Android 6.0 Marshmallow アプリ開発入門 Vol.1

IT研究所

INDEX

第 1 章　Android の概要

第 2 章　Android の環境設定

第 3 章　Android 開発の基本

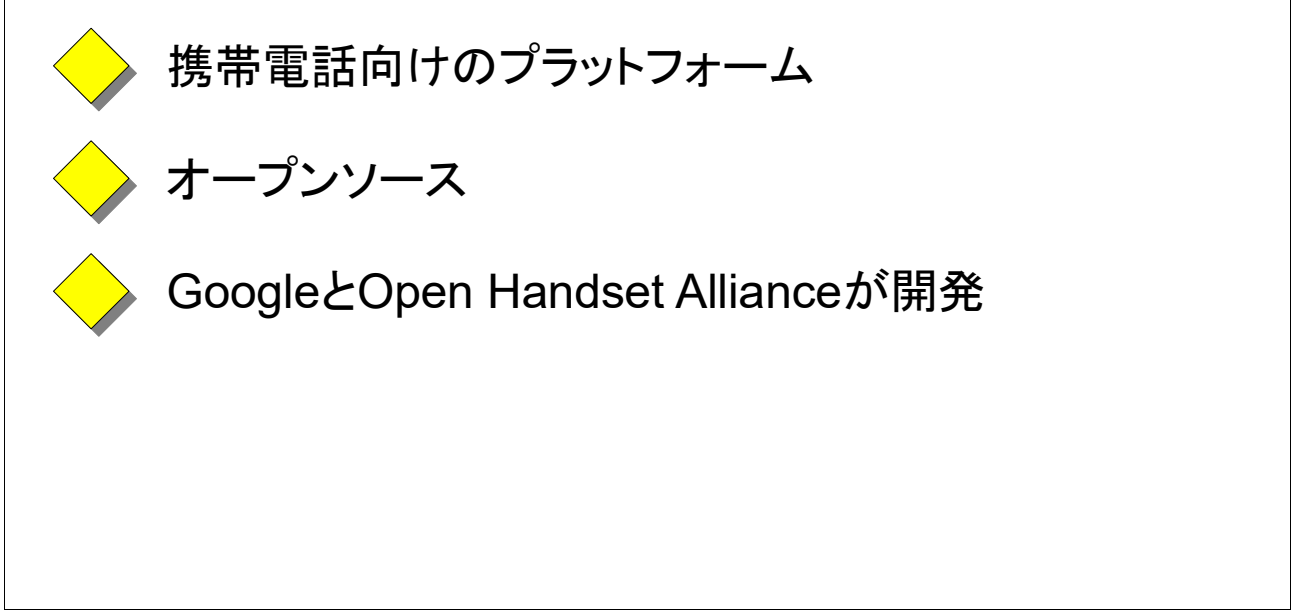
第 4 章　プロジェクトの基本構成

第 5 章　GUI の基本

第 6 章　GUI の利用

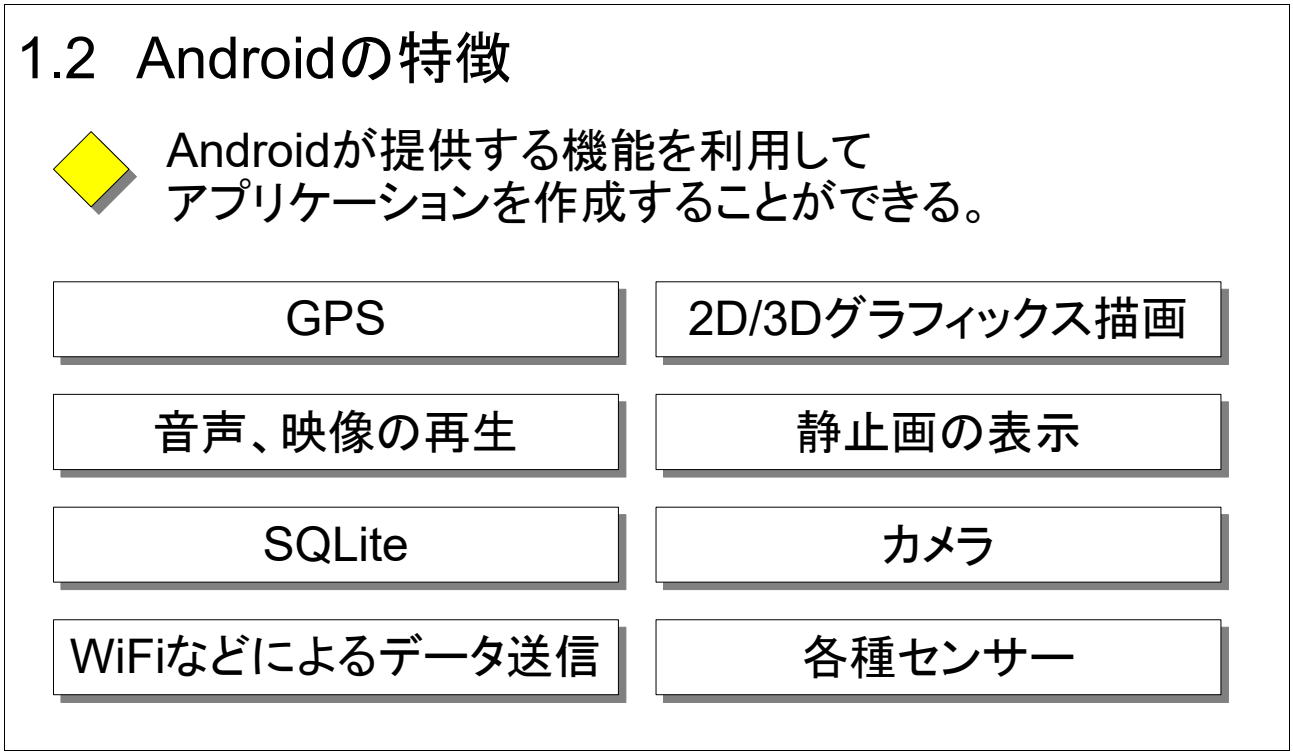
第１章
Android の概要

1.1 Androidとは



Android は NTT ドコモ、HTC、Google などが参画する OHA（Open Handset Alliance）によって提供される携帯端末向けのプラットフォームです。これは、スマートフォンやタブレットなど幅広いデバイスに搭載されており爆発的な勢いで普及しています。その理由は、Android がオープンソースであり、またプログラムを実行するための仕組みとして仮想マシンを持っている点が大きい。

Android は Java の重要な思想のひとつである「Write Once, Run Anywhere」を継承しており、Android 上で作成したアプリケーションは、Android が搭載されている機器であれば、どこでも動作することができます。



Android には下記のような特徴があり、開発者は Android が提供する機能を利用してアプリケーションを作成することができます。

◆Android の特徴

・コンポーネントの再利用と再配置を可能にするアプリケーション・フレームワーク
・モバイル機器のために最適化された Dalvik 仮想マシン
・WebKit エンジンをベースとした統合ブラウザ
・カスタム 2D グラフィックスライブラリによって最適化されたグラフィックス
OpenGL ES1.0 仕様に基づく 3D グラフィックス（ハードウェアアクセラレーションは任意）
・構造化データストレージとして SQLite がある
・メディアは一般的なオーディオ、ビデオ、および静止画像フォーマットをサポート（MPEG4 、H.264 、MP3 、AAC、AMR、JPG、PNG、GIF）
・GSM 電話（ハードウェア依存）
・Bluetooth、EDGE、3G、と WiFi に対応(ハードウェア依存)
・カメラ、GPS、コンパス、そして加速度計に対応(ハードウェア依存)
・デバイスエミュレータ、デバッグツール、メモリおよびパフォーマンスプロファイリング
および Eclipse IDE のためのプラグインを含む豊かな開発環境

1.3 Androidのシステム構成

Linuxをベースとしたシステム上にDalvik仮想マシンを搭載し、その上でアプリケーションが動作する。

Android という OS の基本的な構成は、まず土台となる部分に Linux カーネルがあり、その上に各種のネイティブなライブラリが用意されており、ここに Dalvik（ダルヴィック）と呼ばれる仮想マシンが配置されています。
仮想マシン上には、コアなライブラリ（Java ライブラリ）があり、その上にアプリケーションフレームワークが構築されています。

Android のアプリケーションは、仮想マシンとアプリケーションフレームワークの上で実行されます。

◆Android のシステム構成
① Linux
・Android は Linux ベースの OS である。

②ネイティブライブラリ
・ハードウェアのアクセスなどを必要とするネイティブライブラリが用意されている。

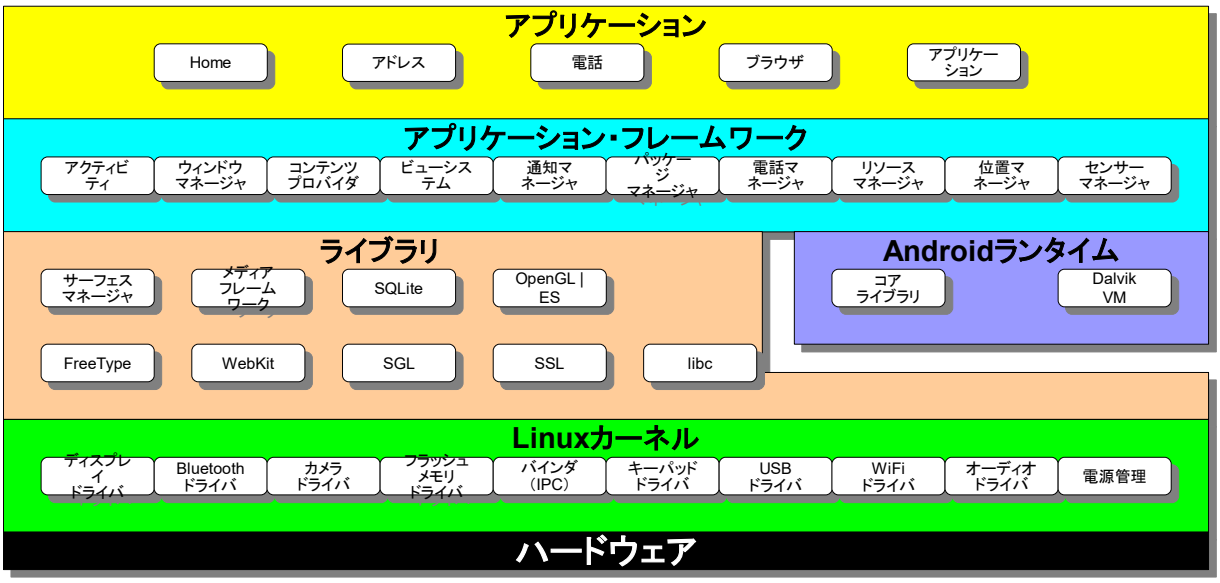
③ Dalvik
・仮想マシン（アプリを動作させる環境）

④ Java ライブラリ
・コアライブラリであり、ネイティブライブラリ機能を呼び出すためのクラスが用意されている。

⑤アプリケーション
・Android アプリケーション。

1.4 Androidの構成要素

◆Androidは5つの要素で構成されている。



◆Android の構成要素

① Linux カーネル

・Linux のバージョン 2.6 をベースに Android 用の変更を加えた Linux カーネル。

・アプリケーションを動作させるために必要な機能を提供しており、ハードウェアドライバを含んでいる。

②ライブラリ

・Android には C/C++で作成されたライブラリ（Web ブラウザ描画、データベース等）が含まれている。

・アプリケーションがアプリケーションフレームワークを通して使用する。

③ Android ランタイム

・Android アプリケーションの実行環境

・Android で使用する Java のコアライブラリ、実行環境である仮想マシン（Dalvik VM）が含まれる。

④アプリケーション・フレームワーク

・アプリケーションの動作やライブラリを使用するために必要な Java API を提供する。

⑤アプリケーション

・アプリケーション・フレームワークを使用して開発された Android アプリケーション。
（電話、Web ブラウザ、アドレス帳など）

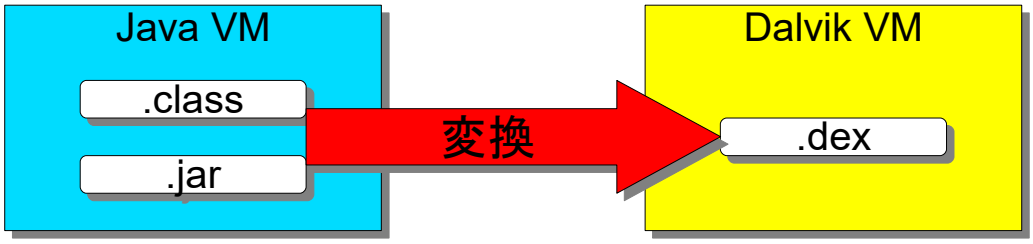
1.5 Androidの実行環境

Dalvik仮想マシン（Dalvik VM）

・Googleが独自開発した仮想マシン
・メモリの少ない環境に最適化されたJava VM

Dalvik実行形式フォーマット（.dex）

・Dalvik VMでは.class、.jarを.dexに変換して実行
・.classと比較して、稼動時の使用メモリサイズが小さい



Dalvik 仮想マシン（Dalvik VM）は、Google 社のエンジニアが Android プラットフォームのために設計・開発した、レジスタベースの仮想マシンです。

Dalvik は低メモリ環境に対して最適化されており、オペレーティングシステムによるプロセス間の分離、メモリ管理、スレッドのサポートを用いて複数の VM インスタンスが同時に動作できるよう設計されています。

Dalvik VM と Java VM には互換性がなく、実行ファイルのバイトコードが異なるため、Dalvik Vm 上では.class 形式や.jar 形式のファイルは実行できません。また、その逆で Java VM 上では.dex 形式のファイルは実行できません。

Android SDK に含まれる dx と呼ばれるツールが正規の Java コンパイラでコンパイルされた Java クラスファイルを別のファイル形式（.dex 形式）に変換します。

【参考】 Androidアプリケーションとiアプリの比較

	Androidアプリケーション	iアプリ
Javaのサポート	Java SE	Java ME
ｱﾌﾟﾘｹｰｼｮﾝｻｲｽﾞ制限	制限なし	1MB（機種により異なる）
ヒープメモリ制限	16MB（SDKの設定値）	6MB～15MB （機種により異なる）
通信制限	制限なし	アプリのダウンロード先以外 通信できない
HTTP通信サイズ制限	制限なし	リクエストボディ：80KB レスポンスボディ：150KB
クッキー	利用可能	利用可能
文字コード	UTF-8	SJIS
永続化方法	ファイル、SQLite	ScratchPad

Android アプリケーションは、i アプリなど従来の携帯アプリケーションと比較して制限が少ないため、パソコン上で動作する Java アプリケーションに近い。

メモ

第 2 章
Android の環境設定

2.1 Androidアプリケーション開発に必要な環境

項目	必要な環境
OS	Windows XP、Windows Vista、Windows 7 Windows 8、Windows 10 Mac OS X 10.5.8以降（x86のみ） Linux
JDK	JDK 7以上
Android開発ツール	Android SDK
統合開発環境	Android Studio

※2015年10月01日時点での情報である。

このテキストではAndroid開発ツールは「Android 6.0」を使用して記述する。

Android アプリケーションを開発するためには Android アプリケーションの開発環境を構築する必要があります。

Android アプリケーションは Java 言語で記述するため、まず必要になるのは Java の開発環境である JDK（Java Development Kit）です。また、Android アプリの開発を効率的に行うため Android Studio という統合開発環境を用います。

JDK のインストールが完了したら、Android Studio のインストールを行います。Android Studio のインストールを行うと、Android SDK のインストールも行うことができます。

Android SDK は Android アプリケーションを開発するために必要なライブラリや、Android アプリケーションを PC 上で動作させるための Android 端末エミュレータなどの開発ツールがまとまった開発キットです。

Android Studio の主な機能としては、以下が挙げられます。

●Android アプリケーション開発プロジェクトが作成可能

●Android アプリケーションのパッケージング、実行、デバッグが可能

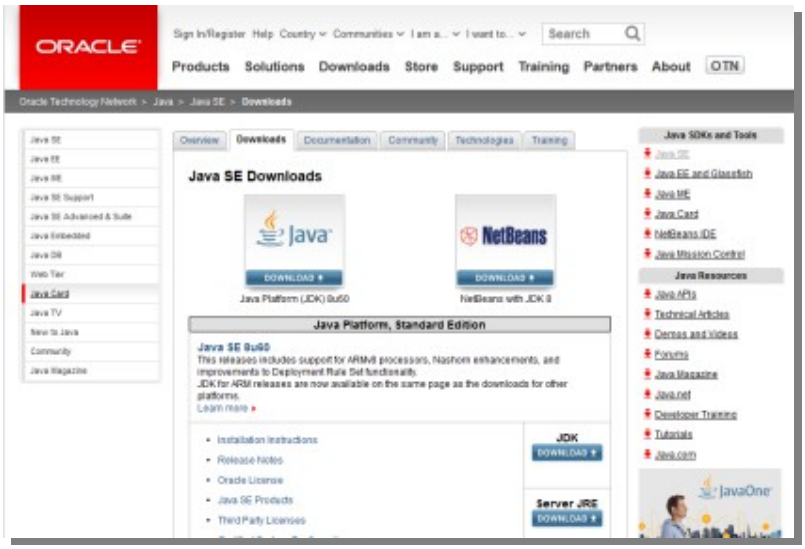
●レイアウトエディタ

●各種 XML ファイル修正時のコード補完など

これらの機能により、Android アプリ開発の効率化を図ることができます。

2.2 JDKのインストール

http://www.oracle.com/technetwork/java/javase/downloads/index.html



◆JDK のインストール

JDK は、下記 URL より取得することができます。

今回は、執筆時点で JDK8 の最新版である「Java SE Development Kit 8u60」をダウンロードします。ダウンロード後は、インストーラに従ってインストールを行ってください。

※Java7 以上であれば Android Studio を利用することができます。

●参照

http://www.oracle.com/technetwork/java/javase/downloads/index.html

2.3 Android Studioのインストール

http://developer.android.com/intl/ja/sdk/index.html

◆Android Studio のインストール

Android Studio は Google が提供する Android プラットフォームに対応する統合開発環境（IDE）であり、Java コードエディタやデバッガなどが統合された環境で開発を行うことができます。

Android Studio は下記 URL からダウンロードすることができます。ダウンロード後は、インストールを行ってください。

Eclipse では、Android SDK を別でインストールする必要がありましたが、Android Studio の場合は、Android SDK も一緒にセットアップすることができます。

●参照

http://developer.android.com/intl/ja/sdk/index.html

◆参考

Android Studio のメニューなどを日本語で表示したい場合は、日本語化リソースファイルをセットアップすることで日本語表記にすることができます。

※このテキストでは、日本語表記で使用できるものとして記述します。

【参考】 AVD（Android Virtual Device）の作成

Android端末をエミュレートするためのソフトウェア。

◆AVD（Android Virtual Device）の作成

Android で作成したアプリは実際の端末を利用して動作確認を行うことができますが、AVD を利用してアプリの動作確認を行うことができます。

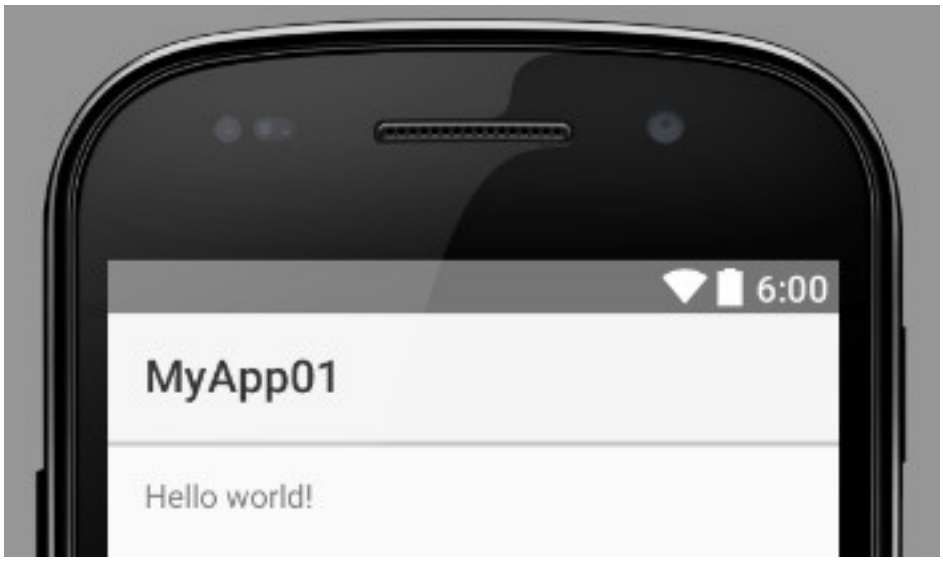
AVD とは Android 端末をエミュレートするためのソフトウェアで、実際の端末がなくても、それに近い環境でアプリの動作を確認することができます。


メモ

第 3 章
Android 開発の基本

3.1 Android開発の基本

Androidの開発手順を理解する。

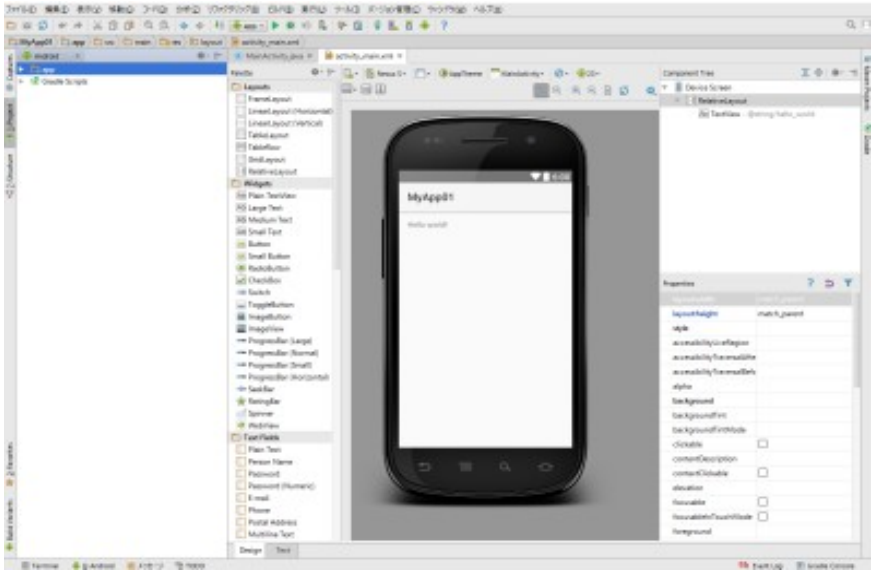


Android の開発を行うためには、手順を理解しなければなりません。

Android Studio により Android プロジェクトという形で作成していきますが、Android プロジェクトには、多数のファイルが用意されています。これらのファイルの内容と役割を理解し、Android アプリケーション作成の初歩として、プロジェクトの作成から Android アプリケーションの実行までの手順を理解します。

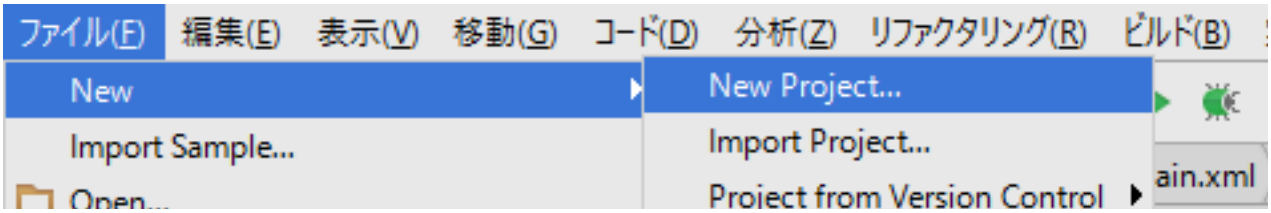
3.2 Androidプロジェクトの作成

Androidのプロジェクト作成及び設定を行う。



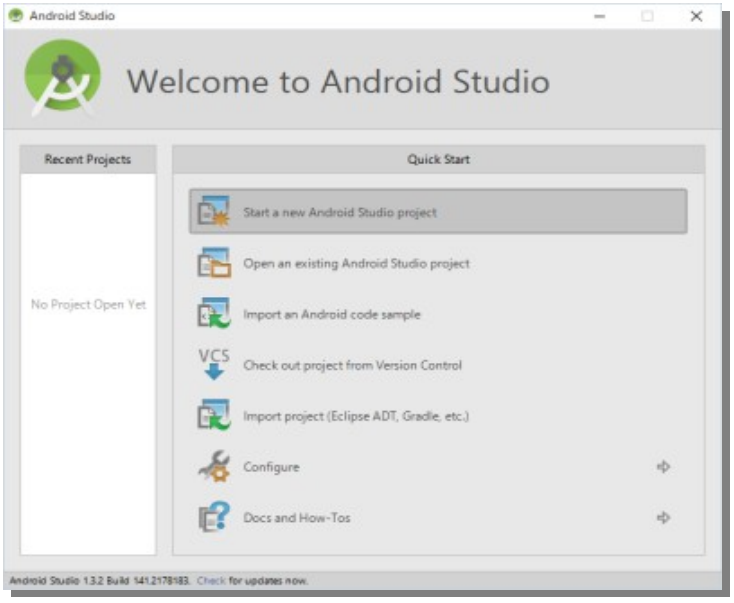
◆新規 Android プロジェクトの作成方法

①[ファイル] → [New] → [New Project]の順で選択する。



※インストール直後は下記画面が表示されますので

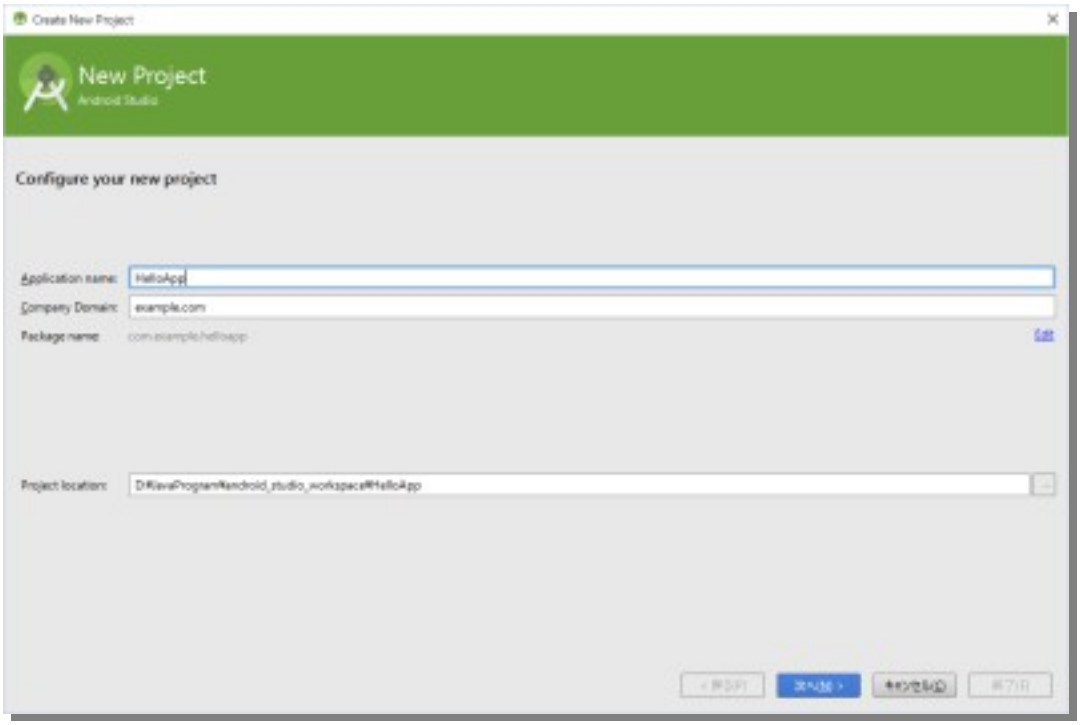
「Start a new Android Studio project」から新規 Android プロジェクトを作成することができます。



②「New Project」画面が表示される。

●Application name に「HelloApp」と入力し「次へ」ボタンを押す。

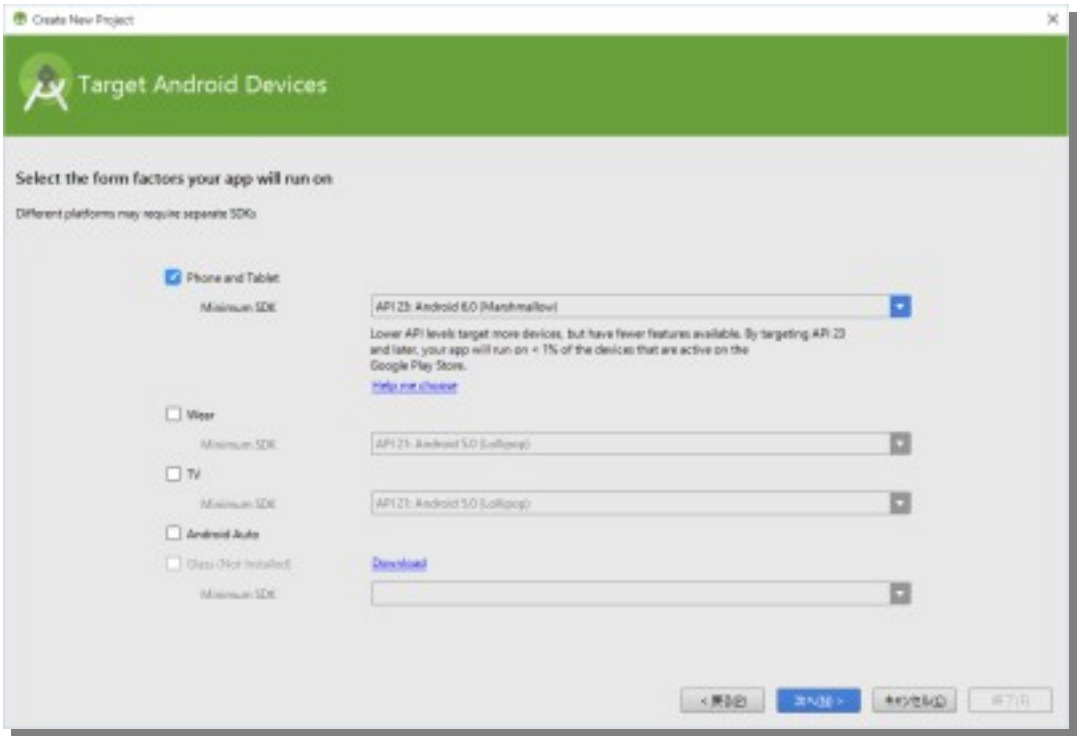
※Application name を入力すると自動的に Project location も連動します。



③「Target Android Devices」画面が表示される。

●Phone and Tablet Minimum SDK を選択し「次へ」ボタンを押す。

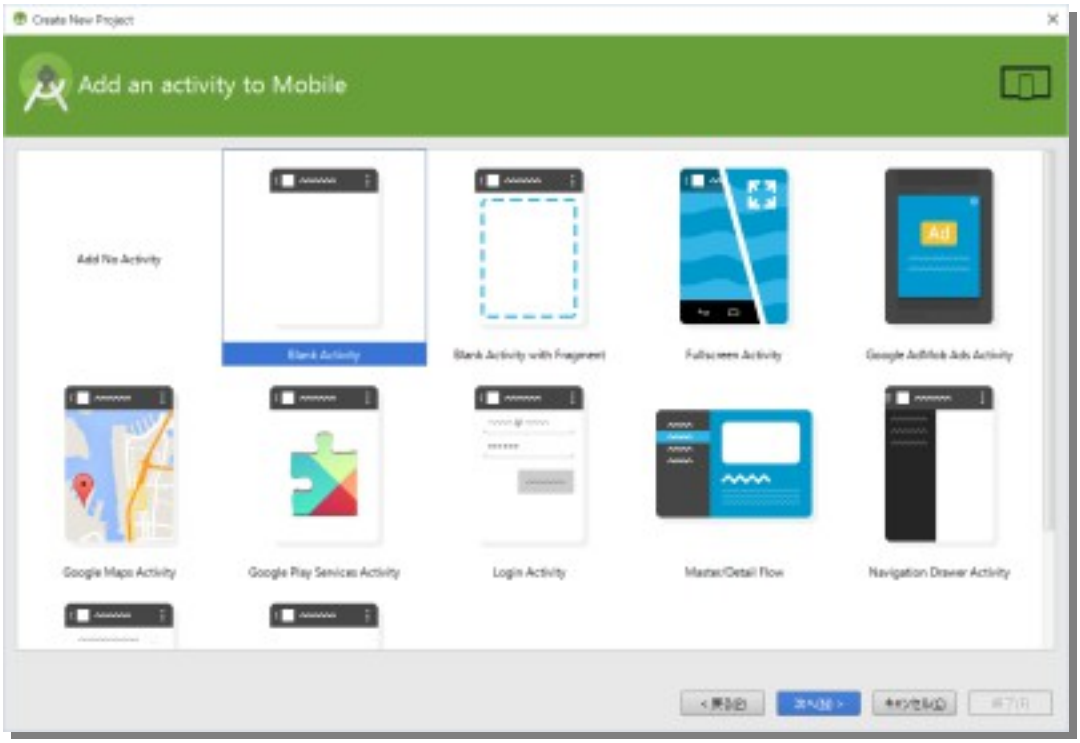
※本テキストでは「API23： Android6.0(Marshmallow)」を選択して記述します。



④「Add an activity to Mobile」画面が表示される。

●Activity を選択し「次へ」ボタンを押す。

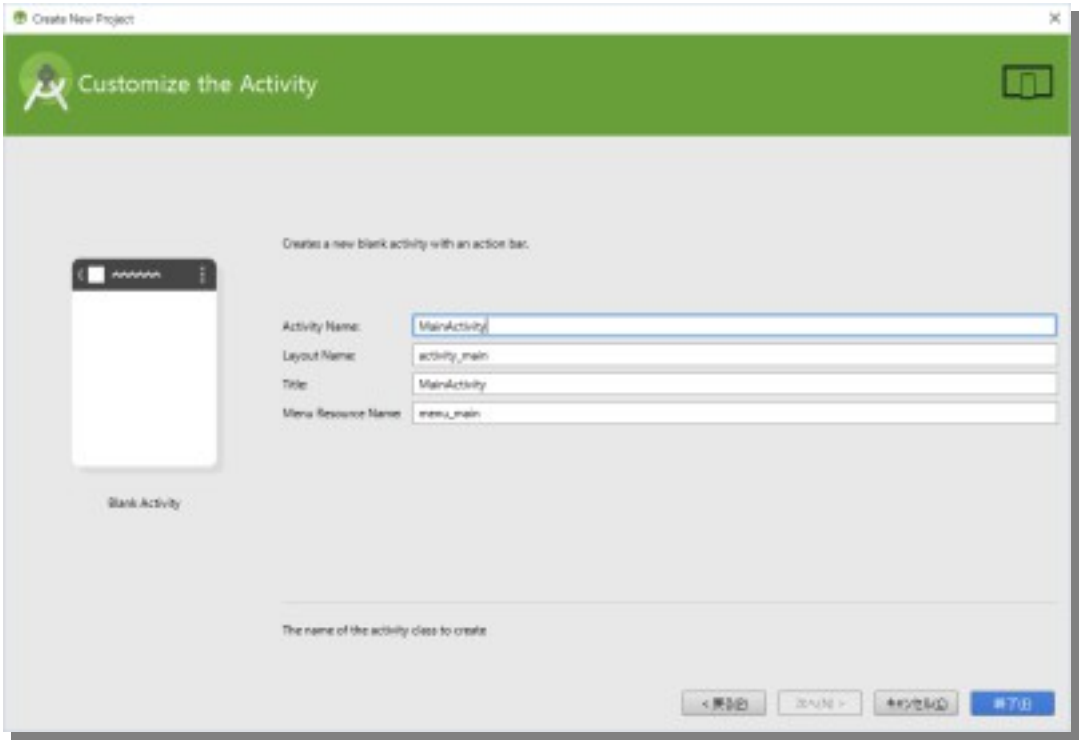
※本テキストでは「Blank Activity」を選択して記述します。



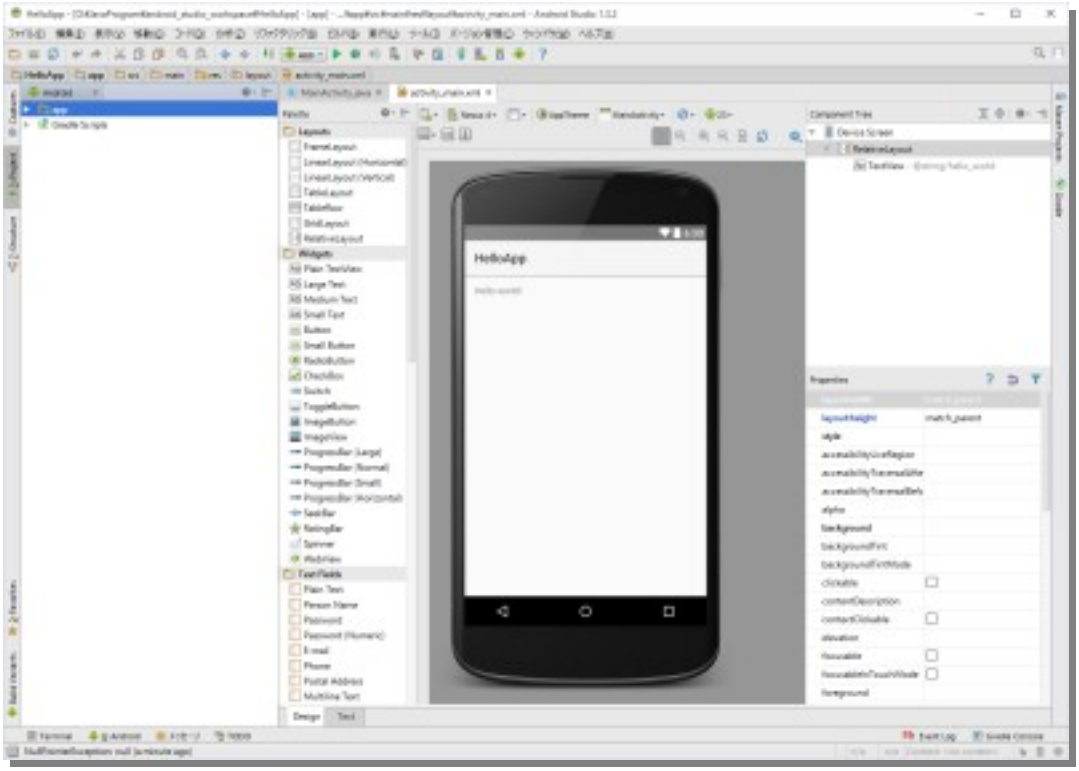
⑤「Customize the Activity」画面が表示される。

●「Activity Name」、「Layout Name」、「Title」、「Menu Resource Name」を入力し「終了」ボタンを押す。

※本テキストではデフォルト状態（変更なし）として記述します。

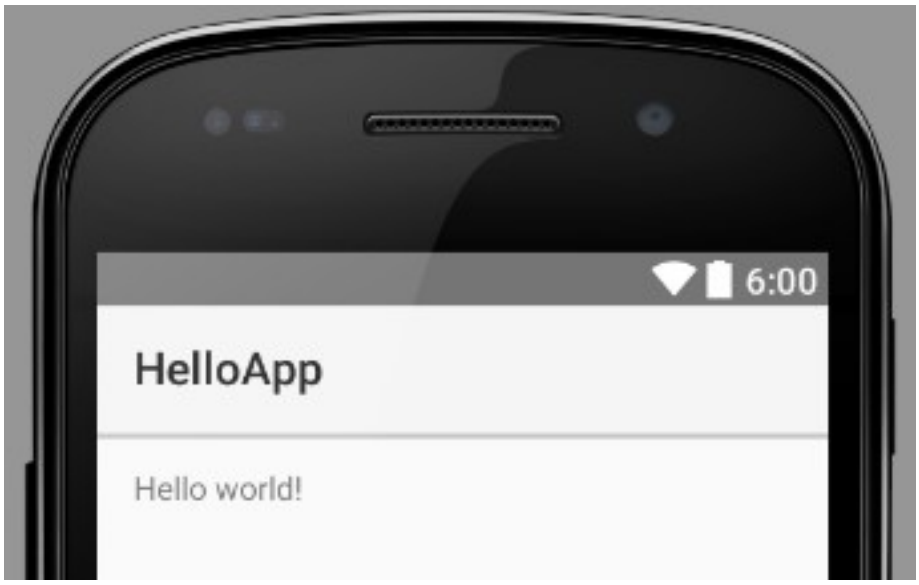


⑥プロジェクトが作成される。



3.3 Androidアプリケーションの実行

エミュレータでAndroidアプリを実行する。

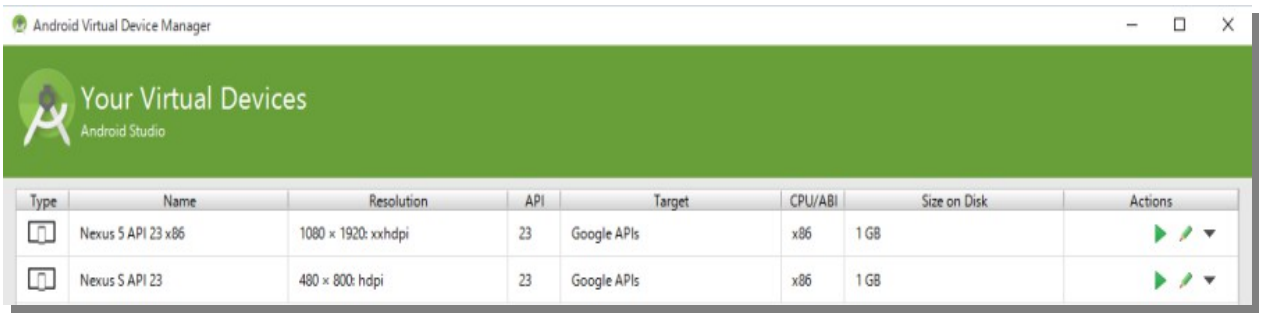


①エミュレータを起動する。

●ツールバーの「AVD Manager」をクリックする。

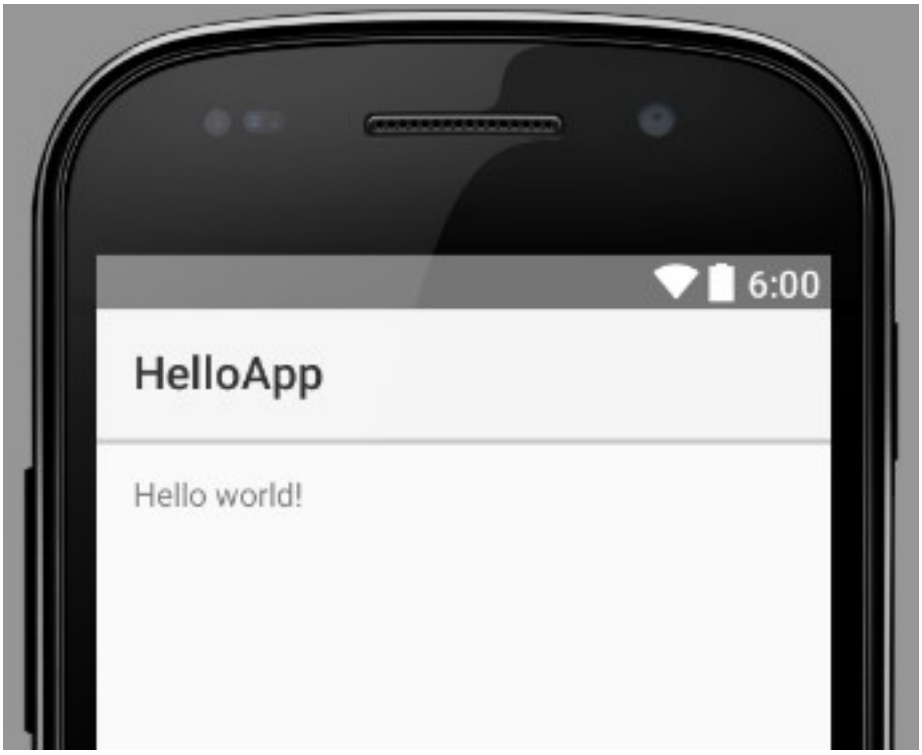
②「Your Virtual Devices」画面から、AVD の選択を行い実行を開始する。

※「Your Virtual Devices」画面で、AVD の設定ができていない場合は、新規設定を行ってください。
本テキストでは「Nexus S API 23」を使用して記述します。



③エミュレータが起動される。
※初回起動時は、数分時間がかかります。

④ Android アプリケーションを実行する。
●「実行」ボタンを押すと Android アプリケーションが実行される。
※メニューの実行からも実行することができます。



3.4 Androidプロジェクトの内容

Androidプロジェクトの内容を理解する。

「manifests」フォルダ

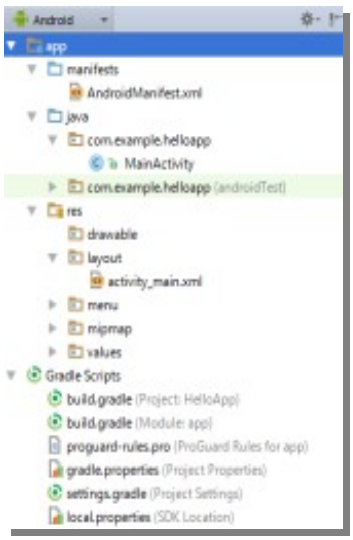
AndroidManifest.xml

「java」フォルダ

「res」フォルダ

Gradle Scripts

gradle.properties



作成された「HelloApp」プロジェクトには、初期状態で多数のフォルダやファイルが作成されています。それらの主な役割について記述します。

◆プロジェクトの内容

●「java」フォルダ

Java のソースコードファイルが作成されるフォルダである。ここでは、プロジェクトで設定した com.example.helloapp パッケージが作成され、その中に「MainActivity.java」というファイルが作成される。

●「res」フォルダ

プロジェクトで利用するイメージファイルやテキストデータなどを配置する。デフォルトでは「drawable-・・・」というフォルダにアイコンのイメージファイル、「layout」にレイアウト情報の XML ファイル、「values」にテキストの値を記述した XML ファイルが、それぞれ用意される。

●AndroidManifest.xml

Android プログラムに関する情報が記述されたマニフェストファイル。XML で記述されている。

●gradle.properties

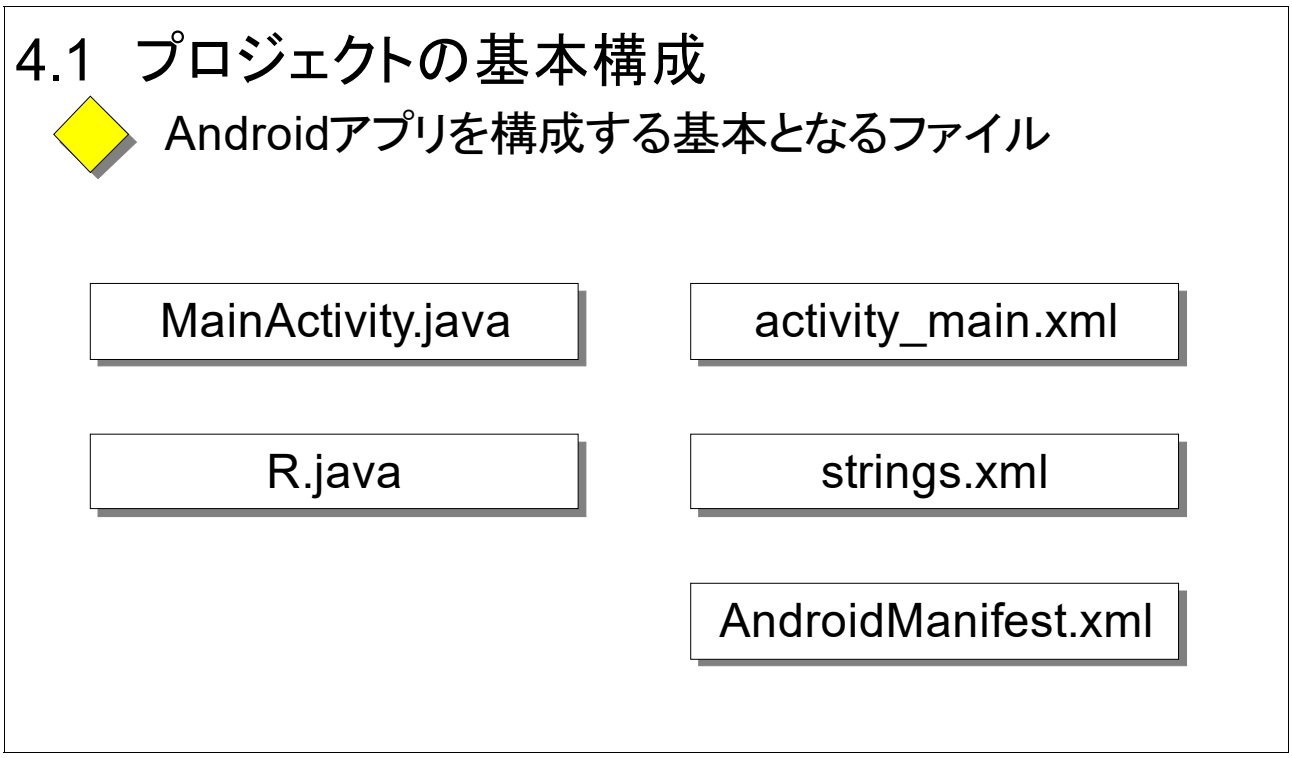
プロジェクトの初期属性が記述されたプロパティファイル。デフォルトでは使用するターゲットに関する設定が記述されている。

メモ

第 4 章
プロジェクトの基本構成

4.1 プロジェクトの基本構成

Androidアプリを構成する基本となるファイル



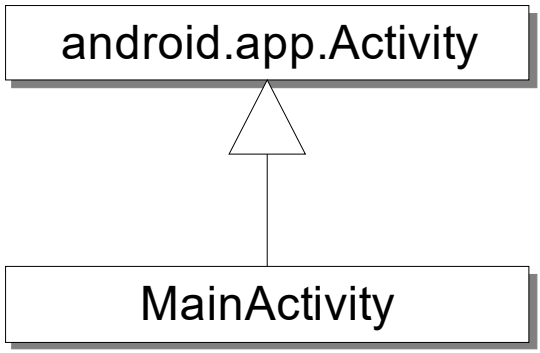
Android アプリケーションを構成する上で、最も基本となるファイルとその役割を記述します。Android アプリケーションは、「Java クラス」「自動生成されたクラス」「レイアウトファイル」「データを保存したファイル」「マニフェストファイル」から構成されています。

◆主要ファイルとその役割

ファイル名	役割
MainActivity.java	「java」フォルダ内に作成される。これがAndroidアプリのメインプログラムとなる。 アプリを起動した時に、最初に画面に表示される内容がプログラムされている。
R.java	リソースIDを管理するJavaファイル。 このファイルはユーザが編集することはない。
activity_main.xml	「res」フォルダ内の「layout」フォルダ内に作成される。 このファイルはMainActivity.javaに記述されているコード内から利用されるもので、実際に画面に表示されるレイアウト情報が記述されている。
strings.xml	「res」フォルダ内の「values」フォルダ内に作成される。 アプリで使用されるテキストリテラルを別ファイルとしてまとめたものである。 このファイルから必要に応じてテキストを取り出し利用する。
AndroidManifest.xml	「マニフェストファイル」と呼ばれ、アプリに関する諸情報が記述されている。

4.2 MainActivity.java

- 最初に実行されるプログラム。
- android.app.Activityクラスを継承して作成される。



MainActivity.java は、Android アプリケーションのメインプログラムであり、アプリケーションを実行した時に、最初に画面に表示される内容がプログラムされています。

MainActivity.java には MainActivity クラスが定義されるが、このクラスには Android SDK で定義されているクラスで、android.app パッケージに定義されている Activity クラスが継承され作成されます。

Activity とは、Android アプリケーションの基本となるコンポートネントのことで、Windows アプリケーションでいえばウインドウオブジェクトに相当するものです。通常は、ユーザーインターフェースとなる画面ごとに、Activity クラスのサブクラスを定義します。複数の画面を持つアプリケーションでは、複数のアクティビティから構成されます。

◆Activity クラスの役割

・Android におけるプログラムの中心部分

・画面表示を管理するクラス

・利用者からの入力や操作などに対する処理を行う

◆Activity クラスを継承したクラスの定義方法（基本形）

```
import android.os.Bundle;
import android.app.Activity;

public class クラス名 extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        //初期化処理を記述する。
    }
}
```

※Android 6.0 では onCreate メソッドの他に、オプションメニューを設定する onCreateOptionsMenu メソッドや onOptionsItemSelected メソッドが自動的に定義されるが、省略して記述します。

◆onCreate メソッド

・Activity クラスに定義されているメソッド

・起動時の処理を行うためのメソッド

◆onCreate メソッドの定義方法

```
@Override
public void onCreate(Bundle savedInstanceState) {
        //初期化処理を記述する。
}
```

引数には、android.os パッケージの Bundle クラスのインスタンスが渡される。

Bundle とは「束、包み」などの意味がある。

Bundle クラスは、OS の判断で強制的に停止、終了する時に一時的にデータを格納するという役割を持っている。

◆onCreate メソッド内のプログラム

```
@Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState); ・・・・・・（1）
        setContentView(R.layout.activity_main); ・・・・（2）
}
```

（1）

スーパークラスにある onCreate メソッドを呼び出し。

プログラムを起動する際に必要な処理などがスーパークラスに用意されている場合もあるので、このように「まずスーパークラスのメソッドを呼び出す」というのは重要である。

（2）

Android の画面に表示される内容を設定する。

activity_main.xml の内容をコンテントビューに設定する。こうすることで、指定した部品が画面に表示される。

●setContentView メソッド

【書式】

```
①public void setContentView (View view)
②public void setContentView ( リソースID)
```

【機能】

画面に表示するコンテンツとなるオブジェクトを設定する 。

●R.layout.activity_main

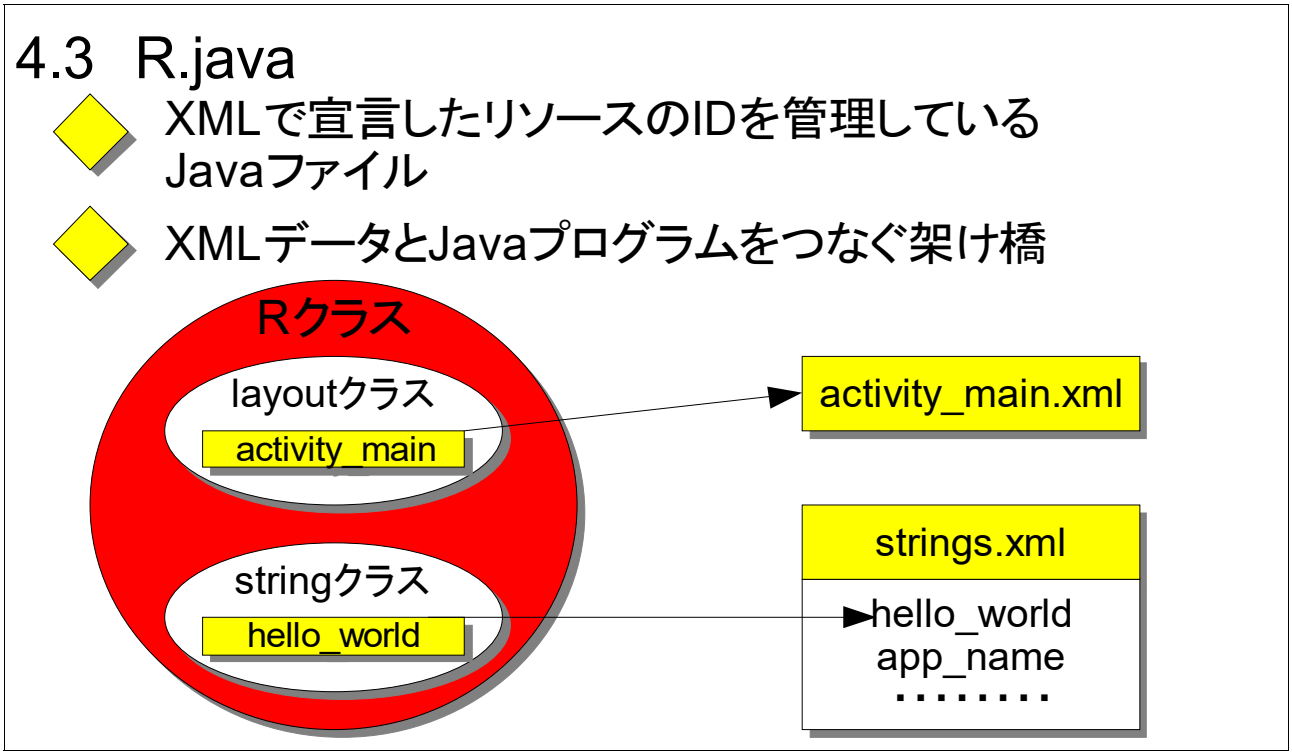
R クラス内の layout クラスに定義されている変数 activity_main を意味する。

```
public final class R {
    public static final class layout {
        public static final int activity_main=0x7f030000;
    }
}
```

※一部抜粋

※activity_main に代入されている 0x7f030000 は指定されたリソースのある場所を示す値でリソース ID といい

XML データと Java のプログラムをつなぐ架け橋のようなものです。



プロジェクトに設置したリソース（資源）をプログラムの中から利用するため、各リソースにはリソースIDが割り当てられます。リソースIDは「R.java」と言う名前のファイルで管理されます。

「R.java」ファイルは初めてプロジェクトをビルドした時に作成され、その後ビルドが行われるたびに自動的に更新されていくため、ユーザはリソースの追加をするだけで、リソースに対するIDの管理などは行う必要がありません。

◆R.java のプログラム

下記プログラムは、新規プロジェクト作成時の初期設定段階でのプログラム内容です。

ファイル名：R.java

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package com.example.helloapp;

public final class R {
    public static final class attr {
    }
    public static final class dimen {
        public static final int activity_horizontal_margin=0x7f050000;
        public static final int activity_vertical_margin=0x7f050001;
    }
    public static final class id {
        public static final int action_settings=0x7f080000;
    }
    public static final class layout {
        public static final int activity_main=0x7f030000;
    }
    public static final class menu {
        public static final int menu_main=0x7f070000;
    }
    public static final class mipmap {
        public static final int ic_launcher=0x7f020000;
    }
    public static final class string {
        public static final int action_settings=0x7f060000;
        public static final int app_name=0x7f060001;
        public static final int hello_world=0x7f060002;
    }
    public static final class style {
        /**  Customize your theme here.
         */
        public static final int AppTheme=0x7f040000;
    }
}
```

R クラスには、リソースの種類ごとに drawable クラスや layout クラスや string クラスが定義されています。各クラスファイルの中では int 型の値が定義されており、これがリソースに対するリソース ID となります。リソース ID の割り当てはリソースの種類によって異なります。また、リソース ID は自動的に割り付けられた値であり、ユーザが勝手に変更することはできません。

●「res」フォルダ内の「layout」フォルダには、新規プロジェクト作成段階では、「activity_main.xml」ファイルが格納されており、layout クラスで変数 activity_main には、レイアウト用 XML ファイルのファイル名に対してリソース ID が割り当てられる。

●「res」フォルダ内の「values」フォルダには、新規プロジェクト作成段階では、「strings.xml」ファイルが格納されており、string クラスの各変数には、「strings.xml」ファイルの中で各要素に付けられた名前に対してリソース ID が割り当てられる。

【R.java】

```
public final class R {
    public static final class string {
        public static final int action_settings=0x7f060000;
        public static final int app_name=0x7f060001;
        public static final int hello_world=0x7f060002;
    }
}
```

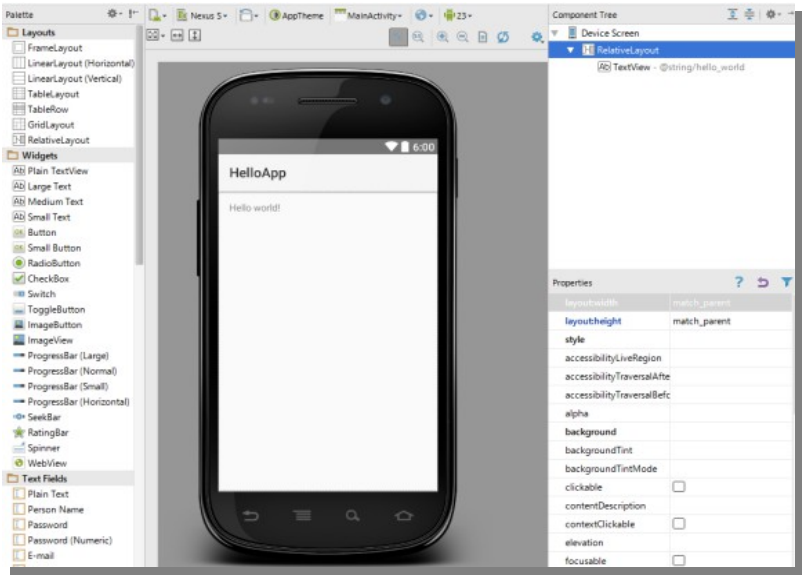
※一部抜粋

【strings.xml】

```
<resources>

    <string name="app_name">HelloApp</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>

</resources>
```

4.4 activity_main.xml

画面のレイアウト用リソースファイル

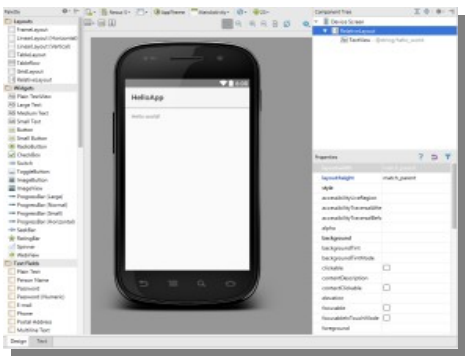


activity_main.xml は、R.layout.activity_main に割り当てられた画面のレイアウト用リソースファイルです。このファイルは、「res」フォルダ内の「layout」フォルダに保存されており、このファイルを編集することで画面表示を構成することができます。

activity_main.xml を編集する方法は、ビジュアルにレイアウトを作成する「Android レイアウト・エディタ」と呼ばれるツールがあり、これを使用して画面表示のリソースファイルを作成することができます。

◆activity_main.xml の編集方法

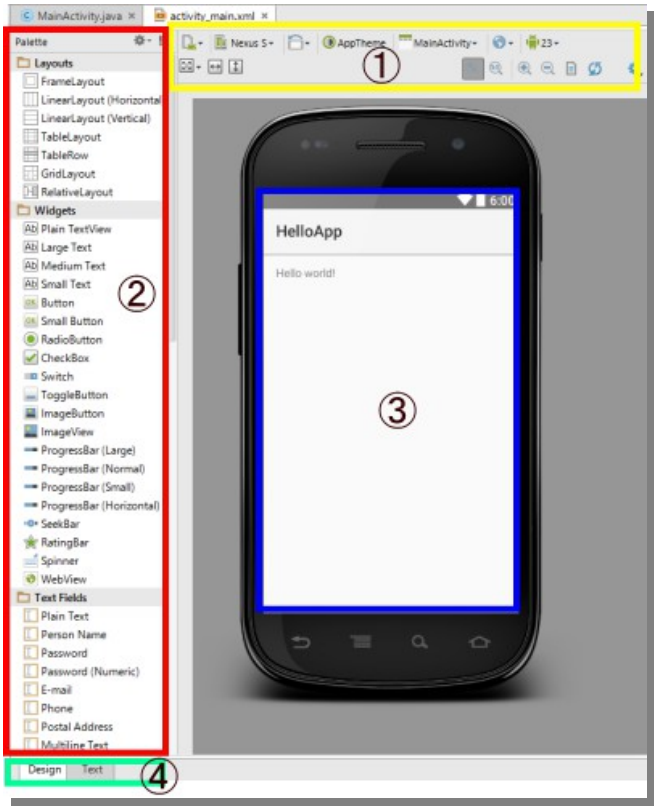
「デザイン・エディタ」でビジュアル的に編集する方法と「テキスト・エディタ」で XML のソースコードを直接編集する方法があります。いずれかの方法で編集することができるが、最終的に得られる XML コードは同じです。



デザイン・エディタ

テキスト・エディタ

◆「Android レイアウト・エディタ」の説明



①ポップアップメニュー

デザインのプレビューで用いられるデバイス環境を設定する。

②リスト表示部

このデザインツールで利用可能な部品をまとめたものである。それぞれの部品は、種類ごとに分類されている。ここから使用したい部品を③のエリアまでドラッグ＆ドロップして配置し、画面のデザインを作成する。

③デザイン部

画面のデザインを行う領域である。②から部品をドラッグ＆ドロップして配置し、表示を整えたりし画面を作成する。配置した部品は、マウス操作で位置や大きさを調整することができる。

④切り替えタブ

「Design」タブをクリックすると「ビジュアル・エディタ」、「Text」タブをクリックすると「テキスト・エディタ」により編集できる。デフォルトでは「Design」が選択されている。

◆「activity_main.xml」のソースコード

下記プログラムは、新規プロジェクト作成時の初期設定段階でのプログラム内容です。

ファイル名：activity_main.xml

```
<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">

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

</RelativeLayout>
```

●タグの基本構成

Android では、まずベースにレイアウトを配置し、その中に使用する部品を組み込んでいきます。

```
〈△△△Layout〉
   〈部品１　・・・　/〉
   〈部品２　・・・　/〉
   〈部品３　・・・　/〉
〈/△△△Layout〉
```

●主なレイアウトの種類

レイアウトの種類	機能
LinearLayout	縦方向または横方向に一直線に部品を並べるレイアウト。
RelativeLayout	相対的な位置指定をするレイアウト。部品同士が関連しあって位置を決める。
TableLayout	表形式に並べるレイアウト。
FrameLayout	ビューの重ね合わせが可能なレイアウト。最もシンプルなレイアウト。

●レイアウトタグの主な属性

属性の種類	機能
xmlns:android	Androidのスキーマの指定を行う。 これは必ず「http://schemas.android.com/apk/res/android」と指定する。
android:layout_width	レイアウトの横幅に関する設定を行う。 ◆wrap_content・・・・・表示されるコンテンツに応じて最適な幅に調整する。 ◆fill_parent・・・・・端から端までいっぱいに広げる。 ◆match_parent・・・・・fill_parentと同様。SDK2.2より、この名称となった。
android:layout_height	レイアウトの縦幅に関する設定を行う。 ◆wrap_content・・・・・表示されるコンテンツに応じて最適な幅に調整する。 ◆fill_parent・・・・・端から端までいっぱいに広げる。 ◆match_parent・・・・・fill_parentと同様。SDK2.2より、この名称となった。
android:orientation	LinearLayout使用時に部品の並べられる方向の設定を行う。 ◆vertical・・・・・垂直方向に部品を配置する。 ◆horizontal・・・・・水平方向に部品を配置する。

●主な部品の種類

部品の種類	機能
TextView	文字列を表示する。
EditText	テキストボックスを設定する。
Button	ボタンを設定する。
CheckBox	チェックボックスを設定する。
RadioButton	ラジオボタンを設定する。

●部品タグの主な属性

属性の種類	機能
android:layout_width	部品の横幅に関する設定を行う。 ◆wrap_content・・・・・表示されるコンテンツに応じて最適な幅に調整する。 ◆match_parent・・・・・端から端までいっぱいに広げる。
android:layout_height	部品の横幅に関する設定を行う。 ◆wrap_content・・・・・表示されるコンテンツに応じて最適な幅に調整する。 ◆match_parent・・・・・端から端までいっぱいに広げる。
android:text	部品に表示するテキストの設定を行う。 ◆そのまま記述・・・・・記述した内容を表示する。 ◆@string/△△△・・・・・res/values/strings.xmlの△△△の値を表示する。
android:id	部品に対してのIDの設定を行う。 「@+id/名前」という形で設定する。

● 「android:text="@string/hello_world"」 の説明

strings.xml ファイルの hello_world 変数より、文字列「Hello World!」 を表示しています。

「activity_main.xml」の一部

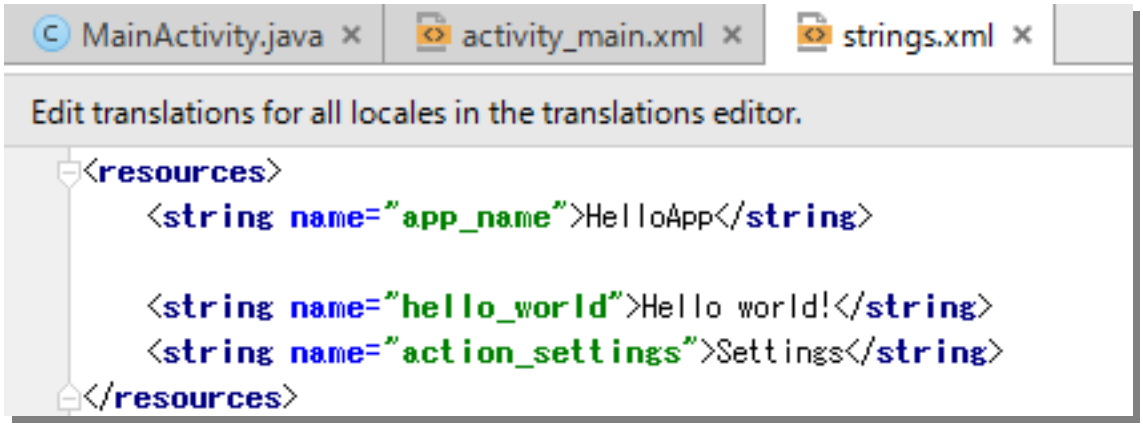
```
<TextView
    ・・・・・・・・・・・・・
    android:text="@string/ hello_world" />
```

「strings.xml」の一部

```
<resources>
    <string name="hello_world">Hello world!</string>
</resources>
```

4.5 strings.xml

部品に表示する文字列を定義しておくファイル



strings.xml は、部品に表示する文字列を定義しておくリソースファイルです。このファイルは、「res」フォルダ内の「values」フォルダに保存されており、このファイルに表示する文字列を設定しておくことで、部品に文字列を表示させることができます。

strings.xml を設定する方法は、「エディタ」あるいは「Translations エディタ」と呼ばれるツールで編集することができ、これを使用して文字列定義用のリソースファイルを作成することができる。

◆「strings.xml」のソースコード

下記プログラムは、新規プロジェクト作成時の初期設定段階でのプログラム内容です。

ファイル名：strings.xml

```
<resources>
    <string name="app_name">HelloApp</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
</resources>
```

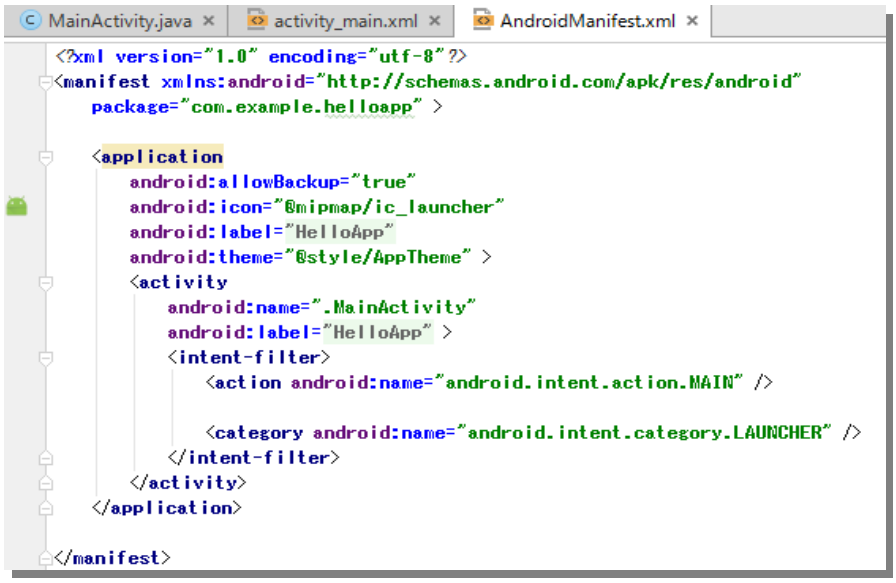
●タグの基本構成

resources タグの中で、設定する文字列を定義します。

```
<resources>
    <string name="名前1">値1</string>
    <string name="名前2">値2</string>
    <string name="名前3">値3</string>
</resources>
```

4.6 AndroidManifest.xml

Androidアプリに関する諸設定情報を定義するファイル



AndroidManifest.xml は、Android のマニフェストファイルといい、Android アプリケーションに関する諸設定情報を定義しておくファイルであり、Android アプリケーションに必ず 1 つ存在します。

AndroidManifest.xml を設定する方法は、エディタで編集することができます。

◆「AndroidManifest.xml」の主な役割

・アプリケーションのアイコンやタイトルの設定

・使用するコンポーネント（Activity、Service 等）の定義

・コンポーネントの振る舞いに関する定義

・アプリケーションのアクセス制限設定

・ライブラリの使用設定

◆「AndroidManifest.xml」のソースコード

下記プログラムは、新規プロジェクト作成時の初期設定段階でのプログラム内容です。

ファイル名：AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.helloapp" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

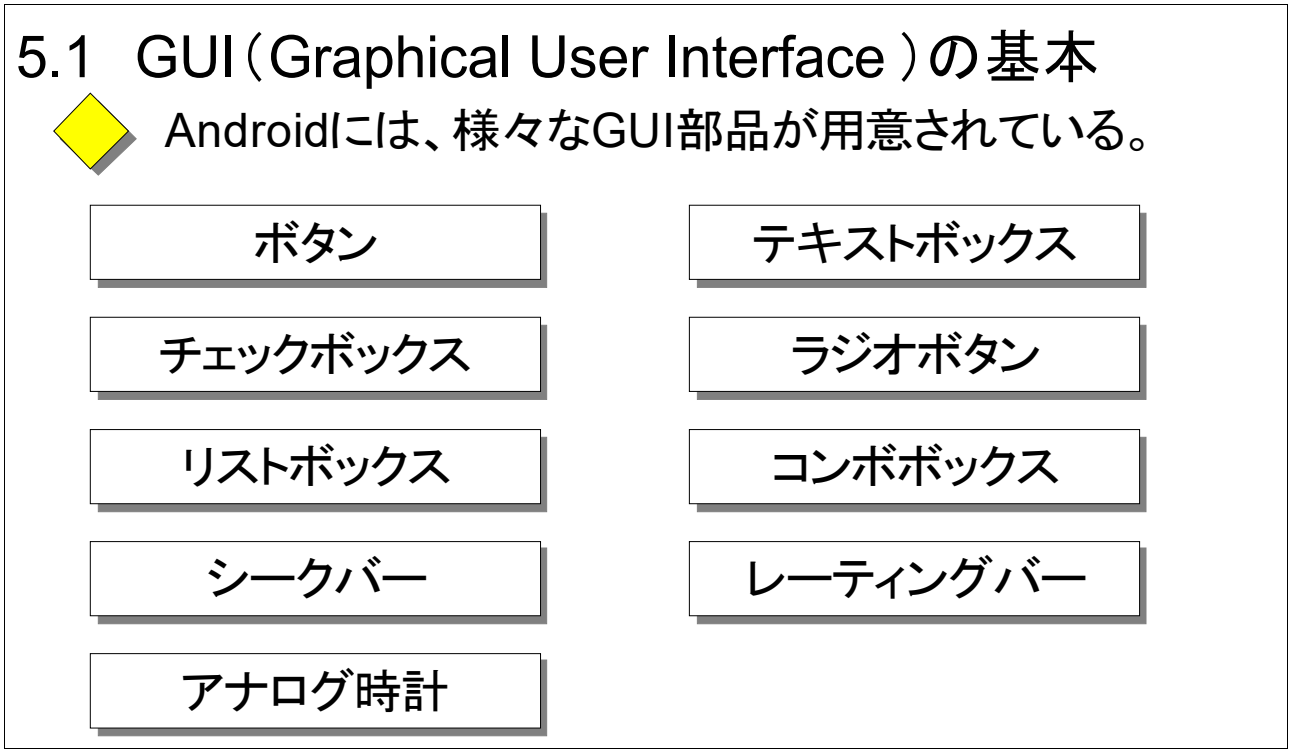
●タグの基本構成

manifest タグの中で、アクティビティ等を設定する。

```
<manifest>
    <application>
        <activity アクティビティの設定 >・・・・</activity>
    </application>
</manifest>
```


メモ

第5章
GUIの基本



Android には、様々な GUI の部品が用意されています。それらを組み合わせることで Android アプリケーションを構築することができます。

◆Android の主な GUI 部品

●Button・・・・・ボタン

●TextView・・・・・テキストボックス

●CheckBox・・・・・チェックボックス

●RadioButton・・・・・ラジオボタン

●ListView・・・・・リストボックス（複数行）

●Spinner・・・・・コンボボックス（単一列）

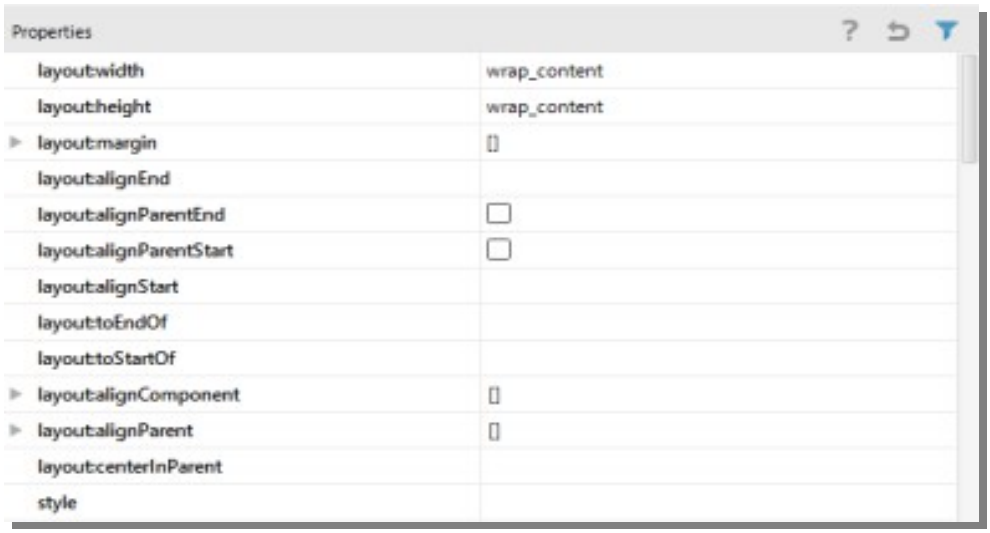
●SeekBar・・・・・シークバー（使用例：音量などの上げ下げ）

●RatingBar・・・・・レーティングバー（使用例：5 段階評価）

●AnalogClock・・・・・アナログ時計

5.2 GUI部品のプロパティ

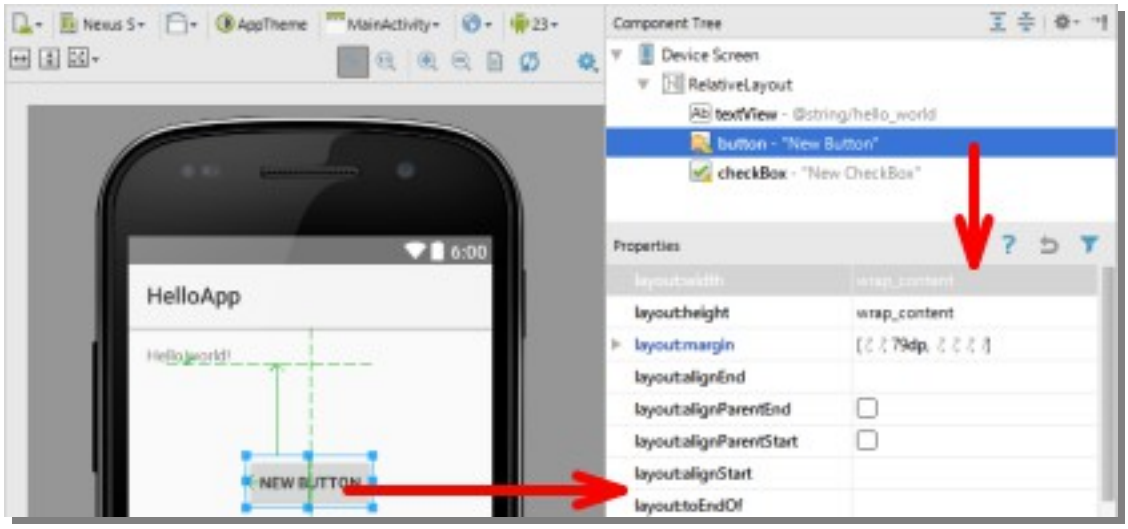
GUI部品には、様々な設定情報が用意されている。



画面に配置するGUI部品には、ID、テキスト、レイアウト情報など様々な設定情報が用意されています。これらを編集する時に使用するのがプロパティです。

◆プロパティ・ビューの表示方法

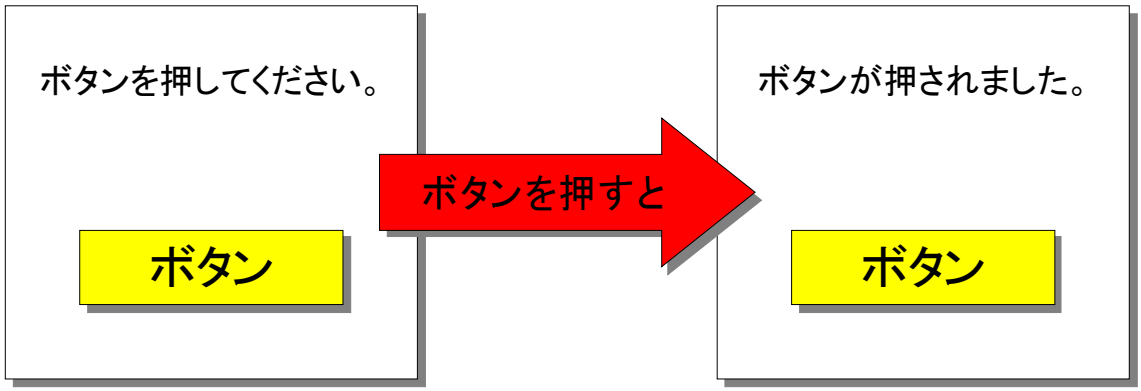
デバイススクリーンに配置した部品をクリックするか、「Component Tree」にリストアップされた部品をクリックするとプロパティを参照できるようになります。



5.3 サンプルプログラム1

アプリケーション名：SampleApp01

動作概要



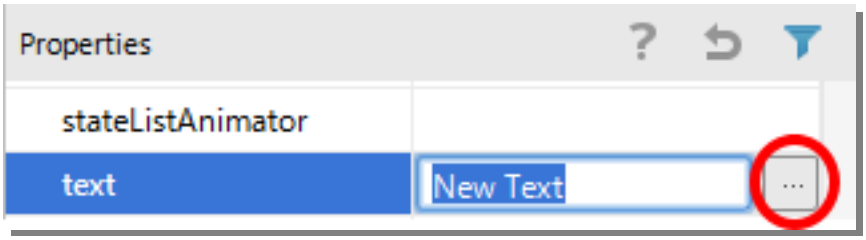
◆作成手順

①新規アプリケーション「SampleApp01」を作成する。

②「activity_main.xml」で「TextView（Plain TextView）」と「Button」を配置する。

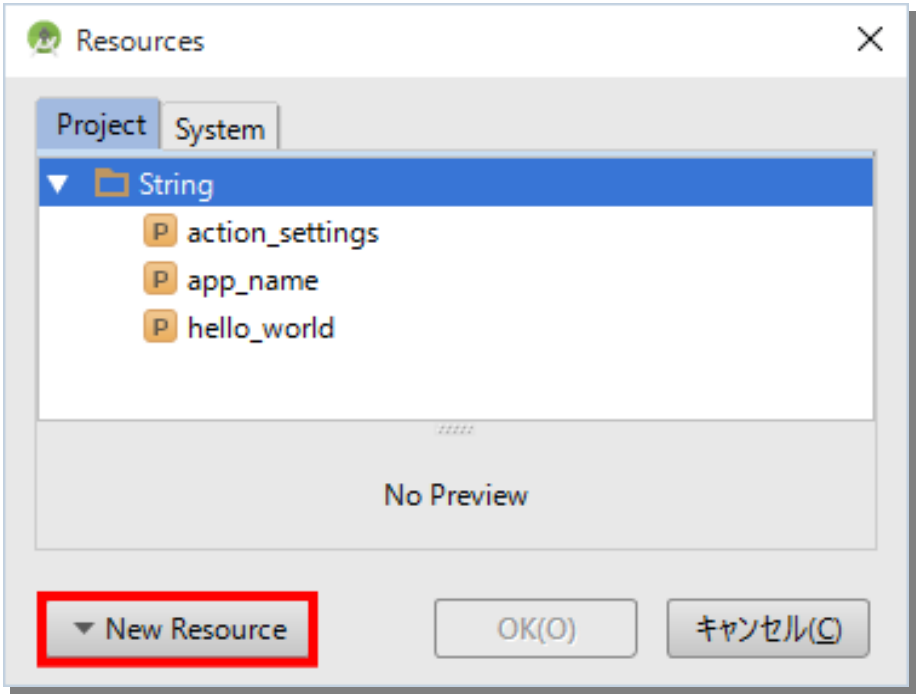
③「TextView」に文字列「ボタンを押してください。」を設定する。

●配置した「TextView」をクリックし、プロパティ・ビューから「Text」という項目を探し丸で囲んだ部分の「・・・」ボタンをクリックする。



● 「Resources」というダイアログの画面で、「New Resource」ボタンをクリックし

「New String Value...」を選択する。



● 「New String Value Resource」というダイアログの画面で、「Resource name」と「Resource value」の項目に次の情報を入力して[OK]ボタンを押す。

・Resource name：text_label01

・Resource value：ボタンを押してください。

● 「text_label01」が strings.xml に追加され、登録した文字列がデバイススクリーンにも反映される。

④「Button」に文字列「ボタン」を設定する。

●③と同様の手順で設定する。

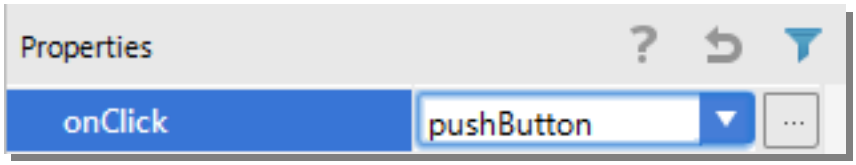
・Resource name：button_label01

・Resource value：ボタン

⑤ボタンをクリックした時の処理を設定する。

●配置した「Button」をクリックし、プロパティ・ビューから「onClick」という項目を探し

「pushButton」と入力する。これによりボタンが押されると pushButton メソッドが呼び出されるようになる。



●「MainActivity.java」に pushButton メソッドが実行された時の処理を記述する。

```
public void pushButton(View view) {
    TextView text = (TextView)findViewById(R.id.textView);
    text.setText("ボタンが押されました。");
}
```

★findViewById メソッド（Activity クラスに定義）

【書式】
public View findViewById(int id)

【戻り値】
Viewクラスのインスタンス

【機能】
Activityに組み込まれている部品のインスタンスを取得する。

★setText メソッド（TextView クラスに定義）

【書式】
public void setText(CharSequence str)

【機能】
部品の表示文字列を設定する。

⑥画面と各種ファイルは次のように設定される。

ファイル名：activity_main.xml

```
<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">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textView"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="41dp"
        android:text="@string/text_label01" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_label01"
        android:id="@+id/button"
        android:layout_below="@+id/textView"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="80dp"
        android:onClick="pushButton" />
</RelativeLayout>
```

ファイル名：strings.xml

```
<resources>

    <string name="app_name">SampleApp01</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="text_label01">ボタンを押してください。</string>
    <string name="button_label01">ボタン</string>

</resources>
```

ファイル名：R.java

```
package com.example.sampleapp01;

public final class R {
    public static final class id {
        public static final int action_settings=0x7f080002;
        public static final int button=0x7f080001;
        public static final int textView=0x7f080000;
    }
    public static final class layout {
        public static final int activity_main=0x7f030000;
    }
    public static final class string {
        public static final int action_settings=0x7f060000;
        public static final int app_name=0x7f060001;
        public static final int button_label01=0x7f060002;
        public static final int hello_world=0x7f060003;
        public static final int text_label01=0x7f060004;
    }
}
```

※一部抜粋

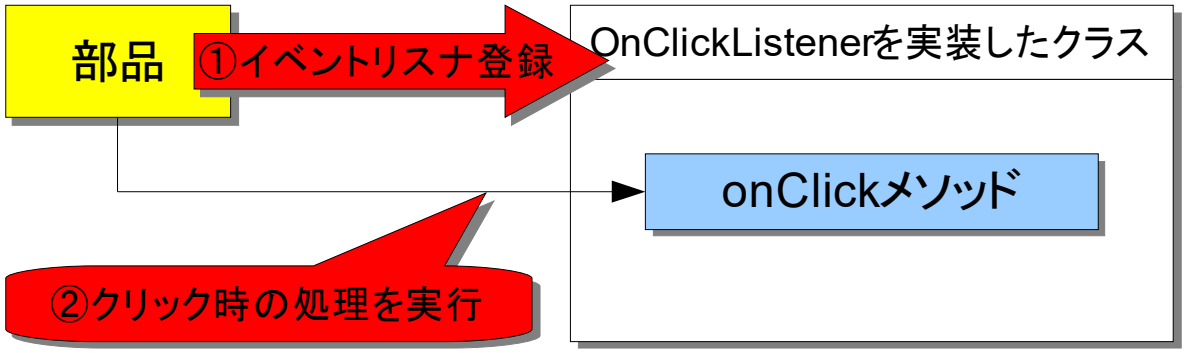
ファイル名：MainActivity.java

```
package com.example.sampleapp01;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;

public class MainActivity extends Activity {

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

    public void pushButton(View view) {
        TextView text = (TextView)findViewById(R.id.textView);
        text.setText("ボタンが押されました。");
    }
}
```

※一部抜粋

5.4 GUI部品のイベント処理

GUI部品のイベント処理を行う場合は
イベントリスナというインターフェースを実装したクラス
を作成し、このクラスをGUI部品と結びつける。



GUI 部品のイベント処理を行う場合は、イベントリスナというインターフェースを実装したクラスを作成し、このクラスを GUI 部品と結びつけます。これをイベントリスナ登録と言います。

例えば、部品がクリックされた時に実行されるクリックイベントの処理を記述するには、イベント処理を記述するクラスに View.OnClickListener インターフェースを実装し、 View クラスの setOnClickListener メソッドを使ってこのクラスと結び付けます。View.OnClickListener インターフェースには onClick メソッドが定義されており、実装先のクラスでオーバーライドすることにより、部品がクリックされた時に記述した処理を実行することができます。

◆OnClickListener インターフェース

OnClickListener インターフェースは View クラスの中で、次のように定義されています。

```
public static interface OnClickListener {
        public abstract void onClick(View view);
}
```

※onClickメソッドの引数には、クリックが発生したViewクラスのオブジェクトが渡される。

◆イベントリスナ登録

部品を OnClickListener にイベントリスナ登録を行うには、setOnClickListener メソッドを使用して行います。

【書式】
public void setOnClickListener(OnClickListener インターフェースを実装したオブジェクト)

【機能】
部品をOnClickListener にイベントリスナ登録を行う。

◆プログラム例

ボタンをクリックした時の処理を行うプログラムを記述する。プログラムの方法は３種類ある。

① OnClickListener インターフェースを実装したクラスを定義する例

ファイル名：MainActivity.java

```
package com.example.sampleapp01;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends Activity {

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

        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(new ButtonClickAdapter());
    }

    public class ButtonClickAdapter implements OnClickListener {
        @Override
        public void onClick(View view) {
            TextView text = (TextView)findViewById(R.id.textView);
            text.setText("ボタンが押されました。");
        }
    }
}
```

※一部抜粋

②匿名クラスを使用する例

ファイル名：MainActivity.java

```
package com.example.sampleapp01;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends Activity {

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

        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(
                new OnClickListener() {
                    @Override
                    public void onClick(View view) {
                        TextView text = (TextView)findViewById(R.id.textView);
                        text.setText("ボタンが押されました。");
                    }
                }
        );
    }
}
```

※一部抜粋

③ MainActivity クラスに OnClickListener インターフェースを実装した例

ファイル名：MainActivity.java

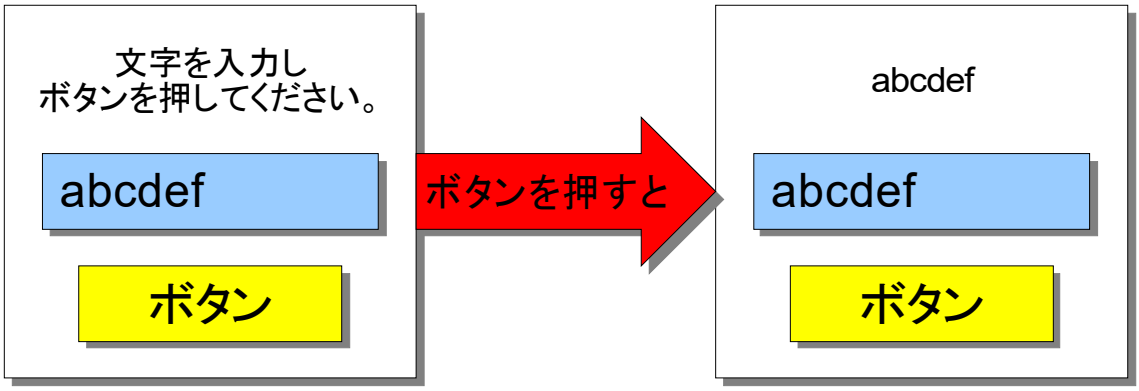
```
package com.example.sampleapp01;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends Activity implements OnClickListener {

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

        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
            TextView text = (TextView)findViewById(R.id.textView);
            text.setText("ボタンが押されました。");
    }
}
```

※一部抜粋

5.5 サンプルプログラム2

アプリケーション名：SampleApp02

動作概要



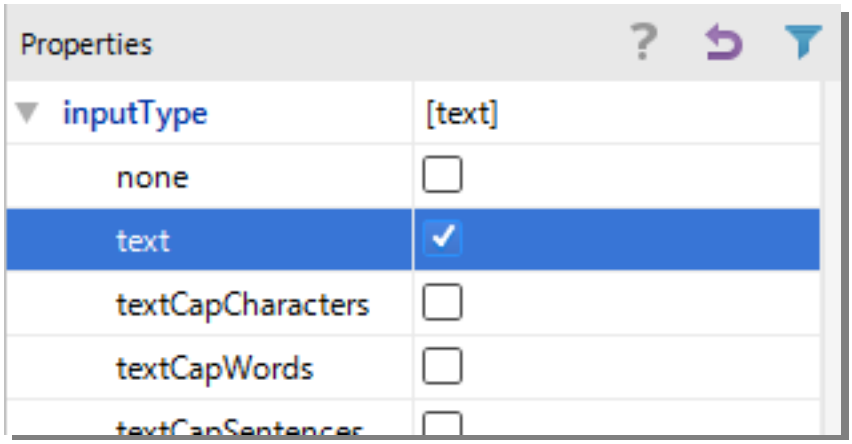
◆作成手順

①新規アプリケーション「SampleApp02」を作成する。

②「activity_main.xml」で「TextView（Plain TextView）」と「EditText（PlainText）」と「Button」を配置する。

●「TextView」と「Button」のプロパティ設定は、「サンプルプログラム 1」と同様にする。

●配置した「EditText」をクリックし、プロパティ・ビューから「Input Type」という項目を探し「text」と設定しておく。



③「MainActivity.java」でボタンをクリックした時の処理を設定する。

◆サンプルプログラム

ファイル名：MainActivity.java

```
package com.example.sampleapp02;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends Activity implements OnClickListener {

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

        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
        TextView text = (TextView)findViewById(R.id.textView);
        EditText edit = (EditText)findViewById(R.id.editText);

        CharSequence str = edit.getText();
        text.setText(str);
    }
}
```

※一部抜粋

★getText メソッド（EditText クラスに定義）

【書式】
public CharSequence getText()

【機能】
部品の文字列を取得する。

5.6 トースト（Toast）

Androidアプリケーションからユーザに対して動的に情報を表示する基本的な手段

Toast は、Android アプリケーションからユーザに対して、動的に情報を表示する基本的な手段のひとつです。

「短時間で自動的に表示が終了してしまう」や「ユーザーからの操作を受け付けることができない」などの理由から用途は限定されるが、一方で以下のような利点もあります。

◆Toast の利点

●表示が非常に簡単

●事前に専用のビューを用意する必要がない

●アプリケーションがフォーカスを持っていない状態でも表示できる

●他の UI に対する操作を阻害しない

◆Toast の基本的な使い方

基本的な Toast の表示には、Toast クラスの以下のメソッドを使用します。

★makeText メソッド（Toast クラスに定義）

【書式】
public static Toast makeText(Context context, CharSequence str, int duration)

【戻り値】
Toastクラスのインスタンス

【機能】
Toastを作成する。

・第 1 引数（Context）

Toast を組み込むコンテキストの指定を行う。

Activity クラスのオブジェクトが使用できるので、Activity クラス内で Toast を使用する場合、第 1 引数には this が指定される。

・第 2 引数（CharSequence）

Toast に表示するメッセージを設定する。

CharSequence 型での指定を求められるが、String 型を使用しても問題ない。また、あらかじめ文字列リソースとして登録した文字列を表示したい場合、第 2 引数には“R.string.xxxxx”といったリソース ID を指定できる。

・第 3 引数（int）

Toast を表示する時間を設定する。

Toast.LENGTH_SHORT、または Toast.LENGTH_LONG を指定する。

※LENGTH_SHORT の場合は約 2 秒、LENGTH_LONG の場合は約 4 秒間表示する Toast となる。

★show メソッド（Toast クラスに定義）

【書式】
public void show()

【機能】
Toastを表示する。

◆サンプルプログラム

「SampleApp02」の「MainActivity.java」を下記のように変更すると、Toast 表示ができる。

ファイル名 : MainActivity.java

```
package com.example.sampleapp02;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends Activity implements OnClickListener {

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

        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
        TextView text = (TextView)findViewById(R.id.textView);
        EditText edit = (EditText)findViewById(R.id.editText);

        CharSequence str = edit.getText();
        text.setText(str);

        Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
        toast.show();
    }
}
```

※一部抜粋

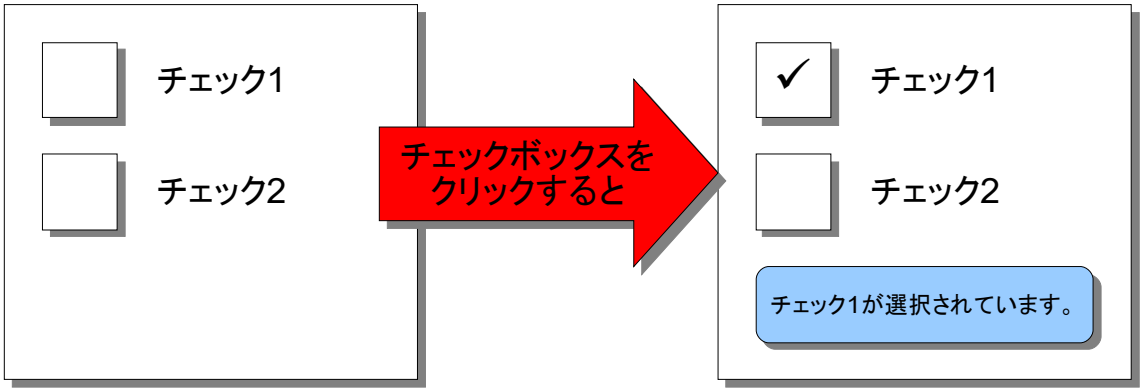
メモ

第 6 章
GUI の利用

6.1 CheckBoxの利用方法

アプリケーション名：CheckBoxApp

動作概要



◆作成手順

①新規アプリケーション「CheckBoxApp」を作成する。

②「activity_main.xml」で「CheckBox」を２つ配置する。

③「MainActivity.java」でチェックボックスをクリックした時のイベント処理を行う。

●OnCheckedChangeListener インターフェース

OnCheckedChangeListener インターフェースは CompoundButton クラスの中で、次のように定義されている。

```
public static interface OnCheckedChangeListener {
        public abstract void onCheckedChanged(CompoundButton buttonView, boolean isChecked);
}
```

※onCheckedChangedメソッドの第1引数には、クリックが発生したCompoundButtonクラスのオブジェクトが渡される。
※onCheckedChangedメソッドの第2引数には、クリック時の選択状態（trueかfalse）が渡される。

●イベントリスナ登録

部品を OnCheckedChangeListener にイベントリスナ登録を行うには、setOnCheckedChangeListener メソッドを使用して行う。

【書式】
public void setOnCheckedChangeListener(OnCheckedChangeListener　インターフェースを実装したオブジェクト)

【機能】
部品をOnCheckedChangeListener にイベントリスナ登録を行う。

●プログラム例

ファイル名：MainActivity.java

```
package com.example.checkboxapp;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.Toast;

public class MainActivity extends Activity implements CompoundButton.OnCheckedChangeListener{

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

        CheckBox checkBox1 = (CheckBox)findViewById(R.id.checkBox);
        checkBox1.setOnCheckedChangeListener(this);

        CheckBox checkBox2 = (CheckBox)findViewById(R.id.checkBox2);
        checkBox2.setOnCheckedChangeListener(this);
    }

    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        String str = "";

        switch (buttonView.getId()) {
            case R.id.checkBox : str = "チェック1が"; break;
            case R.id.checkBox2 : str = "チェック2が"; break;
        }

        str += isChecked ? "選択されています。" : "選択されていません。";

        Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
        toast.show();
    }
}
```

※一部抜粋

★getId()メソッド（CompoundButton クラスに定義）

【書式】
public int getId()

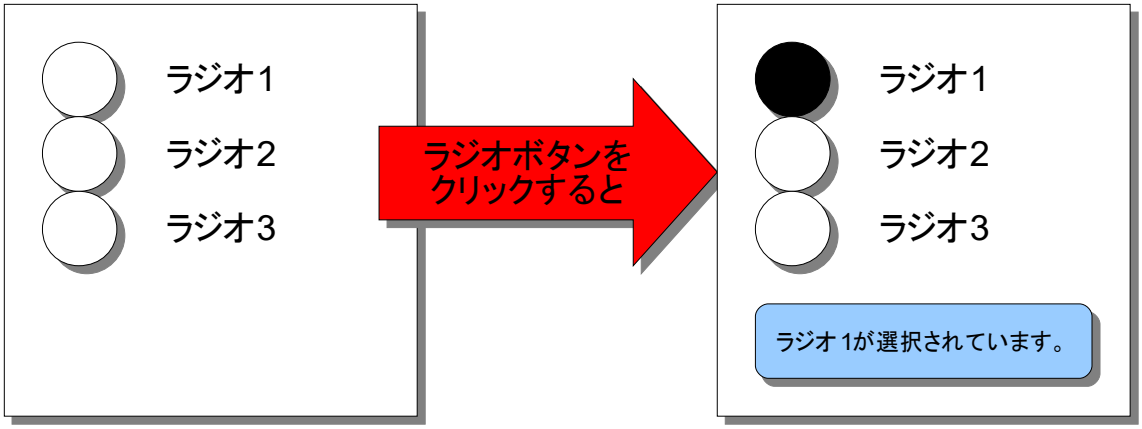
【戻り値】
ViewのリソースID

【機能】
ViewのリソースIDを取得する。

6.2 RadioButtonとRadioGroupの利用方法

アプリケーション名：RadioApp

動作概要



◆作成手順

①新規アプリケーション「RadioApp」を作成する。

②「activity_main.xml」で「RadioGroup」を配置する。

③３つの「RadioButton」に文字列を設定する。

④「MainActivity.java」でチェックボックスをクリックした時のイベント処理を行う。

●OnCheckedChangeListener インターフェース

OnCheckedChangeListener インターフェースは RadioGroup クラスの中で、次のように定義されている。

```
public static interface OnCheckedChangeListener {
        public abstract void onCheckedChanged(RadioGroup group, int checkedId);
}
```

※onCheckedChangedメソッドの第1引数には、クリックが発生したRadioGroupクラスのオブジェクトが渡される。
※onCheckedChangedメソッドの第2引数には、クリック時のRadioButtonのリソースIDが渡される。

●イベントリスナ登録

部品を OnCheckedChangeListener にイベントリスナ登録を行うには、setOnCheckedChangeListener メソッドを使用して行う。

【書式】
public void setOnCheckedChangeListener(OnCheckedChangeListener　インターフェースを実装したオブジェクト)

【機能】
部品をOnCheckedChangeListener にイベントリスナ登録を行う。

●プログラム例

ファイル名：activity_main.xml

```
<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">

    <RadioGroup
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:id="@+id/radioGroup">

        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="ラジオ1"
            android:id="@+id/radioButton"
            android:layout_gravity="center_horizontal" />

        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="ラジオ2"
            android:id="@+id/radioButton2"
            android:layout_gravity="center_horizontal" />

        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="ラジオ3"
            android:id="@+id/radioButton3"
            android:layout_gravity="center_horizontal" />
    </RadioGroup>
</RelativeLayout>
```

ファイル名 : MainActivity.java

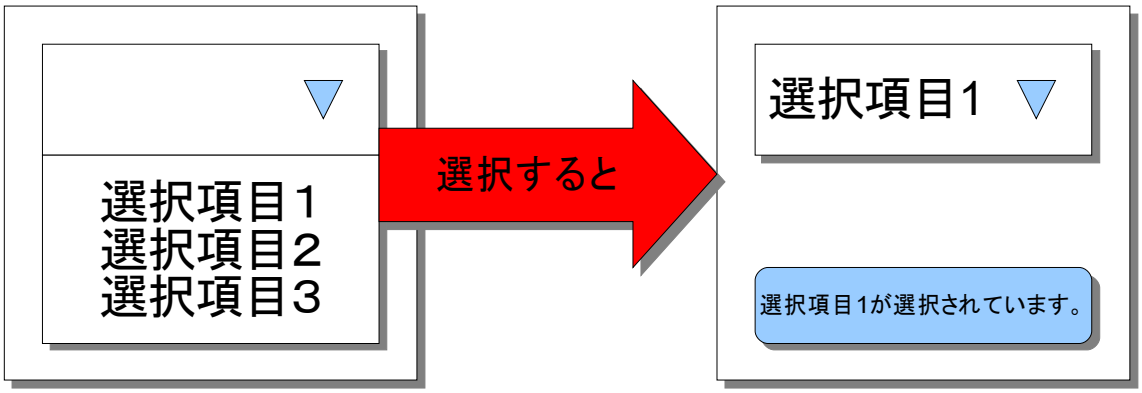
```
package com.example.radioapp;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.RadioGroup.OnCheckedChangeListener;
import android.widget.Toast;

public class MainActivity extends Activity implements OnCheckedChangeListener{

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

        RadioGroup group = (RadioGroup)findViewById(R.id.radioGroup);
        group.setOnCheckedChangeListener(this);
    }

    @Override
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        RadioButton button = (RadioButton)findViewById(checkedId);

        CharSequence str = button.getText() + "が選択されています。";

        Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
        toast.show();
    }
}
```

※一部抜粋

6.3 Spinnerの利用方法

アプリケーション名：SpinnerApp

動作概要



◆作成手順

①新規アプリケーション「SpinnerApp」を作成する。

②「activity_main.xml」で「Spinner」を配置する。

③「Spinner」に３つの選択項目を設定する。

●「strings.xml」に下記項目を設定する。

ファイル名：strings.xml

```
<resources>

    <string-array name="select_list">
        <item >選択項目1</item>
        <item >選択項目2</item>
        <item >選択項目3</item>
    </string-array>

</resources>
```

※一部抜粋

④「Spinner」に「strings.xml」で設定した配列「select_list」を組み込む。

●配置した「Spinner」をクリックし、プロパティ・ビューから「entries」という項目を探し「@array/select_list」と入力する。

●「activity_main.xml」は、結果的に下記のように設定される。

ファイル名：activity_main.xml

```
<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">

   <Spinner
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/spinner"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="26dp"
      android:focusable="false"
      android:entries="@array/select_list" />
</RelativeLayout>
```

⑤「MainActivity.java」で項目を選択した時のイベント処理を行う。

●OnItemSelectedListener インターフェース

OnItemSelectedListener インターフェースは AdapterView クラスの中で、次のように定義されている。

```
public static interface OnItemSelectedListener {
        public abstract void onItemSelected(AdapterView<?> parent, View view, int position, long id);
        public abstract void onNothingSelected(AdapterView<?> parent);
}
```

※onItemSelectedメソッドの第1引数には、イベントが発生したリストを呼び出した部品（Spinnerクラスなど）の
オブジェクトが渡される。
※onItemSelectedメソッドの第2引数には、表示されたリストが渡される。
※onItemSelectedメソッドの第3引数には、選択した項目の位置情報が渡される。
※onItemSelectedメソッドの第4引数には、選択した項目を示すID番号が渡される。

※onNothingSelectedメソッドの第1引数には、イベントが発生したリストを呼び出した部品（Spinnerクラスなど）の
オブジェクトが渡される。

●イベントリスナ登録

部品を OnItemSelectedListener にイベントリスナ登録を行うには、setOnItemSelectedListener メソッドを使用して行う。

【書式】
public void setOnItemSelectedListener(OnItemSelectedListener　インターフェースを実装したオブジェクト)

【機能】
部品をOnItemSelectedListener にイベントリスナ登録を行う。

●プログラム例

ファイル名：MainActivity.java

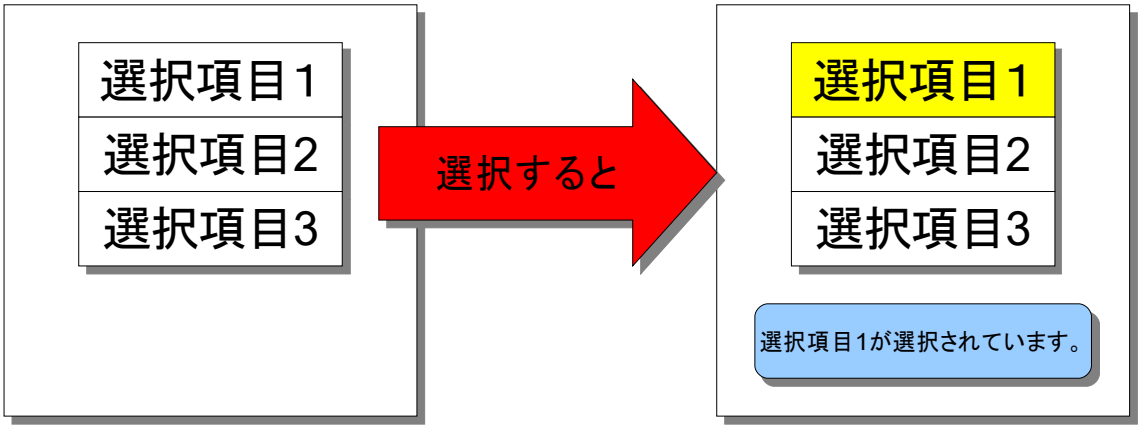
```
package com.example.spinnerapp;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.Spinner;
import android.widget.Toast;

public class MainActivity extends Activity implements AdapterView.OnItemSelectedListener{

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

        Spinner spinner = (Spinner)findViewById(R.id.spinner);
        spinner.setOnItemSelectedListener(this);
    }

    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        Spinner spinner = (Spinner)parent;

        String str = (String)spinner.getSelectedItem();
        str += "が選択されています。";

        Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
        toast.show();
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {
        Toast toast = Toast.makeText(this, "選択されていません。", Toast.LENGTH_SHORT);
        toast.show();
    }
}
```

※一部抜粋

6.4 ListViewの利用方法

アプリケーション名：ListViewApp

動作概要



◆作成手順

①新規アプリケーション「ListViewApp」を作成する。

②「activity_main.xml」で「ListView」を配置する。

③「ListView」に３つの選択項目を設定する。

●「Spinner」の設定と同じ方法で行う。

④「MainActivity.java」で項目を選択した時のイベント処理を行う。

●OnItemClickListener インターフェース

OnItemClickListener インターフェースは AdapterView クラスの中で、次のように定義されている。

```
public static interface OnItemClickListener {
        public abstract void onItemClick(AdapterView<?> parent, View view, int position, long id);
}
```

※onItemClickメソッドの第1引数には、イベントが発生したリストを呼び出した部品（ListViewクラスなど）の
　オブジェクトが渡される。
※onItemClickメソッドの第2引数には、表示されたリストが渡される。
※onItemClickメソッドの第3引数には、選択した項目の位置情報が渡される。
※onItemClickメソッドの第4引数には、選択した項目を示すID番号が渡される。

●イベントリスナ登録

部品を OnItemClickListener にイベントリスナ登録を行うには、setOnItemClickListener メソッドを使用して行う。

【書式】
public void setOnItemClickListener(OnItemClickListener インターフェースを実装したオブジェクト)

【機能】
部品をOnItemClickListenerにイベントリスナ登録を行う。

●プログラム例

ファイル名：MainActivity.java

```
package com.example.listviewapp;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.Toast;

public class MainActivity extends Activity implements AdapterView.OnItemClickListener{

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

        ListView list = (ListView)findViewById(R.id.listView);
        list.setOnItemClickListener(this);
    }

    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        ListView list = (ListView)parent;

        String str = (String)list.getItemAtPosition(position);
        str += "が選択されています。";

        Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
        toast.show();
    }
}
```

※一部抜粋

★getItemAtPosition()メソッド（ListView クラスに定義）

【書式】
public Object getItemAtPosition(int position)

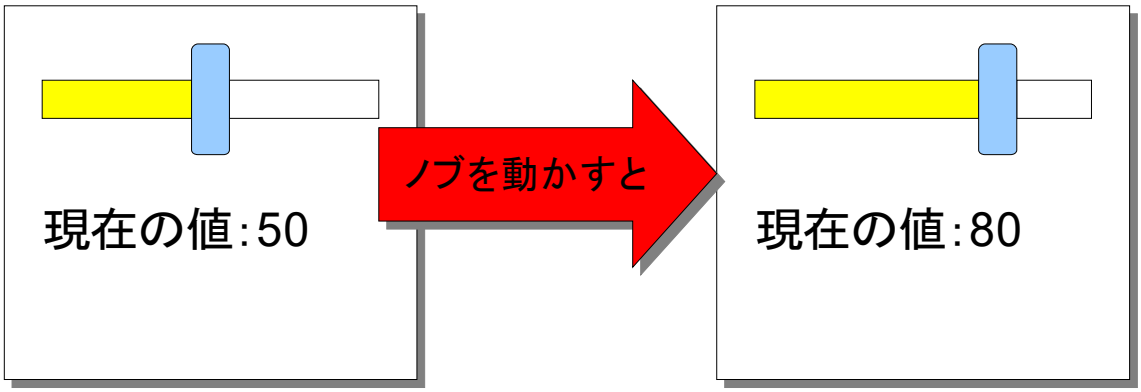
【戻り値】
指定したposition のオブジェクトを返す。

【機能】
選択した位置の文字列を取得する。

6.5 SeekBarの利用方法

アプリケーション名：SeekBarApp

動作概要



◆作成手順

①新規アプリケーション「SeekBarApp」を作成する。

②「activity_main.xml」で「SeekBar」と「TextView（Plain TextView）」を配置する。

③「SeekBar」のプロパティで下記項目を設定する。

●Max・・・100（最大値）

●Progress・・・0（現在値）

④「MainActivity.java」で項 SeekBar を操作した時のイベント処理を行う。

●OnSeekBarChangeListener インターフェース

OnSeekBarChangeListener インターフェースは SeekBar クラスの中で、次のように定義されている。

```
public static interface OnSeekBarChangeListener {
        public abstract void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser);
        public abstract void onStartTrackingTouch(SeekBar seekBar);
        public abstract void onStopTrackingTouch(SeekBar seekBar);
}
```

※3つのメソッドの第1引数には、イベントが発生したSeekBarクラスのオブジェクトが渡される。
※onProgressChangedメソッドの第2引数には、SeekBarの現在値が渡される。
※onProgressChangedメソッドの第3引数には、ユーザの操作によるものかどうかを示す情報が渡される。

●イベントリスナ登録

SeekBar を OnSeekBarChangeListener にイベントリスナ登録を行うには、setOnSeekBarChangeListener メソッドを使用して行う。

【書式】
public void setOnSeekBarChangeListener(OnSeekBarChangeListener　インターフェースを実装したオブジェクト)

【機能】
SeekBarをOnSeekBarChangeListener にイベントリスナ登録を行う。

●プログラム例

ファイル名 : MainActivity.java

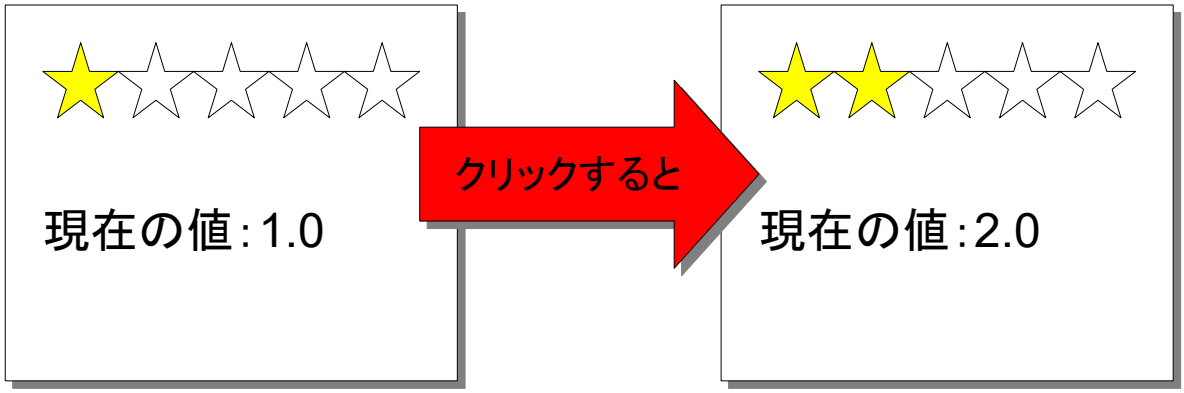
```
package com.example.seekbarapp;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.SeekBar;
import android.widget.SeekBar.OnSeekBarChangeListener;
import android.widget.TextView;

public class MainActivity extends Activity implements OnSeekBarChangeListener {

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

        SeekBar seekBar = (SeekBar)findViewById(R.id.seekBar);
        seekBar.setOnSeekBarChangeListener(this);
    }

    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
        TextView text = (TextView)findViewById(R.id.textView);

        CharSequence str = "現在の値 : " + progress;
        text.setText(str);
    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {

    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {

    }
}
```

※一部抜粋

6.6 RatingBarの利用方法

アプリケーション名：RatingBarApp

動作概要



◆作成手順

①新規アプリケーション「RatingBarApp」を作成する。

②「activity_main.xml」で「RatingBar」と「TextView（Plain TextView）」を配置する。

③「MainActivity.java」で項 RatingBar を操作した時のイベント処理を行う。

●OnRatingBarChangeListener インターフェース

OnRatingBarChangeListener インターフェースは RatingBar クラスの中で、次のように定義されている。

```
public static interface OnRatingBarChangeListener {
        public abstract void onRatingChanged(RatingBar ratingBar, float rating, boolean fromUser);
}
```

※onRatingChangedメソッドの第1引数には、イベントが発生したRatingBarクラスのオブジェクトが渡される。
※onRatingChangedメソッドの第2引数には、RatingBarの現在値が渡される。
※onRatingChangedメソッドの第3引数には、ユーザの操作によるものかどうかを示す情報が渡される。

●イベントリスナ登録

RatingBar を OnRatingBarChangeListener にイベントリスナ登録を行うには

setOnRatingBarChangeListener メソッドを使用して行う。

【書式】
public void setOnRatingBarChangeListener(OnRatingBarChangeListener　インターフェースを実装したオブジェクト)

【機能】
RatingBarをOnRatingBarChangeListener にイベントリスナ登録を行う。

●プログラム例

ファイル名：MainActivity.java

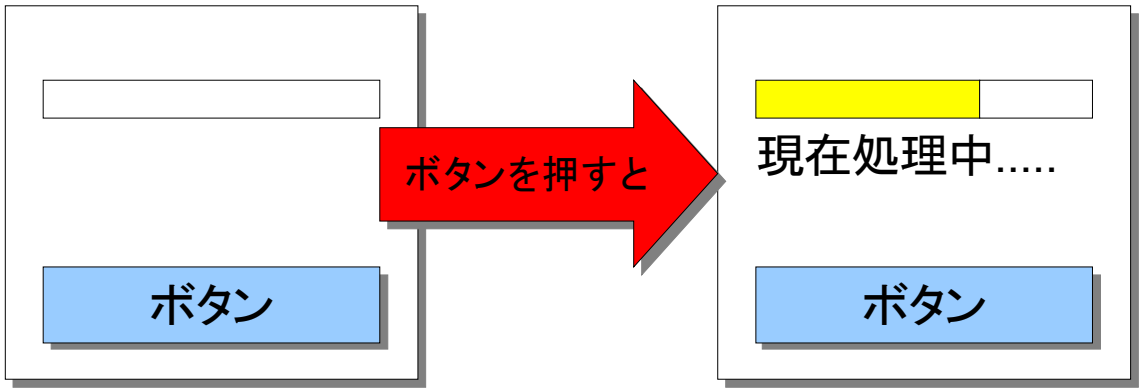
```
package com.example.ratingbarapp;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.RatingBar;
import android.widget.RatingBar.OnRatingBarChangeListener;
import android.widget.TextView;

public class MainActivity extends Activity implements OnRatingBarChangeListener {

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

        RatingBar ratingBar = (RatingBar)findViewById(R.id.ratingBar);
        ratingBar.setOnRatingBarChangeListener(this);    }

    @Override
    public void onRatingChanged(RatingBar ratingBar, float rating, boolean fromUser) {
        CharSequence str = "現在の値　：　" + rating;

        TextView text = (TextView)findViewById(R.id.textView);
        text.setText(str);
    }
}
```

※一部抜粋

6.7 ProgressBarの利用方法

アプリケーション名：ProgressBarApp

動作概要



◆作成手順

①新規アプリケーション「ProgressBarApp」を作成する。

②「activity_main.xml」で「ProgressBar（Horizontal）」と「TextView（Plain TextView）」と「Button」を配置する。

③「ProgressBar」のプロパティで下記項目を設定する。

●Visibility・・・Gone（ProgressBar を非表示にする設定）

④「MainActivity.java」で項 Button をクリックした時のイベント処理を行う。

●プログラム例

ファイル名：MainActivity.java

```
package com.example.progressbarapp;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.TextView;

public class MainActivity extends Activity implements OnClickListener {

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

        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(this);
    }
```

```
    @Override
    public void onClick(View view) {
            ProgressBar bar = (ProgressBar)findViewById(R.id.progressBar);
            TextView text = (TextView)findViewById(R.id.textView);

            int nowValue = bar.getProgress();
            int maxValue = bar.getMax();

            CharSequence str = "";

            if (maxValue == nowValue) {
                nowValue = 0;
            } else {
                bar.setVisibility(View.VISIBLE);
                nowValue++;
            }

            if (nowValue == 0) {
                bar.setVisibility(View.GONE);
                str = "ボタンを押してください。";
            } else {
                str = "現在処理中..." + "[" + nowValue + "]";
            }

            bar.setProgress(nowValue);
            text.setText(str);
    }
}
```

※一部抜粋

★setVisibility()メソッド（ProgressBar クラスに定義）

【書式】
public void setVisibility(int visibility)

【機能】
Viewの表示/非表示を設定する。

※引数 visibility には、可視状態を表す定数を設定する。

・view.VISIBLE・・・表示

・view.INVISIBLE・・・非表示（非表示にしたスペースは詰めない）

・view.GONE・・・非表示（非表示にしたスペースを詰める）

メモ