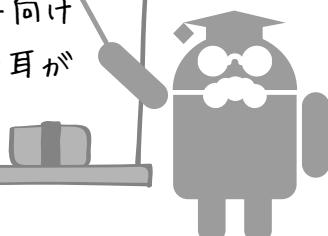


# 第 20 章

## ローカライズと アクセシビリティ

著：深見浩和

本章では、ローカライズとアクセシビリティについて解説します。ローカライズとは、作成したAndroidアプリを特定の地域（日本語圏→海外、または海外→日本語圏）のユーザー向けに提供することで、アクセシビリティとは、高齢者や目や耳が不自由なユーザー向けに行うべきことです。



## この節で学ぶこと

- ・言語別の文字列リソースを用意する方法
- ・言語別の画像リソースを用意する方法
- ・TalkBack機能の使い方
- ・TalkBack機能が有効な時、適切なナビゲーションを行う方法
- ・アプリ内で再生する動画に字幕をつける方法

## この節で出てくるキーワード一覧

言語コード

TalkBack



## 20-1 なぜ多言語対応が重要か

「このアプリ、すごくいいんだけど英語なんだよな」といった経験はありませんか？ すばらしいユーザー体験を提供するアプリであっても、言語が母国語でないだけでインストールされなかったりします。そこで、日本語を母国語としないユーザーにもアプリを使ってもらうため、翻訳などの対応を行いましょう。この対応を「ローカライズ」と呼びます。本章では端末の言語設定に応じて文字列や画像を変えることをローカライズと呼ぶことにします。



## 20-2 言語コード

言語の名称の略号は、「ISO 639」という国際規格で決められています。**表1**にその一部を示します。Androidでは、言語の指定にこの言語コードを使用します。

言語名	コード
日本語	ja
英語	en
中国語	zh
スペイン語	es

表1:ISO 639で定義されている言語コード



## 20-3 文字列をローカライズする

Androidには、アプリ名やTextViewで表示する文字列を端末の言語設定に従って切り替える仕組みがあります。この仕組みを利用するには、次の手順で文字列をリソース参照に置き換えていきます。

- 1.res/values/strings.xmlを作成する
- 2.res/values-en/strings.xmlなど別の言語リソースを作成する
- 3.レイアウトXML内の文字列参照を@string/xxxxに変更する
- 4.Javaプログラム内の文字列をgetString(R.string.xxxx)で取得したも  
のに変更する

では、順に見ていきましょう。

### res/values/strings.xmlを作成する

まず、「strings.xml」にアプリケーションで利用する文字列の一覧を作成します。次はアプリ名とメッセージの対応を記述した例です。name属性を用いて文字列(メッセージ)に名前を付けていきます。この時、<resources>の下に、<string name="文字列の名前">文字列</string>という形式で対応を記述します。

- ・数字で始まる名前
- ・Javaの予約語(たとえばswitchなど)

は使用することができないので注意しましょう。

### strings.xmlの例

```
<resources>
    <string name="app_name">メモ帳アプリ</string>

    <string name="error_failed_to_connect">サーバーに接続できませんでした。</string>
    <string name="settings">設定</string>
</resources>
```

## res/values-en/strings.xmlなど別言語リソースを作成する

次に、以下の手順で各言語用のstrings.xmlを用意します。

- 1.res/values-<言語コード>フォルダーを作ります。たとえば英語であれば  
res/values-enフォルダーを作成する
- 2.res/values/strings.xmlをres/values-<言語コード>フォルダーにコピーする
- 3.コピーしたstrings.xmlファイルの文字列をその言語に翻訳する

これにより、端末の言語設定に応じて参照するstrings.xmlファイルが変更されます。なお、「res/values」フォルダー内のstrings.xmlはデフォルトリソースと呼ばれ、現在の言語設定に対応したstrings.xmlファイルが見つからない時に使用されます。次は英語版を用意した例です。

### 英語用strings.xmlの例

```
<resources>
    <string name="app_name">NotePad App</string>

    <string name="error_failed_to_connect">Failed to connect to the server.</string>
    <string name="settings">Settings</string>
</resources>
```

## レイアウトXML内の文字列参照を@string/xxxxに変更する

文字列リソースを用意した後、レイアウトXML内に現れる文字列(Text ViewやButtonのandroid:textなど)を@string/xxxx形式によるリソース参照に置き換えていきます。xxxxの部分は、先ほど作成したstrings.xmlのname属性に設定した名前を指定します。たとえば、次では、Text Viewに「メモ帳アプリ」が表示されます。

## TextViewの文字列を置き換える

```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/app_name"/>
```

## Javaプログラム内の文字列をgetString(R.string.xxxx)で参照する

Toastの引数やダイアログのメッセージなど、Javaプログラム内でユーザーに表示するために文字列を指定するシーンはいくつかあります。端末の言語設定を反映させるため、Javaプログラム中では文字列を""で指定するのではなく、次のような方法で指定します。

- int resourceIdを引数にとるメソッドの場合、R.string.xxxxを渡す
- StringやCharSequenceを引数にとるメソッドの場合、getString(R.string.xxxx)の戻り値を渡す

TextViewのsetTextメソッドやAlertDialog.BuilderのsetMessageメソッドなど、一部のメソッドは表示する文字列としてCharSequence(String)だけでなく、int型のresourceIdを指定することができます。この場合、引数としてR.string.xxxxを渡すことで多言語に対応させます。xxxxの部分はstrings.xmlのname属性に設定した名前にします。次はAlertDialogでエラーメッセージを表示する例です。

## Javaプログラムで文字列リソースを使う例

```
@Override  
public Dialog onCreateDialog(Bundle savedInstanceState) {  
    Activity activity = getActivity();  
    if (activity == null) { return null; }  
  
    AlertDialog.Builder builder = new AlertDialog.Builder(activity);  
    // タイトルとメッセージをR.string.xxxxで指定  
    builder.setTitle(R.string.error);  
    builder.setMessage(R.string.error_failed_to_connect);  
    builder.setPositiveButton(android.R.string.ok, null)  
    return builder.create();  
}
```

セットする文字列としてStringやCharSequenceしか受け取れないメソッドや、文字列の一部をユーザー名などに置き換えて使用したい場合、ActivityやFragmentのgetString(int resourceId)メソッドを使用して、端末の言語設定に従った文字列を取得し、それを使用します。次は「%ls」の部分をユーザー名に置き換えて

Toastメッセージを表示する例です。

#### getString()を使う例

```
private void showWelcomeMessage(String username) {  
    // msg_welcomeは次の通り  
    // <string name="msg_welcome">ようこそ %1s さん!</string>  
    String msg = getString(R.string.msg_welcome, username);  
    Toast.makeText(this, msg, Toast.LENGTH_SHORT).show();  
}
```



## 20-4 画像をローカライズする

アプリのアイコンやロゴなど、文字列を画像で用意することもあるでしょう。Androidでは、画像も端末の言語設定に応じて切り替える仕組みがあります。この仕組みを利用するには、ローカライズした画像ファイルを用意し、res/drawable-`<言語コード>-<ピクセル密度>`フォルダーに同じファイル名で入れます。



## 20-5 デフォルトの言語を英語にする

res/valuesフォルダーや、res/drawable-hdpiフォルダーなど、言語コードを含まないフォルダー内のリソースは「デフォルトリソース」と呼ばれます。これらは、端末の言語設定に一致する言語コードのフォルダーが存在しない時に使用されます。つまり、res/values(strings.xml)に日本語のリソースをいれた場合は、サポートしていない言語のユーザーには日本語が表示されてしまい、すぐアンインストールされてしまいます。世界的に見て、英語であれば多少は読めるというユーザーが多いため、res/valuesには英語リソースを入れ、res/values-jpに日本語リソースを入れておきましょう。これにより、サポートしていない言語の場合でも英語で表示されるため、開発したアプリを使ってもらえる可能性が高まります。



## 演習問題

res/values-en(strings.xml)を用意し、英語版アプリ名(app\_name)を作成してみましょう。作成後、端末の言語設定を英語に変更し、アプリ名が英語になっていることを確認しましょう。



## 20-6 アクセシビリティとは

アクセシビリティとは、本来、高齢者や障害者を含む誰もが製品・サービスを問題なく利用できるかどうかの度合いを表すものです。Androidでは目や耳が不自由な方のアプリ利用をサポートする仕組みが備わっています。これをアクセシビリティ機能と呼び、アプリは適切な対応を行う必要があります。本節以降で対応方法を紹介します。



## 20-7 アクセシビリティ機能を試す

まず、アクセシビリティ機能がどのようなものか体験してみましょう。Androidでは標準で目の不自由な方向けにTalkBack(図1)というアプリがインストールされています。「設定」→「ユーザー補助」→「TalkBack」を選択し、TalkBackをONにしてみましょう。

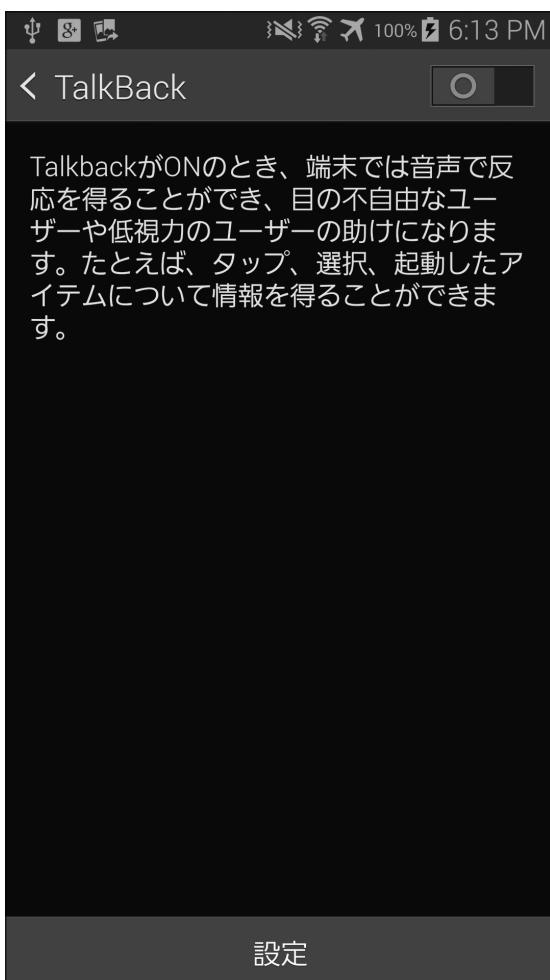


図1:TalkBack

図2:フォーカスが当たった状態でダブルタップする

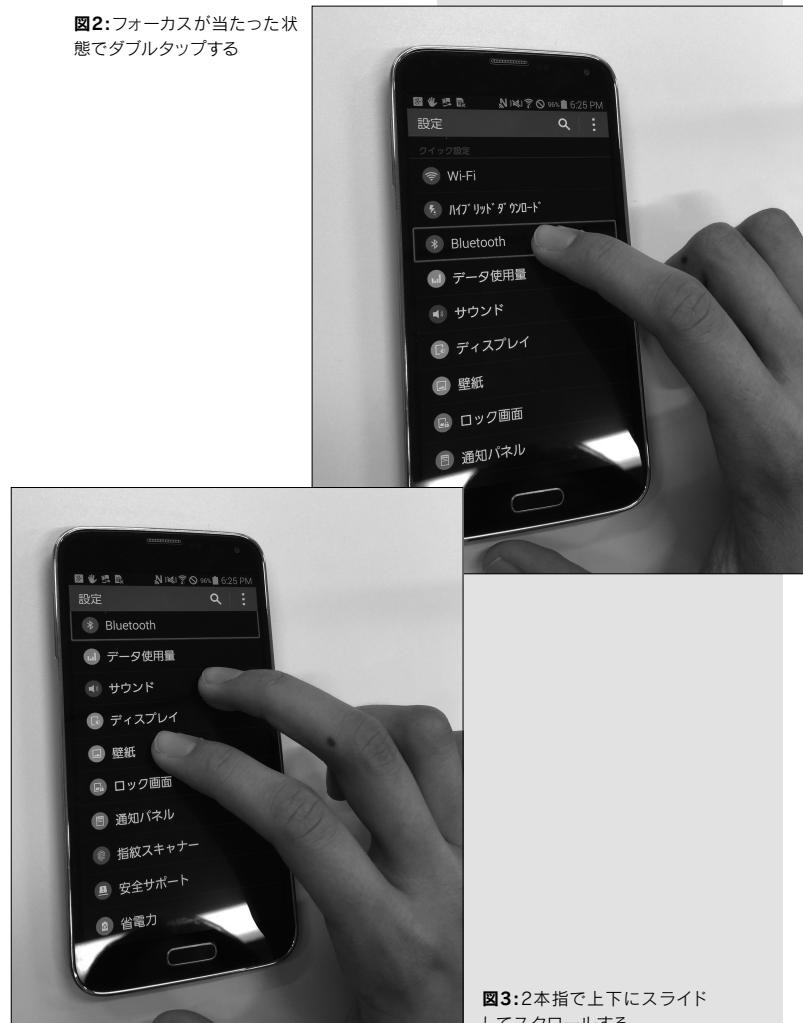


図2:フォーカスが当たった状態でダブルタップする

ボタンやテキストをタップすると、フォーカスが移動し、読み上げてくれます。左右フリックでフォーカスが移動します。

クリックイベント(ボタンのタップや、ListViewの行を選択など)は、フォーカスが当



## 20-8 読み上げ用文字列を設定する

ViewやViewGroupにフォーカスがあたった時、TalkBackは「android:text」または「android:contentDescription」の内容を読み上げます。TextViewやButtonであれば、android:textが設定されているので読み上げが行われますが、ImageViewなど、android:textを持たないViewやViewGroupは、android:contentDescriptionが設定されていないと、読み上げが行われません。

これでは目の不自由な方が画像をタップした時、どのような画像をタップしたか分からないので、android:contentDescriptionに読み上げ用の文字列を設定しましょう。読み上げ用の文字列もローカライズのことを考え、@string/xxxx形式で指定するようにします。次はImageViewで表示されるロゴに説明を付ける例です。

android:contentDescriptionを指定する例

```
<ImageView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:src="@drawable/app_logo"  
    android:contentDescription="@string/msg_app_logo"/>
```



## 20-9 左右フリックに対応する

TalkBackがONの時は、左右フリックでフォーカスが移動します。この時の順序はViewのツリー構造に依存します。@+id/button1にフォーカスがあたっている状態で右フリック(順送り)を行うと、@+id/button2にフォーカスが移動します。

## レイアウトXML

```
<Button  
    android:id="@+id/button1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/button1"  
/>  
<Button  
    android:id="@+id/button2"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_below="@+id/button1"  
    android:text="@string/button2"  
/>
```



## 20-10 フォーカスの制御を行う

図4のようにレイアウトを重ねて表示するケースもあるでしょう。この時、後ろのViewにフォーカスがあたってしまうとユーザーは混乱してしまいます。これを防止するため、Viewにフォーカスを当てない設定を行うことができます。



図4:Viewを重ねて表示したレイアウトの例

レイアウトXMLでフォーカスを当てないようにするには、`android:importantForAccessibility="no"`をフォーカスを当てたくないViewに指定します。次は`@+id/text_base`にフォーカスを当てないようにする例です。

#### レイアウトXMLでフォーカスを当てないよう設定

```
<RelativeLayout  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:background="#FFCCCCCC">  
    <TextView  
        android:id="@+id/text_base"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@string/base_text"  
        android:importantForAccessibility="no"/>  
    </RelativeLayout>  
    <RelativeLayout  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        android:background="#88888888"  
        android:layout_margin="32dp">  
        <TextView  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:text="@string/overlay_text"/>  
    </RelativeLayout>
```

Javaプログラム中で動的に指定する時は、次のように「`setImportantForAccessibility`」メソッドを使用します。

#### Javaプログラムでフォーカスを当てないよう設定

```
// API Level 16以上で使えます  
View view = findViewById(R.id.text_base);  
view.setImportantForAccessibility(View.IMPORTANT_FOR_ACCESSIBILITY_NO);
```



## 20-11 方向キーに対応する

日本ではあまり見かけなくなりましたが、方向キーのある端末もあります。ユーザーは画面を見ることができなくても、Viewの並びを想像できるため、方向キーによる操作と想像しているViewの並びを合わせておく必要があります。これに対応するため、フォーカスが当たっている状態で上下左右が押された時、どのViewにフォーカスが移動すべきか指定できます。

表2は制御用の属性一覧です。たとえば、上キーが押された時`@+id/button`

on1にフォーカスを移動させた場合、`android:nextFocusUp`に`@+id/button1`を指定します。

属性名	意味
<code>android:nextFocusDown</code>	下キーが押された時にフォーカスの当たるViewを指定
<code>android:nextFocusLeft</code>	左キーが押された時にフォーカスの当たるViewを指定
<code>android:nextFocusRight</code>	右キーが押された時にフォーカスの当たるViewを指定
<code>android:nextFocusUp</code>	上キーが押された時にフォーカスの当たるViewを指定

表2: フォーカスの制御用属性

例として、図5のようなレイアウトに方向キー対応を入れてみましょう。

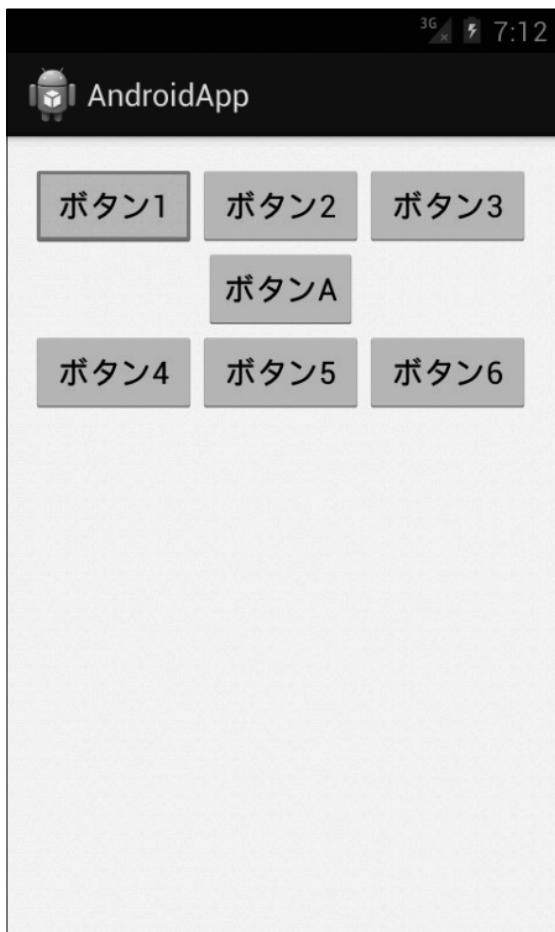


図5: 方向キー対応を加えるレイアウト

ボタン1にフォーカスが当たっている時に下キーを押すと、ボタン4ではなくボタンAにフォーカスが移動してしまいます。次は、下キーが押されたときにボタン4にフォーカスが移動するよう設定した例です。

```
<RelativeLayout 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:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <LinearLayout
        android:id="@+id/layout_1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <Button
            android:id="@+id/button1"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:nextFocusDown="@+id/button4"
            android:text="@string/button1"/>
        <Button
            android:id="@+id/button2"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@string/button2"/>
        <Button
            android:id="@+id/button3"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:nextFocusDown="@+id/button6"
            android:text="@string/button3"/>
    </LinearLayout>
    <Button
        android:id="@+id/button_a"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/layout_1"
        android:layout_centerHorizontal="true"
        android:text="@string/buttonA"
        />
    <LinearLayout
        android:id="@+id/layout_2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/button_a"
        android:orientation="horizontal">
        <Button
            android:id="@+id/button4"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:nextFocusDown="@+id/button5"
            android:text="@string/button4"/>
    </LinearLayout>
</RelativeLayout>
```

```
    android:layout_weight="1"
    android:nextFocusUp="@+id/button1"
    android:text="@string/button4"/>
<Button
    android:id="@+id/button5"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="@string/button5"/>
<Button
    android:id="@+id/button6"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:nextFocusUp="@+id/button3"
    android:text="@string/button6"/>
</LinearLayout>
</RelativeLayout>
```

adbコマンドを使用して、設定が正しく反映されているかを方向キーの無い端末で確認することができます。

adbコマンドで方向キーの操作をシミュレートする

```
$ adb shell input keyevent KEYCODE_DPAD_DOWN
```

KEYCODE\_DPAD\_xxxxの部分に指定可能な方向を表3に示します。

KEYCODE	向き
KEYCODE_DPAD_DOWN	下
KEYCODE_DPAD_UP	上
KEYCODE_DPAD_LEFT	左
KEYCODE_DPAD_RIGHT	右

表3:KEYCODE部に指定可能な方向



## 20-12 動画に字幕を付ける

開発したAndroidアプリ内で、説明のためにVideoViewを用いて動画再生を行うこともあるでしょう。しかし、耳の不自由な方は台詞を聞くことができないため、動画に字幕をつける必要があります。Androidでは、VideoViewに再生中の動画に字幕を付けるAPIがAndroid 4.4(API Level 19)で追加されました。以下の手順で、VideoViewで再生する動画に字幕を付けることができます。

## 1.WebVTT形式の字幕ファイルを作成する

## 2.VideoViewに字幕ファイルをセットする

### WebVTT形式の字幕ファイルを作成する

WebVTTとは、「The Web Video Text Tracks Format」の略で、HTML5のvideoタグで表示される動画に字幕を付ける時などに使用されます。

WebVTTの詳細なドラフトは<http://dev.w3.org/html5/webvtt/>で読むことができます

WebVTTファイルの次に示します。

#### WebVTTフォーマット

```
WEBVTT
Kind:captions
Language:ja

1
00:00:06.500 -> 00:00:08.200
ようこそ

2
00:00:09.000 -> 00:00:11.300
Tech Instituteへ
```

この形式のファイルを、「res/raw」フォルダーにいれます。ここではcaption.vttとして保存したとして話を進めます。

### VideoViewに字幕ファイルをセットする

次に、作成したvttファイルをVideoViewにセットします。VideoViewのインスタンスを取得し、「addSubtitleSource」メソッドで字幕データをセットします。次に例を示します。

#### VideoViewに字幕データをセットする

```
mVideoView.addSubtitleSource(
    getResources().openRawResource(R.raw.caption),
    MediaFormat.createSubtitleFormat(
        "text/vtt", locale.JAPANESE.getLanguage()));
```



## 20-13 テストする

第20章

ローカライズとアクセシビリティ

作成したアプリがアクセシビリティの機能を正しくサポートしているかテストしましょう。

アクセシビリティテストのゴールは以下の3点です。

- ・作成したアプリを、視覚情報なしで利用できること
- ・アプリ内の作業を方向操作のみで容易に行うことができること
- ・その際のフィードバック(読み上げ)は正しいこと

次の6点を特に検証しましょう。

- ・画面のタップを用いず、方向キーのみで操作できる
- ・TalkBackがONの時、フォーカスが当たったものが正しく読み上げられる
- ・タッチガイドがONの時、タップしたものが正しく読み上げられる
- ・ユーザーが操作可能なViewは、Android Designで推奨されている1辺  
48dp以上である
- ・TalkBackがONの時、画像の拡大縮小やスクロールなどのジェスチャー  
が正しく動作する
- ・音のみのフィードバックを行っていないか？メール着信を音のみで通知し  
ていた場合、耳の不自由な方は通知に気づくことができません



## 演習問題

レイアウトXMLにImageViewを1つ配置し、`android:contentDescription`を設定しましょう。設定後、端末でTalkBackをOnにしてアプリを起動し、配置した ImageViewをタップして、設定した読み上げ用文字列で読み上げが行われるか確認しましょう。



## まとめ

本章では、開発したアプリをより多くのユーザーに使用してもらうため、ローカライズの手法とアクセシビリティ機能に対応する手法を紹介しました。日本語を読むことができないユーザーへや、画面を見ることができないユーザー向けの改善を行うことで、より多くのユーザーにすばらしいユーザー体験を提供しましょう。

