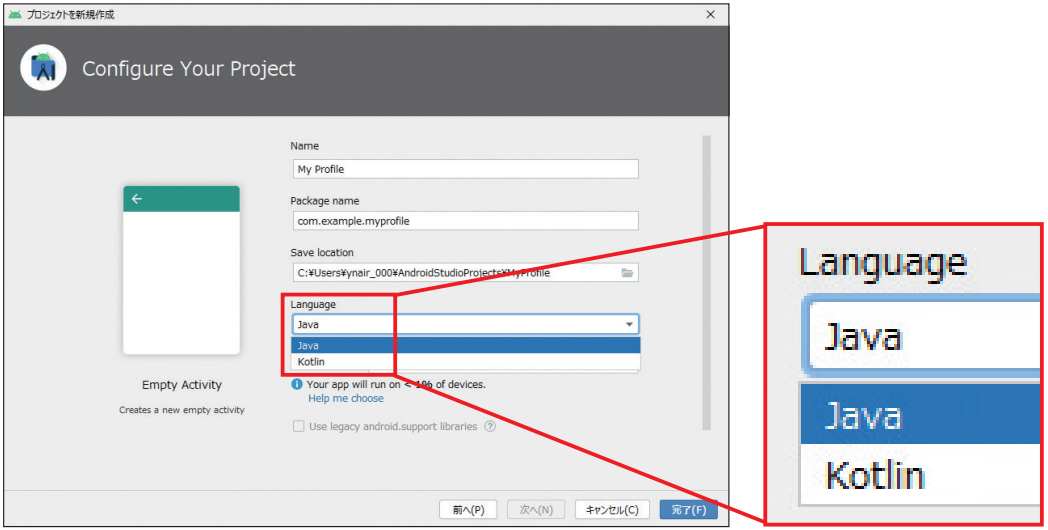
図 2-4 新規プロジェクトの作成



言語を選択する

本書では、Javaによるプログラミングを解説しているので、ここでは「Java」を選択します（図2-5）。Androidアプリは Kotlin 言語でも作成できます。本書では Kotlin でのサンプルを用意しませんが、別の書籍で Kotlin を学んでから、本書の内容をなぞってみるのも面白いかもしれません。

図 2-5 言語の選択



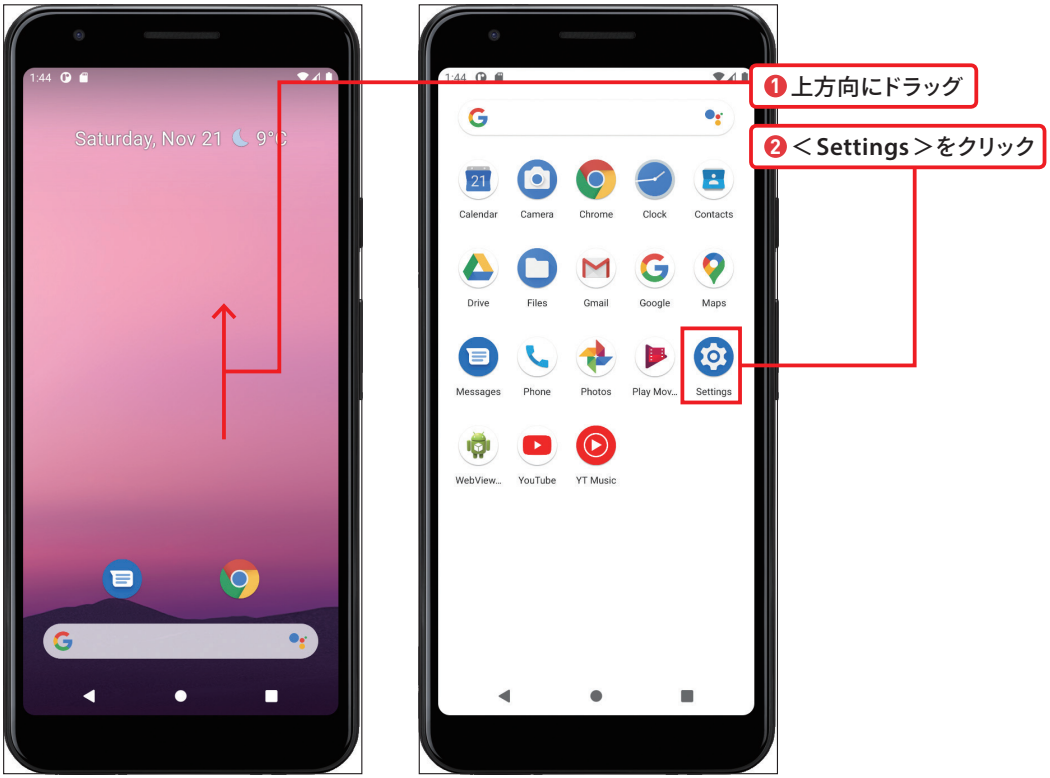
仮想デバイスを日本語化する

次は、今後の操作をしやすくするために、仮想デバイスの設定を日本語にします。通常のAndroidスマートフォンにおける言語設定と同じ方法なので、もしスマートフォンを使い慣れているようであれば、読み飛ばしても構いません。

まずは、画面上を下から上にドラッグ（クリックしたまま上方向にカーソルを動かす）して、アプリ一覧を開きます。

アプリ一覧から＜Settings＞と書かれた設定アプリを探して、クリックします（図2-18）。

図2-18 仮想デバイスが使用可能になった



次は、設定アプリのメニューを＜System＞が表示されるまでスクロールします。

＜System＞→＜Languages & input＞→＜Languages＞と進みます。この時点ではアメリカ英語しか選択できません。日本語を追加するために、＜Languages＞の画面で＜Add a language＞をクリックします（図2-19）。

SECTION 01

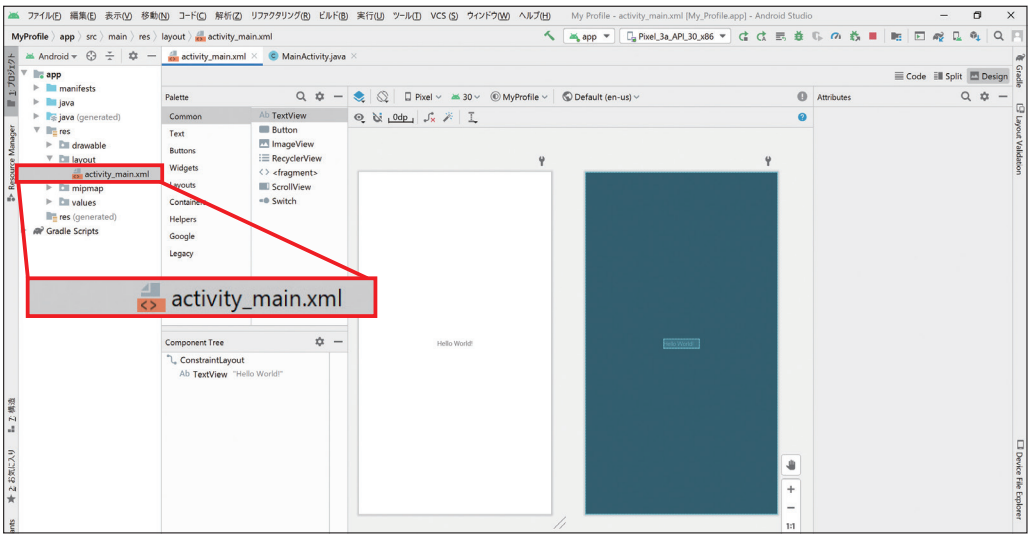
アプリのレイアウトの仕組みを知ろう

画面内の画像や文字の配置（レイアウト）は、アプリの魅力を決める重要な要素の1つです。Androidアプリ開発では、XMLという言語を用いてレイアウトの作成を行うのが主流ですが、Android Studioのレイアウトエディターを活用することで、プログラミングをほとんどすることなくレイアウトを作成できます。本節では、レイアウトエディターの使い方を解説します。

レイアウトエディター

第2章でアクティビティーを作成した際に、一緒に作られたresフォルダの中のlayoutフォルダにあるactivity_main.xmlというレイアウトファイルがありました。これをダブルクリックすると、**レイアウトエディター**でファイルが開きます（図3-1）。

図3-1 activity_main.xmlをレイアウトエディターで表示



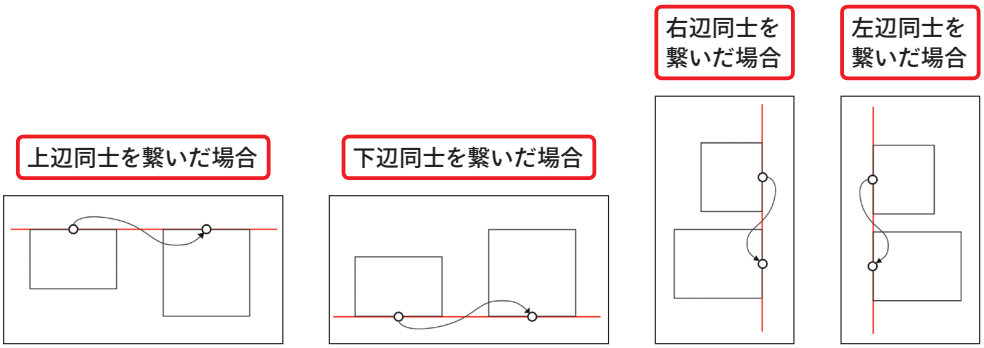
ConstraintLayoutの制約の挙動

ConstraintLayoutの制約は、ビュー同士の垂直方向（上下）や水平方向（左右）の関係性を定義することで実現されます。本項では、どう繋げるとどんな結果になるのか、主なものを解説します。

同じ側の辺を繋いだ場合

同じ側の辺を繋いだ場合、その辺をベースラインとして揃えて配置されます（図3-19）。

図3-19 同じ側の辺を繋いだ場合



異なる辺を繋いだ場合

異なる辺を繋いだ場合、部品同士が隣り合うように配置されます。このタイプの制約を行ったからといって、部品同士が接するとは限りません。レイアウトAの右辺とレイアウトBの左辺を繋いだ場合、「レイアウトAは必ずレイアウトBの左側に配置される」という制約が作られますが、隙間が空いていても特に問題はありません（図3-20）。

図3-20 異なる辺を繋いだ場合

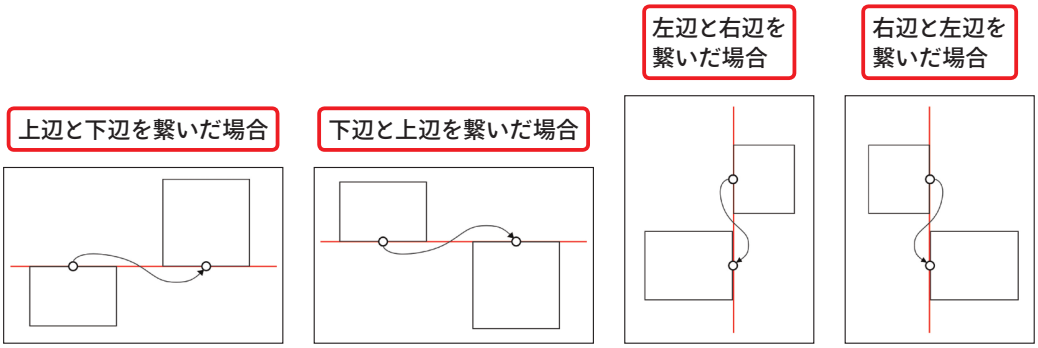
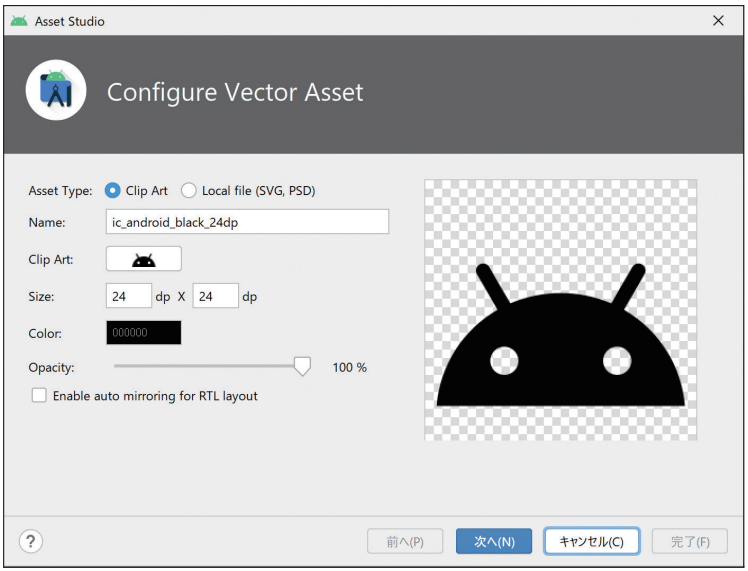


図3-65 Asset Studio



すでにドロイド君のアイコンは選択されているので、サイズを96dpに調整します。＜Size＞の行を96dp×96dpに書き替えます。また、アイコンの名前にも「24dp」の文言が入っていますが、「96dp」に書き替えておきましょう（図3-66）。

図3-66 アイコンのサイズと名前を変更する

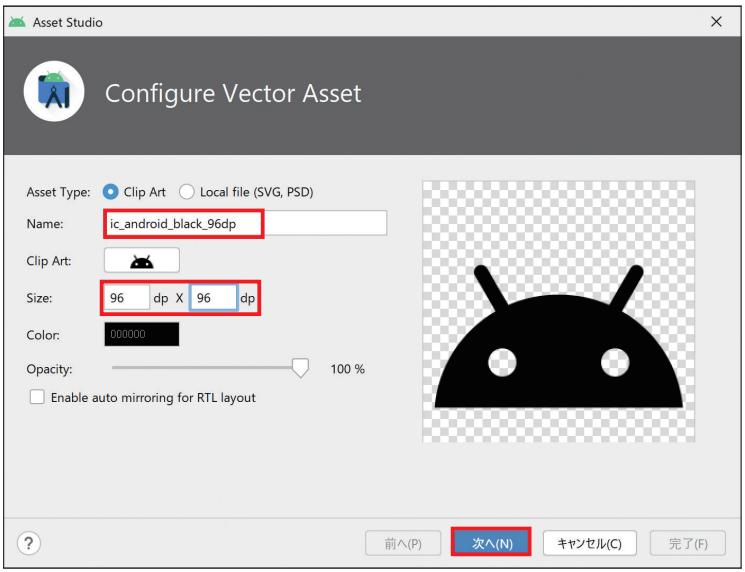
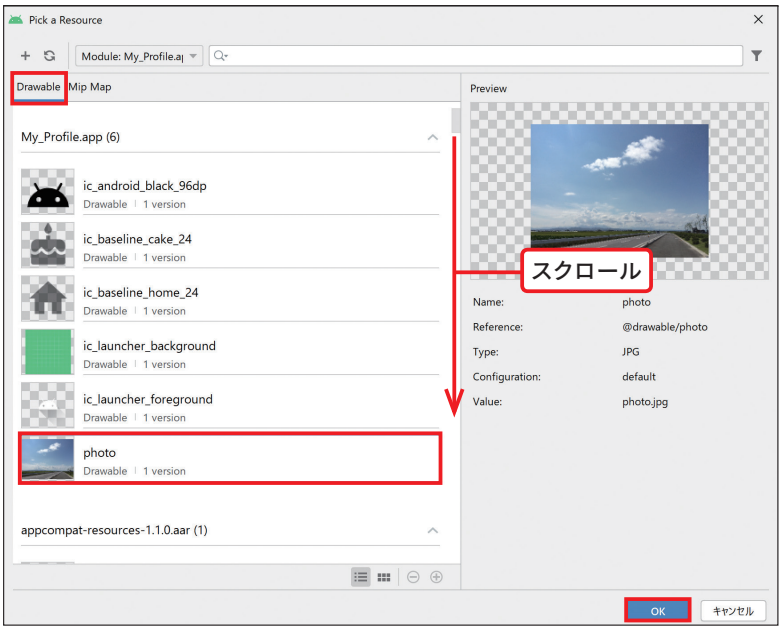
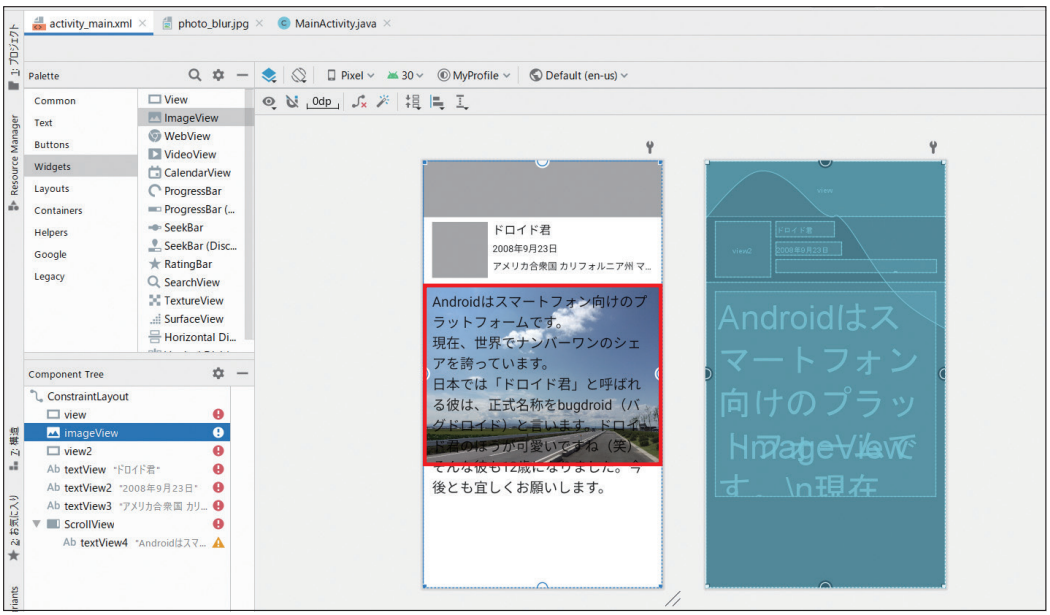


図 3-77 リソースを選択



レイアウトにImageViewが配置され、デザインエディターに写真が現れました（図3-78）。

図 3-78 写真が表示された様子



何やら灰色のバルーンが現れていますね。これは「Log.dにデータを渡す方法は2つあります」「1つはtagという名前の文字列データとmsgという名前の文字列データを渡す方法です」「もう1つはtagという名前の文字列データとmsgという名前の文字列データ、それにtrという名前のエラーデータを渡す方法です」という情報を表現しています。String msgのような表記の読み方・書き方については後述します。

今回は前者の方法を採用して、Log.dに文字列データを2つ渡すことにしましょう。次のようにカッコの中を埋めます。なお、「"MainActivity"」と入力すると「tag: "MainActivity"」と表示され、「"Hello, World!"」と入力すると「msg: "Hello, World!"」と表示されますが、そのままで構いません。

```
Log.d("MainActivity", "Hello, World!");
```

文字列を表現するために使った「"」は、ダブルクォートという1文字の文字です。シングルクォート(')を2つ書いてもエラーになりますので、間違えないようにしてください。最後にセミコロン(;)を記述するのもお忘れなく。プログラム全体としては次のような形になります(リスト4-3)。

リスト4-3 MainActivity.java

```
008:  public class MainActivity extends AppCompatActivity {
009:
010:      @Override
011:      protected void onCreate(Bundle savedInstanceState) {
012:          super.onCreate(savedInstanceState);
013:          setContentView(R.layout.activity_main);
014:
015:          Log.d("MainActivity", "Hello, World!");
016:      }
017:  }
```

今回はLog.dに渡す1つめのデータ(tag)にMainActivityという文字列を書きました。tagは、後述するログキャットでログを探しやすくするためのものです。わかりやすければどんな文字列を書いても構いませんが、クラスの名前を書くのが慣例です。

さて、これでログを表示できるようになりました。次は仮想デバイスを起動して、実際にログが表示されるところを見ていきましょう。

Logcatでログを表示する

それでは、アプリを動かして、ログが表示される様子を見ていきましょう。ログはLogcat(ログキャット)というツールウィンドウに表示されます。もし、Android Studioの下部に<Logcat>の表示があ