2024

2024 年マイコン部講習会
Android アプリコース

第 1 回講習会資料

講師　大政悠暉

目次

4章 アクティビティの追加

3章までで一通りのアプリ制作の準備ができました。これからは、デザインエディタを活用することによって、アプリの実際に触る部分であるUI（ユーザーインターフェース）を作成していきましょう。実際にUIを作成する前に、「Androidアプリの画面」であるアクティビティについて少し学びます。

4.1 アクティビティとは

Androidアプリを作成する上で、土台となるものが「アクティビティ」です。ここでは、アクティビティの内容と、簡単な使い方を紹介します。

4.1.1 定義と役割

アクティビティ(Activity)は、Androidアプリケーションの基本的な部分で、ユーザーが見たり操作したりする画面のことを指します。例えば、スマホのメールアプリでメールを読む画面や、カメラアプリで写真を撮る画面がアクティビティです。

アクティビティは、ユーザーがアプリを使う際の画面を表示し、ユーザーの操作を受け付ける役割を持っています。アクティビティは、アプリの画面同士をつなげる役割も果たしており、ユーザーがある画面から別の画面に移動するのを助けます。

4.1.2 アクティビティライフサイクル

アクティビティには「ライフサイクル」と呼ばれる一連の状態変化があります。これにより、アクティビティがどのように作成され、表示され、停止され、破棄されるかが管理されます。以下は、アクティビティの作成や開始、停止などのタイミングで実行されるライフサイクルの主要なステップです。

命令（メソッド）	内容	説明
onCreate()	生成	クティビティが最初に作成されるときに呼ばれます。このとき、画面の初期設定を行います。
onStart()	開始	アクティビティが画面表示される直前に呼ばれます。
onResume()	操作開始	アクティビティがユーザーとやり取りを開始する直前に呼ばれます。
onPause()	一時停止	アクティビティが一時停止する直前に呼ばれます。このとき、必要なデータを保存します。
onStop()	停止	アクティビティが画面から消え、バックグラウンドになる直前に呼ばれます。

onRestart()	再表示	Activity が非表示の状態から再度表示される際に呼び出されます。この後は onStart()が呼び出されます。
onDestroy()	終了	アクティビティが破棄される直前に呼ばれます。このとき、使用していたリソースを解放します。

4.2 新しいアクティビティの追加方法

今回は、3 章までで使用したアクティビティとは別のアクティビティを使用するため、新しくアクティビティを作成します。

4.2.1 アクティビティの追加

1. アクティビティを追加する

   まずは新たなアクティビティを追加します。以下の手順を行ってください。

   - ✓ Android Studio のプロジェクトビューで、app フォルダを右クリック
   - ✓ 「New」→「Activity」→「Empty Activity」を選択
   - ✓ 新しいアクティビティの設定ウィンドウが表示

2. 設定を行う

   次に設定を行います。各項目を以下のように設定してください。

設定	内容
アクティビティ名	アクティビティの名前を入力します。 ここでは「MyActivity」とします。
レイアウト名	レイアウトファイルの名前を入力します。 ここでは同じように「MyActivity」と入力します。
言語	Java を選択します。
Launcher Activity	最初に起動するアクティビティかどうかの設定です。 これを**オン**にします。

   すべての設定が終了したら、「Finish」ボタンをクリックします。

3. ファイルを確認する

   ボタンをクリックしたら、以下のファイルの生成を確認してください。

   - MyActivity.java
   - MyActivity.xml

   うまくいけば新しいアクティビティの作成は完了です。お疲れさまでした。

4.2.2 定義ファイルの内容

AndroidManifest.xml は、アプリケーションの基本的な情報を定義するファイルです。アプリの名前、アイコン、許可(パーミッション)などが記述されています。ここでは、アクティビティの定義について説明します。

AndroidManifest.xml ファイルの場所

- プロジェクトビューで、app フォルダの中にある src/main フォルダを展開します。
- AndroidManifest.xml ファイルを見つけて開きます。

アクティビティの定義

新しいアクティビティを追加した場合、自動的に AndroidManifest.xml にアクティビティの情報が追加されます。以下は、MainActivity に加えて SecondActivity が追加された状態の例です。今回は 3 章で作成したファイルと先ほど作成した MyActivity があります。

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">

    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".SecondActivity"></activity>
</application>
```

AndroidManifest.xml にはいくつか約束があります（覚えなくても構いません）

1. マニフェストは規定された要素のみで独自の要素や属性を追加することはできない
2. マニフェストには <manifest> と <application> の要素は必須

3. <manifest> と<application> の要素の記載は AndroidManifest 内で 1 回のみ
4. <application> 要素は <manifest> 要素内に含む

起動アクティビティの変更方法

アプリを起動したときに表示される最初のアクティビティを変更するには、intent-filter を使います。上記の例では、MainActivity が起動アクティビティとして設定されています。これを SecondActivity に変更するには、intent-filter を SecondActivity に移動します。今回は、先ほどアクティビティを作製した際に Launcher Activity のチェックを ON にしたため、既に MyActivity が起動アクティビティになっています。

```
    <activity android:name=".SecondActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".MainActivity"></activity>
</application>
```

このようにして、アクティビティの追加や管理方法、起動アクティビティの設定を理解することができます。これで、新しいアクティビティを追加し、アプリケーションの挙動をカスタマイズする準備が整いました。

5 章 デザインエディタの使い方

5.1 デザインエディタの概要

ここでは、これまでにも何回か登場したデザインエディタの詳しい使い方を解説します。また、Android アプリの画面に表示される要素の部品であるビューの説明と、それらの配置、設定方法を説明します。

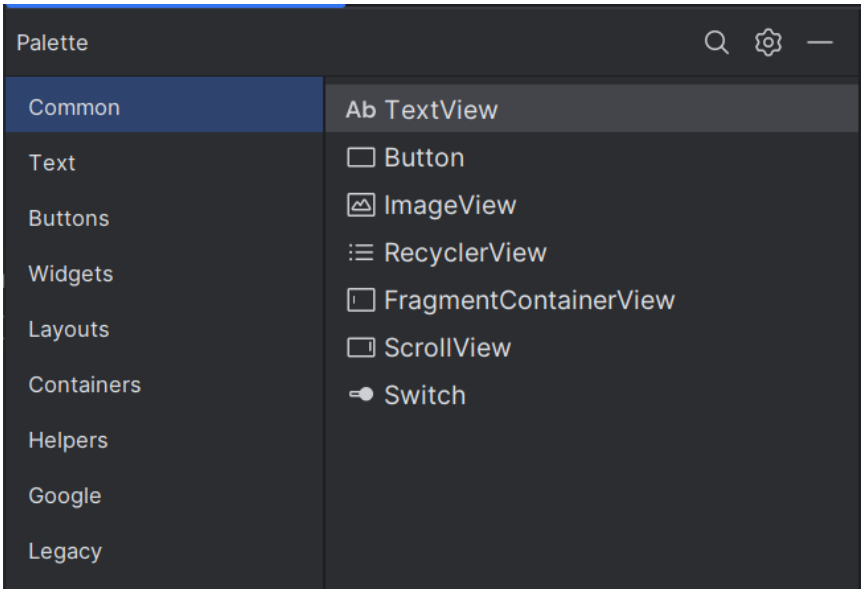
5.1.1 デザインエディタとは

デザインエディタは、Android Studio でアプリの見た目を作成するためのツールです。このエディタを使うことで、コードを直接書かずに視覚的にレイアウトや UI（ユーザーインターフェース）を作成できます。UI とは、ユーザーがアプリを使うときに操作する画面やボタン、入力フィールドなどのことです。

5.1.2 各部分の説明

パレット

パレットは、さまざまな UI 部品を一覧表示する部分です。ここからドラッグ＆ドロップで部品をレイアウトに追加することができます。例えば、ボタンやテキストボックスなどがパレットにあります。

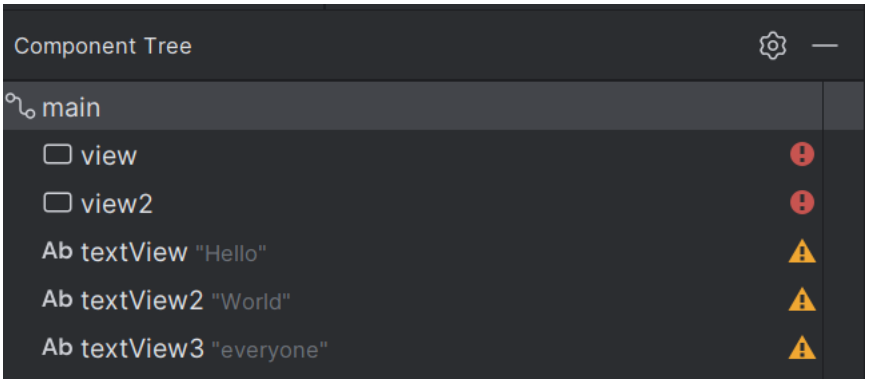


パレットの中のビュー（UI 部品）はジャンルごとに分けられており、それぞれの内容は以下のようになります

ジャンル	ビューの説明
Common	よく使う基本的なビュー
Text	文章を表示するテキストビュー（平文用、数字用、パスワード用など）
Buttons	ボタンを表示するビュー（通常のボタンやチェックボックス、ラジオボタン、スイッチなど）
Widgets	そのほかの画面に配置するビュー（シークバーや画像のビュー、WEB ビュー、カレンダー、動画など）
Layouts	ビューの配置（横向き、縦向き、表、スペース空け）など
Containers	他の要素の入れ物（ツールバー、リスト、スクロールバーなど）
Helpers	お助け機能（補助線、仮の要素の配置、画像の補正など）
Google	広告のビューと GoogleMap のビュー
Legacy	昔のバーションのビュー

コンポーネント・ツリー

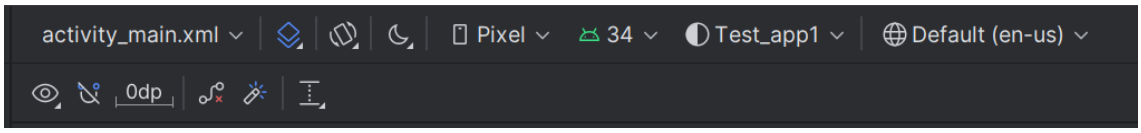
コンポーネント・ツリーは、現在のレイアウトに追加された全ての UI 部品を階層的に表示します。ここで部品を選択して、配置や属性を変更することができます。



後ほど説明するビューグループなどで囲うと、対象となるビューがインデント（少し右側にずれる）されて表示されます。このようにして、どの要素がどのグループに含まれているのかを分かりやすく表示しています。

ツールバー

ツールバーには、レイアウトを編集するための様々なツールが配置されています。例えば、ズームインやズームアウト、デザインビューとコードビューの切り替えなどの機能があります。



ツールバーの詳細な説明は以下の通りです。

上段左から順に

1. ファイル選択

   編集しているファイルを変更することがします

2. デザイン/ブループリント

   デザインエディタでレイアウトを表示する方法をリストから選択します

3. 画面の向き

   プレビュー表示の画面を縦向き、横向きに変更します

4. 端末のタイプとサイズ

   プレビュー表示の端末タイプ（どのスマホを使用するか）を変更します

5. API バージョン

   プレビュー表示の API レベル（Android のバージョン）を選択します

6. アプリのテーマ

   プレビューに適用する UI テーマを選択します

7. 言語

   アプリを多言語対応させている場合、表示する文字列の言語を選択します

下段では、左から順に

8. ビューオプション

   プレビューに表示させる内容を変更します

9. 親要素への自動結合の表示/非表示

   自動的に結合された要素の結合の表示または非表示を選択します

10. 基本マージン

アクティビティ内で使用される基本マージン（ほかのビューとの余白）を設定します

11. Constraint（制約）の削除

    アクティビティ内の制約（ビューの配置の決まり※6 章で詳しく解説）を削除します

12. 制約の推測（自動追加）

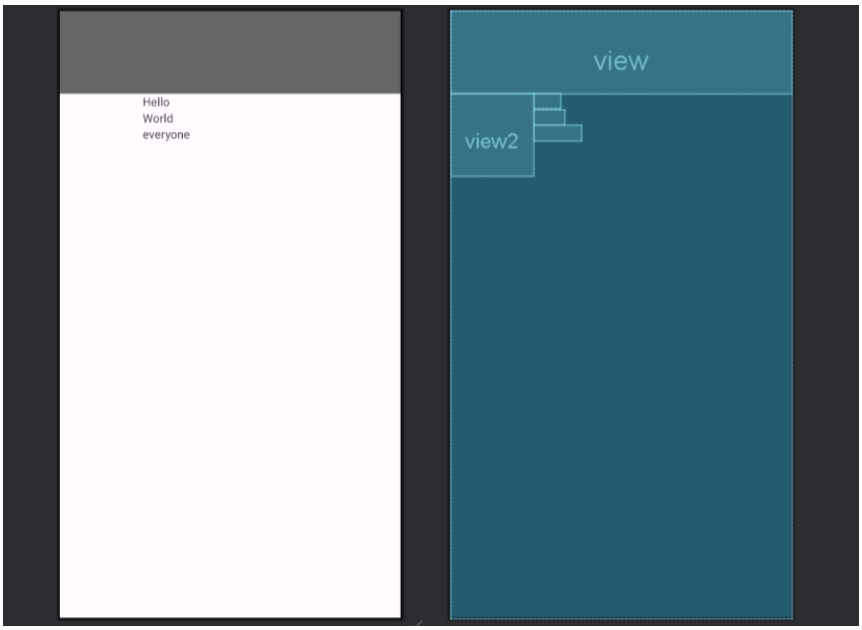
    現在プレビューに配置されているビューの制約を推測して追加します

13. グリッドライン

    ビューを配置する際の補助線を追加します

デザインエディタ

デザインエディタは、実際に UI 部品を配置してレイアウトを作成する部分です。ここにパレットから UI 部品をドラッグ＆ドロップして、画面の見た目を直感的に作成します。



- デザインビュー（Design View）

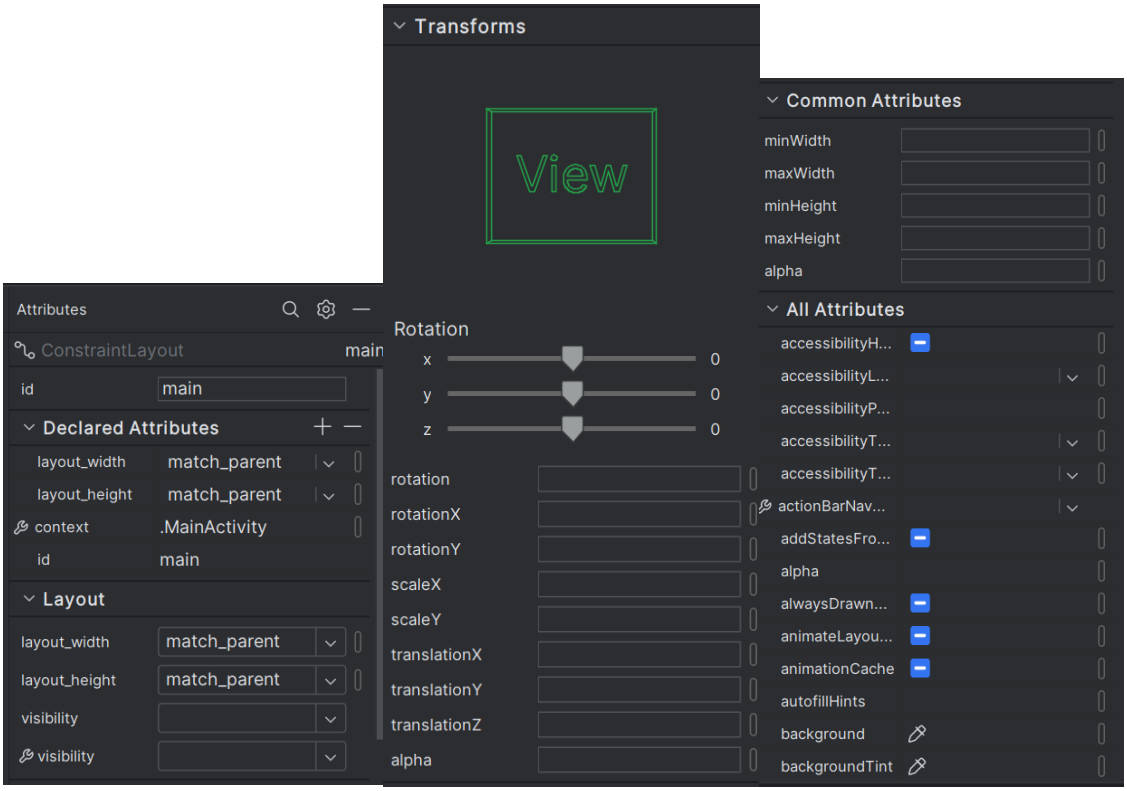
  実際にアプリを実行した画面とほぼ同じものが表示されているため、実際の見え方を確認するときに役立つ。

- ブループリントビュー（Blueprint View）

  各画面部品の枠線だけが表示されているため、単なる画面部品の配置だけを確認したい場合ブループリントビューの方がわかりやすい。

属性（プロパティ）

属性は、選択した UI 部品の細かい設定を行う部分です。例えば、ボタンのテキストや色、位置などをここで変更することができます。



ビュー属性のプロパティの詳細は以下の通りです。

- Declared Attributes

  ビューの ID や大きさ、内容など、指定されたビューなど何かしら変更されている設定内容を変更できます。（初期状態で変更されている設定も表示されます）

- Layout

  ビューの高さや幅などの設定を行うことができます。また、該当のビューに関する制約が一覧で表示されます。

- Common Attributes

  選択したビューの一般的な属性が一覧表示されます。大体の属性はここから編集することができます。

- All Attributes

利用可能なすべての属性を表示します。Common Attributes に表示されていない属性は、ここから編集することができます。

- Search

  プロパティの一番上の虫メガネのマークから使用でき、プロパティの項目名を入力することによって特定のビュー属性を検索できます。

※1. 属性値の周囲が赤色でハイライト表示されている場合は、値にエラーがあることを示します。たとえば、レイアウト定義属性に無効な入力がされていることを示すエラーが考えられます。また、オレンジ色のハイライトは、この値に関する警告を示します。たとえば、リソース参照が想定されるハードコードされた値を使用すると、警告が表示されることがあります。

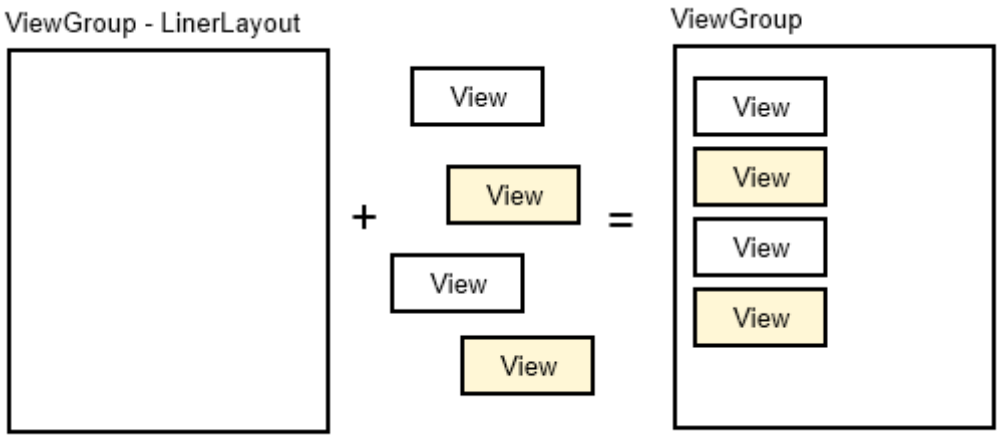
※2. 各属性値の右側にあるアイコンは、その属性値が直接入力されているかどうかを表します。自動で値が入力されているときは中身が埋まっており、値が直接入力されている場合は空になります。

5.2 ビューグループとは

5.2.1 定義と役割

ビューグループ(ViewGroup)は、複数のビュー（UI 部品）をまとめて管理するためのコンテナです。ビューグループは、子ビューの配置やサイズを決める役割を持っています。ビューグループを使うことで、複雑なレイアウトを簡単に作成することができます。

具体的には、以下のようにビューグループを使うことで複数のビューを整列させることができます。



出典：JavaDrive　ビューとビューグループ

5.2.2 主なビューグループ

Linear Layout（リニアレイアウト）

すべての子ビューを一列に並べるレイアウトです。縦方向（垂直）または横方向（水平）にビューを並べることができます。例えば、ボタンを縦に並べたり、テキストとボタンを横に並べたりする場合に使います。

Talbe Layout（テーブルレイアウト）

ビューを表形式に並べるレイアウトです。行と列で構成され、各セルにビューを配置します。フォームや表形式のデータを表示する場合に使います。

Frame Layout（フレームレイアウト）

左上を原点にビューを重ねるレイアウトです。複数のビューを重ねる場合に使用します。画像の上にテキストを重ねたり、重なったビューを使ったりする場合に便利です。

Relative Layout（リラティブレイアウト）

子ビューを他のビューや親ビューに対して相対的に配置するレイアウトです。例えば、ボタンをテキストの下に配置したり、画像を画面の中央に配置したりする場合に使います。

Constraint Layout（コンストレイントレイアウト）

Constraint Layout は、子ビュー同士の位置関係を制約（Constraint）として設定することで、複雑なレイアウトを作成できるビューグループです。例えば、ボタンを画面の中央に配置したり、テキストをボタンの右側に配置したりできます。

5.3 レイアウトの制約

レイアウトの制約は、ビューの位置やサイズを他のビューや親ビューに対して設定するルールのことです。制約を使うことで、ビューが他のビューとどのように関係するかを決めることができます。

5.3.1 レイアウトの制約の設定方法

レイアウトエディタでビューを選択し、ドラッグして他のビューや親ビューに対して位置を調整します。制約線（青い線）を使って、ビューの位置関係を決めます。例えば、ボタンを画面の左上に固定する場合、その位置に制約を設定します。

5.4 ビューの配置

5.4.1 ビューの配置方法

ビューをレイアウトに配置するには、デザインエディタでビューをドラッグ＆ドロップして配置します。レイアウトに応じてビューを配置する場所を決定します。例えば、LinearLayout では縦か横に並べて配置し、ConstraintLayout では制約を設定して配置します。

5.4.2 必要なビューの配置

今回は、入力ボックス（EditText）とボタン（Button）、テキストビュー（TextView）スイッチ（Switch）を使って自己紹介アプリを作成します。以下の手順でビューを配置します。また、今回はビューグループを使用せず直接ビューを配置します

1. 入力ボックス（Text カテゴリ→PlainText）

   名前を入力するための EditText を配置します。属性パネルでヒント（Hint）を「名前を入力」に設定します。

2. ボタン（Button カテゴリ→Button）

   「実行」ボタンを配置します。属性パネルでテキストを「送信」に設定します。

3. テキストビュー（Text カテゴリ→TextView）

   メッセージを表示するためのテキストを配置します。テキストには「こんにちは！」と設定します

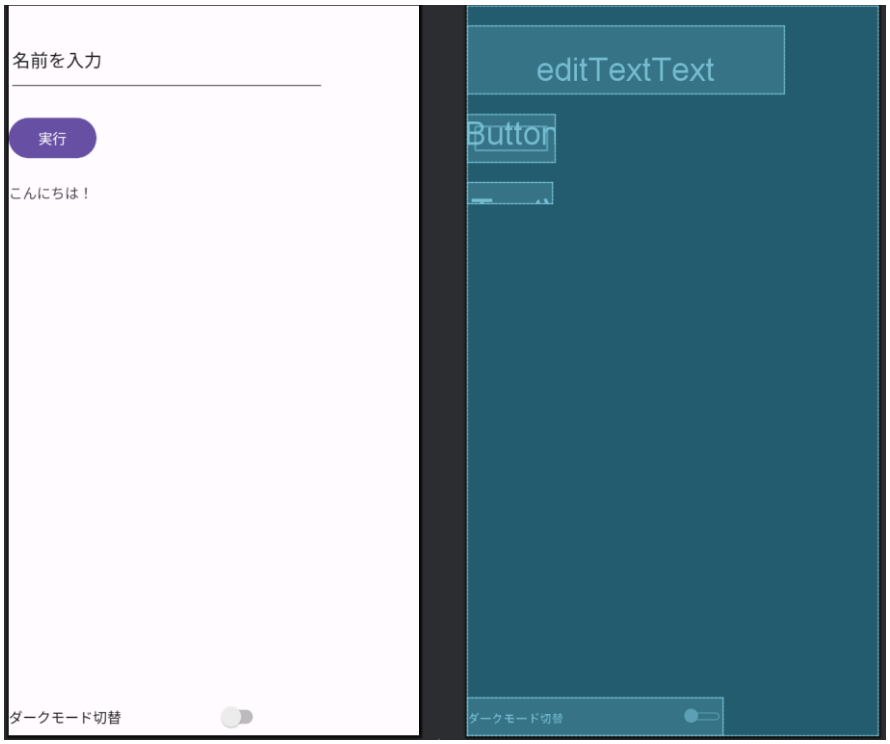
4. スイッチ（Button カテゴリ→Button）

   プロフィールの表示方法を切り替えるための Switch を配置します。属性パネルでテキストを「ダークモード切り替え」に設定します。

以上のビューを配置し、制約とプロパティを設定して完成です。

Parent（ビューをくっつけられる元となるビュー）は、縦と横を画面の外側にくっつけてください。

うまく配置できれば以下の画像のようになります。



6 章イベント処理の基礎

6.1 イベント処理について

イベント処理とは、ユーザーがアプリを操作したときに特定の動作を実行する仕組みのことです。例えば、ボタンをクリックしたときやスイッチをオンにしたときに特定の処理を行います。イベント処理を設定することで、ユーザーがアプリを使いやすくなります。

6.2 ビュー（UI 部品）の種類

先ほどから何度か出てきているビューのおさらいをしましょう。ビュー（UI 部品）は、アプリの画面上でユーザーが操作できる要素のことです。代表的なビューには、ボタン、スイッチ、入力ボックスなどがあります。これらのビューは、ユーザーの操作に応じてアクションを起こすためにイベント処理を設定することができます。デフォルトで用意されている主なビューには以下のものがあります.

ビュー	内容
TextView	文字列の表示
EditView（Plain Text）	入力ボックス
Button	ボタン
RadioButton	ラジオボタン
CheckBox	チェックボックス
Spinner	ドロップダウンリスト
ListView	リスト表示
SeekBar	スライダー
Switch	ON・OFF のスイッチ
Scroll View	スクロールバー
ImageView	画像の表示
ProgressBar	進捗バー
ToolBar	ツールバー

6.3 ボタンのクリックイベント

ボタンをクリックしたときに特定のアクションを実行するには、ボタンをクリックした際に発生するイベントであるクリックイベントを設定します。以下に、ボタンのクリックイベントの基本的な設定方法を説明します。下記の例では、ボタンがクリックされたときに「ボタンがクリックされました」というメッセージが表示されます。今回は以下の例に従って作業を実施してください。

1. レイアウトファイルにボタンを配置する

   先ほど配置したボタンの XML を見てみましょう。中身は以下のようになっており、id はビューを判別するために個別につけられている ID、layout_width/height は配置場所、text はボタンに表示される文字の内容を示しています。今回、text は「送信」としておきましょう。

```
<Button
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="送信" />
```

2. アクティビティファイルでボタンを取得し、状態変更イベントを設定する

   ボタンを押したときの処理は次章以降に説明します

6.4 スイッチの状態変更イベント

スイッチのオン/オフが変更されたときに特定のアクションを実行するには、状態変更イベントを設定します。以下に、スイッチの状態変更イベントの基本的な設定方法を説明します。下記の例では、スイッチがオンになったときに「スイッチがオンになりました」、オフになったときに「スイッチがオフになりました」というメッセージが表示されます。今回は以下の例に従って作業を実施してください。

1. レイアウトファイルにボタンを配置する

   次はスイッチの XML です。基本的にはボタンと似ています。今回 text は「ダークモード切り替え」としてください。

```
<Switch
    android:id="@+id/switch_profile"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="プロフィール表示" />
```

2. アクティビティファイルでボタンを取得し、状態変更イベントを設定する

今度はアクティビティの設定を行います。こちらもボタンと似ていますが、スイッチの場合はオンになった時の処理とオフになった時の処理の両方を記述します。

7 章 実践練習：自己紹介アプリの作成

7.1 アプリの設計

7.1.1 アプリの概要

今回作成するアプリは、名前を入力してボタンを押すと「〇〇さん、こんにちは！」と表示されるシンプルな自己紹介アプリです。また、スイッチでアプリのダークモードとライトモードを切り替える機能も追加します。

7.1.2 必要なビュー

今回のアプリに必要なビューはとその説明は以下の通りです。

ビュー	内容
EditText（入力ボックス）	名前を入力するためのボックス
Button（ボタン）	名前を入力した後に押すボタン
TextView（テキスト表示）	入力された名前とメッセージを表示するためのテキストビュー
ImageView（画像表示）	プロフィール画面を表示させるための画像ビュー
Switch（スイッチ）	ダークモードとライトモードを切り替えるためのスイッチ
RelativeLayout（レイアウト）	すべての UI 部品を配置するためのレイアウト

7.1.3 アプリの画面設計

アプリのレイアウトを決定し、各 UI 部品の配置を行います。シンプルで使いやすいデザインにするために、次のように配置していきます。

- 上部に EditText を配置して名前を入力させる
- その下に Button を配置し、名前を利用して挨拶を出力する
- さらに下に TextView と ImageView を配置して自己紹介を作成する
- 最下部に Switch を配置して、ライトモードとダークモードの切り替えを行う

7.2 アプリの実装

7.2.1 レイアウトファイルの作成

activity_main.xml ファイルに各 UI 部品を配置していきます。

7.2.2 イベント処理の実装

MainActivity.java ファイルにイベント処理を実装していきます。

7.2.3 プログラムの簡単な解説

onCreate()メソッド

アクティビティが起動したときに呼び出されるメソッドです。ここで UI 部品の初期化とイベントリスナーの設定を行います。

buttonGreet.setOnClickListener()メソッド

ボタンがクリックされたときに呼び出される処理を設定します。入力された名前を取得し、TextView に「○○さん、こんにちは！」と表示します。

switchTheme.setOnCheckedChangeListener()メソッド

スイッチの状態が変わったときに呼び出される処理を設定します。スイッチがオンの場合はダークモード、オフの場合はライトモードにテーマを変更し、アクティビティを再生成します。

7.2.4 アプリのテスト

実機やエミュレータでアプリをテストし、各機能が正しく動作することを確認します。うまく動かない場合は以下の場合が考えられます。