

Android 2.x / 4.x 対応

Android

プログラミング Bible

初級 基礎編

河西 朝雄 著



KASAI . SOFTWARELAB

定価 1,000 円 (税込)

はじめに

Androidは、スマートフォンやタブレットPCなどの携帯情報端末を主なターゲットとしたプラットフォーム(OS)で、Linuxカーネル層、ライブラリ層、Androidランタイム層、アプリケーションフレームワーク層、アプリケーション層などで構成されます。Androidのアプリケーションを開発するための言語はJavaとXMLです。

AndroidやiPhoneなどのスマートフォンやiPadなどのタブレット端末のユーザーインターフェースは指のタッチを基本とし、カメラやセンサを内蔵し、音声認識・音声合成などが簡単に利用できる画期的なコンピュータです。マウス、キーボード、ディスプレイが主なユーザーインターフェースとするパソコンとは大きく異なります。「コンピュータ=パソコン」の時代から「コンピュータ=スマートフォン、タブレット端末」の時代に急速にパラダイムシフトしようとしています。スマートフォンは子供から女性、シニアまでの広い層に渡って、今までのパソコンユーザとは比べ物にならない数のユーザが見込まれます。

スマートフォンをiPhone VS Androidという構図で見た場合どちらにもメリット、デメリットがあり、一概にどちらが良いとは言えません。アプリケーションの開発言語の違いで見るとiPhone(OS名はiOS)はObjective-C、AndroidはJavaです。Objective-Cはややマイナーな言語であるのに対しJavaはネットワーク関連ではメジャーな言語であるということです。このため数多くいるJava経験者にはAndroidの方が移行しやすい環境であると思います。

本書はAndroidのアプリを開発することを目的にしていますので、話をAndroidに絞ります。Androidは2007年にGoogleを中心にした規格団体「Open Handset Alliance」から発表され、2008年からAndroid対応のスマートフォンが多数販売されるようになりました。また、アプリケーションマーケットであるAndroid Marketが提供されていて、2011年5月時点で有料、無料含め30万を超えるアプリケーションが提供されています。Google Play(旧Android Market)を通して企業だけでなく、一般ユーザーが自作のアプリケーションを販売することができる点もいままでにない利点です。つまり、ソフト会社の技術者以外にも、学生を中心に一般の人でもAndroidアプリで商売ができるようになる可能性がありAndroidアプリ市場は今後急速に普及すると思います。

本シリーズは、Androidアプリを開発するための基本的なテクニックをすべて網羅するように34の章(カテゴリ)に分類し、「初級 基礎編」、「中級 Android的プログラミング法」、「上級 各種処理」の3分冊で構成することにし、本書はその中の「初級 基礎編」です。

34の章というのはいちばん多い章分けですが、細かく章分けをすることでカテゴリが分かり易く、各章のサイズは小さくなり初心者には、ひとつのまとまった単位がボリュームが少ないので、取りかかり易くなります。また、章の順序ではなく、知りたい章を先に学習することもできます。

既存の書籍やネット上の情報は重要な内容とそうでない情報がまぜこぜになっていたり、このプログラムをどこに書けばいいのかが曖昧だったり、サンプルが長すぎたりなど、初心者には理解しにくい内容が多いです。本シリーズではAndroidアプリを作る上で必要な技術的要素やテクニックを切り出し短いサンプルを付けて簡潔に提示します。

「初級 基礎編」は何らかの言語でプログラム経験はあるが、JavaやAndroidアプリを初めて勉強する人を主な対象とします。Androidアプリを作るためにはJavaとXMLの知識が必要になります。JavaやXMLを本格的に学ぶにはそれぞれ入門書が必要になります。本書ではAndroidアプリを作りながらJavaもXMLも手っ取り早く学べるように工夫してあります。そこでまず、Androidグラフィックスを利用して画面に文字、直線、イメージなどを描画するプログラムを例にJavaの基本的な言語仕様について2章で学びます。AndroidではテキストビューやボタンなどのGUI部品をウィジェット(Widget)と呼んでいます。ウィジェットはmain.xmlというXMLファイル中で定義します。3章ではウィジェットを例にXMLについて学びます。4章ではウィジェットを配置するレイアウトについて説明します。5章ではXMLを使わずにウィジェットやレイアウトをJavaコードで記述する方法を説明します。6章、7章ではウィジェット以外のユーザーインターフェースとしてメニュー、-toast、ダイアログ、ログについて説明します。8章、9章では画面のタッチで発生するタッチイベント、キー操作で発生するキーイベントなどの各種イベント処理について説明します。ということで本書は次のような章の構成となります。

- Chapter01 JavaによるAndroidアプリの作り方
- Chapter02 AndroidグラフィックスによるJava入門
- Chapter03 ウィジェットとXML
- Chapter04 レイアウト
- Chapter05 main.xmlを使わずにレイアウトする
- Chapter06 メニュー
- Chapter07 トースト、ダイアログ、ログ
- Chapter08 タッチイベント
- Chapter09 キーイベント、フォーカスイベント

これからAndroidアプリの開発を志す方々にとって、本書が少しでもお役に立てば幸いです。

2013年1月 河西朝雄

本書のプログラムは「Eclipse 3.6 Helios」と「Android 2.2(API 8)」で開発しエミュレータAVDの画面サイズはWVGA(480×800)です。実機は「SAMSUNG GALAXY S」で確認しました。

本書のプログラムはエミュレータAVDの画面サイズをHVGA(320×480)でも確認しました。また「Eclipse 3.7 Indigo」と「Android 4.0.3(API 15)」でも確認しました。これらの環境において差異が生じるものは、その差異について個々の例題に「注」として記述しました。Android, Android SDK, Eclipseの特徴と注意点に関して「付録 Android, Android SDK, Eclipseのバージョン」にまとめてあります。Android, Android SDK, Eclipseの最新情報についてはカサイ.ソフトウェアラボの電子書籍サイト(<http://kasailab.jp/>)を参照して下さい。

AndroidプログラミングBibleシリーズの他の本

AndroidプログラミングBibleシリーズは「初級 基礎編」、「中級 Android的プログラミング法」、「上級 各種処理」の3分冊構成です。本書は「初級 基礎編」です。他の本の内容は以下です。

☆中級 Android的プログラミング法

- Chapter10 インテントとアクティビティ
- Chapter11 Thread,Handler,Message
- Chapter12 サービス
- Chapter13 ブロードキャストレシーバ
- Chapter14 コンテンツプロバイダ
- Chapter15 マニフェスト
- Chapter16 基本ウィジェットを機能強化したウィジェット
- Chapter17 小物ウィジェット
- Chapter18 高度なビュー系ウィジェット
- Chapter19 アプリケーション・ウィジェット
- Chapter20 マルチメディア
- Chapter21 リソース
- Chapter22 アニメーション

☆上級 各種処理

- Chapter23 グラフィックス
- Chapter24 SurfaceView
- Chapter25 OpenGL
- Chapter26 ファイル処理
- Chapter27 SQLite
- Chapter28 Gmail
- Chapter29 GoogleMap
- Chapter30 センサー(実機のみ)
- Chapter31 カメラ(実機のみ)
- Chapter32 音声認識(実機のみ)
- Chapter33 音声合成
- Chapter34 ネットワーク通信

CONTENTS

Chapter 01 JavaによるAndroidアプリの作り方..... 10

1-1	プロジェクトの作り方.....	11
1-2	作成されたファイル	15
1-3	パッケージ・エクスプローラとフォルダの関係	18
1-4	作成したプログラムの実行.....	20
1-5	表示内容を変えてみる.....	22
1-6	Androidアプリの典型的なJavaコードの意味	24

Chapter 02 AndroidグラフィックスによるJava入門..... 30

2-1	Androidグラフィックスの基礎.....	31
2-2	for文による繰り返し	34
2-3	イメージの表示.....	38
2-4	if else文による条件判定.....	41
2-5	二重ループ	43
2-6	else if文.....	45
2-7	配列.....	47
2-8	ユーザ定義メソッド	49
☆	応用サンプル タートル・グラフィックス.....	51

Chapter 03 ウィジェットとXML 54

3-1	AndroidのXMLファイル	55
3-2	ボタンとクリック・リスナー.....	63
3-3	エディットテキスト (EditText)	66
3-4	チェックボックス (CheckBox)	70
3-5	ラジオボタン (RadioGroup と RadioButton)	72
3-6	スピナー (Spinner)	74
3-7	リストビュー (ListView)	77
3-8	イメージビュー (ImageView)	80
3-9	Viewクラスの属性.....	85
3-10	LinearLayoutクラスの属性	89

CONTENTS

3-11	TextView クラスの属性	92
3-12	EditText クラスの属性	99
3-13	ImageView クラスの属性.....	101
3-14	ArrayAdapter.....	104
3-15	onClick リスナーの作り方	108
3-16	ウィジェット自身にリスナーを付ける.....	111
3-17	AdapterView を使ったデータのバインディング	117
3-18	「Graphical Layout」で配置できるウィジェットとレイアウト.....	131

Chapter 04 レイアウト..... 134

4-1	LinearLayout (リニア・レイアウト)	135
4-2	レイアウトのネスト	139
4-3	RelativeLayout (相対レイアウト)	141
4-4	FrameLayout (フレーム・レイアウト).....	145
4-5	TableLayout (テーブル・レイアウト)	152
4-6	スタイル	161
4-7	テーマ	165
☆応用サンプル	入力フォーム.....	169

Chapter 05 main.xml を使わずにレイアウトする..... 172

5-1	LinearLayout を Java コードで配置	173
5-2	FrameLayout を Java コードで配置	175
5-3	Gravity を指定	178
5-4	RelativeLayout を Java コードで配置.....	180
5-5	TableLayout を Java コードで配置.....	183
5-6	main.xml と Java コードの併用	185
5-7	main.xml のレイアウトを取得し、そこにウィジェットを追加する	187
5-8	複数のボタンを配置しイベントリスナーを付ける	189
5-9	ウィジェットと View の併存.....	191
5-10	LayoutInflater.....	194

CONTENTS

5-11	画面制御.....	198
5-12	画面サイズに依存しないコード.....	212
Chapter 06 メニュー.....		222
6-1	メニューの表示.....	223
6-2	ポップアップメニュー(サブメニュー).....	226
6-3	「More」項目.....	228
6-4	チェック付きメニュー.....	230
6-5	メニューのグループ化.....	233
6-6	menu.xmlを使わずにJavaコードでメニューを作る.....	236
6-7	コンテキストメニュー.....	239
Chapter 07 トースト、ダイアログ、ログ.....		242
7-1	トースト.....	243
7-2	カスタム・トースト.....	245
7-3	アラート・ダイアログ.....	248
7-4	AlertDialogにリスト項目の表示.....	252
7-5	プログレス・ダイアログ.....	255
7-6	日付選択、時刻選択ダイアログ.....	258
7-7	カスタム・ダイアログ.....	261
7-8	ログ.....	264
☆応用サンプル	食文化判定.....	266
Chapter 08 タッチイベント.....		270
8-1	タッチアクションの種類.....	271
8-2	タッチムーブでイメージを移動.....	273
8-3	タッチされたイメージの判定.....	275
8-4	複雑なタッチ動作.....	278
8-5	スクロール,フリック(フリング).....	283
8-6	マルチタッチ(実機でのみ動作).....	287
8-7	ピンチ(実機でのみ動作).....	290

CONTENTS

8-8 イベントリスナー、イベントハンドラの種類.....	293
☆応用サンプル 羅針盤	298
☆応用サンプル 相性占い	300
Chapter 09 キーイベント、フォーカスイベント	306
9-1 キーイベントの種類	307
9-2 Menu キーをフックする	310
9-3 Back キーをフックする	312
9-4 通常のキーボードのキーイベントの取得	315
9-5 十字キーのキーイベントの取得.....	317
9-6 ソフトキーの作成.....	319
9-7 フォーカスイベント	324
9-8 タッチモードとフォーカスのハンドリング	327
9-9 IMEとソフトキー	332
付録 Android,Android SDK,Eclipse のバージョン	338

Chapter 01

Java による Android アプリの作り方

以下の作業が終わってEclipseでAndroidアプリを開発する環境が整備されているものとして話を進めます。

- Android SDKのインストール
- Eclipse (日本語版)のインストール
- Eclipse へのAndroid Plugin (ADT)のインストール
- Androidアプリを実行するために必要なAVD (Android Virtual Device)の作成

この章ではデフォルトのスケルトン(システムが自動生成するプログラムの骨格)を使って「Hello World,Test1!」というメッセージをTextView(テキストビュー)に表示するプログラムの作り方の手順を説明します。また作成されたファイルの意味と役割を説明します。



「注」

Eclipse や Android SDK のバージョンにより作業手順が異なる場合があります。

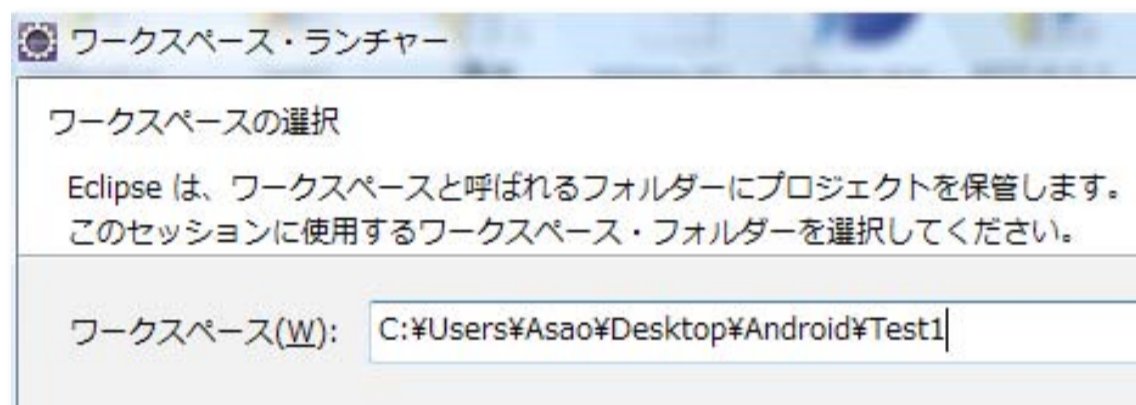
Android アプリを作るための環境設定についてはカサイ・ソフトウェアラボの電子書籍サイト

<http://kasailab.jp/> を参照してください。

Android アプリはプロジェクトで管理します。プロジェクトを保管するためのフォルダをワークスペースと呼びます。

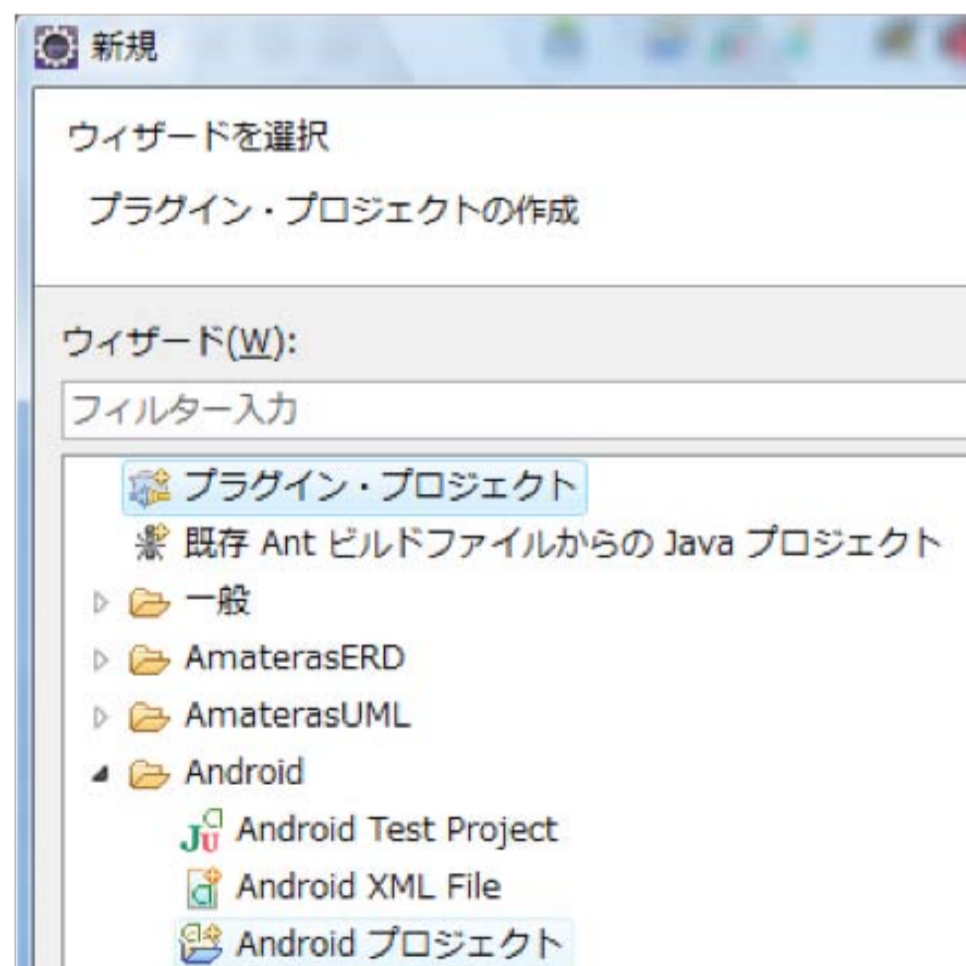
1. ワークスペースの作成

Eclipse を起動し「ワークスペース・ランチャー」画面でワークスペース名を入力します。ここでは、デスクトップの「Android」フォルダにワークスペースを「Test1」として作成します。



2. プロジェクトの作成

①「ファイル」－「新規」－「その他」を選択し、「Android」－「Android プロジェクト」を選択します。

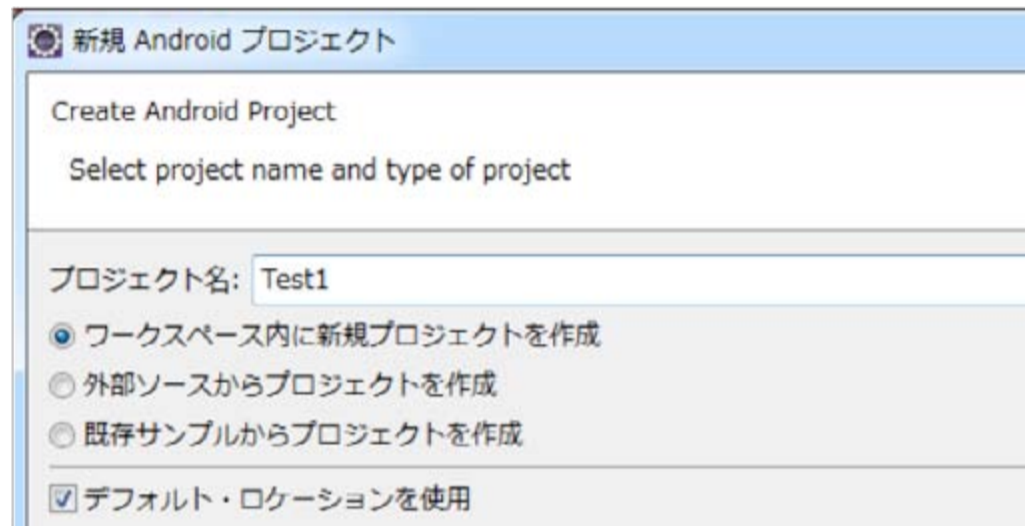


②プロジェクト名等の入力

プロジェクト名、アプリケーション名、Activityクラス名を「Test1」、パッケージ名を「jp.test1」とします。

■プロジェクト名

1つのJavaアプリを構成する各種ファイルを管理するための基本をプロジェクトと呼びます。プロジェクト名のフォルダ内に各種ファイルが格納されます。



■ビルドターゲット

AndroidSDKのバージョンを選択します。ここでは「Android 2.2」を選択しました。

ターゲット名	ベンダー	プラットフォーム	API レベル
<input checked="" type="checkbox"/> Android 2.2	Android Open Source Project	2.2	8
<input type="checkbox"/> Google APIs	Google Inc.	2.2	8
<input type="checkbox"/> Android 4.0.3	Android Open Source Project	4.0.3	15
<input type="checkbox"/> Google APIs	Google Inc.	4.0.3	15

■アプリケーション名、パッケージ名、Activityクラス名

以下の各項目にアプリケーション名、パッケージ名、Activityクラス名を入力します。

アプリケーション名:	Test1
パッケージ名:	jp.test1
<input checked="" type="checkbox"/> アクティビティの作成:	Test1
Minimum SDK:	8

「注」デフォルトのActivityクラス名は「Test1Activity」ですが「Test1」とします。

「注」「アクティビティの作成」とはAndroid画面の基本はActivityで、このクラスを自動的に作る場合ににチェックを入れ、そのクラス名を入力します。

「注」 Min SDK Version

アプリケーションが動作する最低の Android SDK バージョン の API レベルを整数値で指定します。Android 2.2 の場合は「8」を指定しますが、指定しなくてもよいです。

Android SDK バージョン	API レベル	コードネーム
4.2	17	Jelly Bean
4.1	16	Jelly Bean
4.0.3	15	Ice Cream Sandwich
4.0	14	Ice Cream Sandwich
3.2	13	Honeycomb
3.1	12	
3.0	11	
2.3.4(2.3.3)	10	Gingerbread
2.3	9	
2.2	8	Froyo
2.1	7	Eclair
2.0.1	6	
2.0	5	
1.6	4	Donut
1.5	3	Cupcake
1.1	2	非公開
1.0	1	非公開

「注」各種名前の意味**■ワークスペース名**

複数のプロジェクトを保管する一番元になるフォルダの名前です。

■プロジェクト名

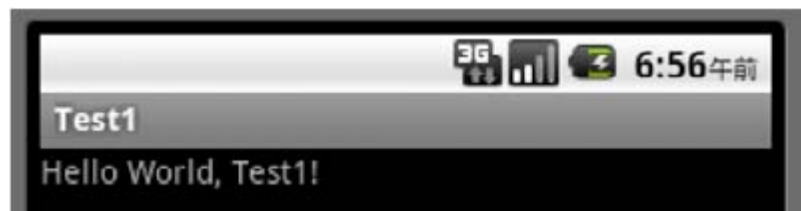
1つのJavaアプリを構成する各種ファイルを管理するための基本をプロジェクトと呼びます。プロジェクト名のフォルダ内に各種ファイルが格納されます。

■アプリケーション名

Javaアプリの中に埋め込まれる名前をアプリケーション名と呼びます。アプリケーション名は「app_name」の値として次のようにres/values/string.xml内に埋め込まれます。

```
<resources>
  <string name="hello">Hello World, Test1!</string>
  <string name="app_name">Test1</string>
</resources>
```

実行時にタイトルバーにこのアプリケーション名が表示されます。



■ パッケージ名

関連するクラスやインタフェースを1つにまとめた単位をパッケージと呼びます。パッケージにまとめることにより、機能ごとのカテゴリとして階層構造で分類することができます。異なるパッケージでは、同じ名前のクラスやインタフェースがあってもそれぞれ別なものとして扱えるため、名前の衝突を防ぐことができます。パッケージ名は「.」で区切って階層構造の名前を付けます。パッケージ名はすべて小文字にする慣習があります。各社が同じような名前でアプリを開発したときに名前の衝突が起きないように、会社のドメイン名などを入れるのが一般的です。ある会社のドメイン名がたとえば、「kasai.co.jp」であるとパッケージ名は、ドメイン名を逆順にならべた「jp.co.kasai.」を先頭にし、それぞれのパッケージ名を階層構造で付けて「jp.co.kasai.android.camaera」や「jp.co.kasai.java.util」などとなります。この他に「com.会社名.プロジェクト名.機能名」のような命名規則も多く見られます。

■ クラス名

Java アプリは複数のクラスで構成されますが、public指定されたクラスがそのアプリを代表するクラス名となります。「2. プロジェクトの作成」で指定したActivityクラスの名前を使って以下のようなクラスが生成されます。そしてこのクラス名と同じ名前のJavaソースファイル「Test1.java」が作成されます。

```
public class Test1 extends Activity {
```

「注」本書での命名規則
本書のアプリは小規模のため「ワークスペース名=プロジェクト名=アプリケーション名」とし、パッケージ名は「jp.プロジェクト名をすべて小文字にしたもの」とします。

Chapter 02

Android グラフィックスによる Java 入門

Android アプリを作るためには Java と XML の知識が必要になります。Java や XML を本格的に学ぶにはそれぞれ入門書が必要になります。本書では Android アプリを作りながら Java も XML も手っ取り早く学べるように工夫してあります。そこでまず、Android グラフィックスを利用して画面に文字、直線、イメージなどを描画するプログラムを例に Java の基本的な言語仕様について学びます。Android グラフィックスを例にしたのは、Android グラフィックスでは XML の知識は必要ないこと、視覚的にも興味のある結果が得られ学習のモチベーションがあがることです。

この章では Android アプリを作る上で当面必要な Java の基礎知識として以下の内容を説明します

■ オブジェクト指向言語特有の概念

クラス、インスタンス、コンストラクタ

■ C 言語などの基本言語と共通の概念

変数、演算子、for 文や if 文などの流れ制御構造、配列

■ Android グラフィックスに特有な概念

描画メソッド、ビットマップ

「注」 この章の例題を HVGA (320 × 480) で表示する場合は座標値やテキストサイズのピクセル値を「1/1.5」にしてください。たとえば例題 2-1 なら以下のように変更します。

「`paint.setTextSize(45);`」 → 「`paint.setTextSize(30);`」

「`canvas.drawText("Android", 30, 75, paint);`」

→ 「`canvas.drawText("Android", 20, 50, paint);`」

「注」

最初の例題のみ全リストを掲載してありますが、その後の例題は `onDraw` メソッド内のみが変更されるので、`onDraw` メソッドだけをリストとして掲載してあります。最初の例題を `Graph1` としています。その後の例題は `Graph2`, `Graph3`・・・などと別なプロジェクトを新たに作ってもよいですが、`Graph1` の `onDraw` 内だけを変更してもよいです。ただしこの場合前に作ったプログラムは無くなってしまうので注意してください。

最初の例は「Android」という文字をグラフィックスで表示します。Android グラフィックスは View クラスを元に行います。描画処理は onDraw メソッドの中に記述します。描画に当たっては、まず Paint クラスのメソッドを使って描画色やテキストサイズを指定し、次に Canvas クラスのメソッドを使ってテキストの描画を行います。

1.View クラス

グラフィックスを描画する画面は View クラスを継承して作ります。以下は GView という名前のユーザ定義クラスを作っています。コンストラクタの GView はスーパークラスのコンストラクタを呼び出す部分でこれが定型です。ユーザが行う描画処理は onDraw メソッド内に記述します。onDraw メソッドが呼び出されたときに Canvas クラスの引数 canvas に描画オブジェクトが渡されますので、この canvas に対し描画メソッドを使ってグラフィックス処理を行います。onDraw メソッドの仮引数を「Canvas canvas」としていますが、仮引数の名前はなんであってもよいので「Canvas c」などとしても構いません。

```
private class GView extends View {← View クラスを継承した  
ユーザ定義クラス Gview  
    public GView(Context context) {←コンストラクタ  
        super(context);  
    }  
    protected void onDraw(Canvas canvas) {← onDraw メソッド  
        描画内容を記述                ↑ Canvas クラスの引数 canvas  
    }  
}
```

この GView クラスを実際に画面に設定するには、「setContentView(R.layout.main);」の代わりに「setContentView(new GView(this));」とします。前者はウィジェットを配置したレイアウトを表示し、後者はグラフィックス画面を表示します。

2.Paint クラスとインスタンス

Paint クラスは描画色や文字サイズなどの描画情報を扱うクラスです。クラスからインスタンスを生成するには `new` 演算子を用いて、以下のように宣言します。コンストラクタはクラス名と同じ名前の特別なメソッドで初期化を行うものです。これで `paint` という名前の Paint クラスのインスタンスが生成されます。以後 `paint` に対しメソッドを適用します。

クラス名

コンストラクタ

```
Paint paint = new Paint();
```

インスタンス(オブジェクト)

■補足

クラスはオブジェクトを生成するための金型(テンプレート)のようなものと考えることができます。オブジェクト指向言語では、金型から実際に生成された実体をオブジェクトと言い、クラスからオブジェクトを作ることインスタンス化(instantiation)と言います。C++ではクラスを実体化したものをオブジェクトと呼び、Javaではインスタンス化したオブジェクトをインスタンスと呼びます。この場合インスタンス=オブジェクトと考えてよいです。本書では「インスタンス」という言葉が適切な場合以外は「インスタンス」と「オブジェクト」を使い分けずに「オブジェクト」と呼ぶことにします。⇒2-3の4参照。

3.Paint クラスのメソッド

描画色を青、フォントのサイズを45ピクセルに設定するには、`paint` オブジェクトに対し `setColor` メソッド、`setTextSize` メソッドを使って以下のようにします。「`Color.BLUE`」は青色を示す `Color` クラスの定数です。

```
paint.setColor(Color.BLUE); ←青色
```

```
paint.setTextSize(45); ←テキストのサイズを45ピクセル
```

4.Canvas クラスの描画メソッド

`onDraw` メソッドの引数 `canvas` に対し描画を行うには `drawText` メソッドを用います。指定する `x,y` 座標はテキストの左下隅の座標です。

描画位置のx,y座標

```
canvas.drawText("Android", 30, 75, paint);
```

Android

左下隅の座標を表示位置として指定

「例題2-1」以上をまとめた全リストを示します。それぞれの位置関係を確認してください。

```
package jp.graph1;

import android.app.Activity;
import android.os.Bundle;
import android.graphics.*;
import android.view.View;
import android.content.Context;

public class Graph1 extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new GView(this)); ← GView を画面に設定する
    }
    private class GView extends View {
        public GView(Context context) {
            super(context);
        }
        protected void onDraw(Canvas canvas) {
            Paint paint = new Paint();
            paint.setColor(Color.RED);
            paint.setTextSize(45);
            canvas.drawText("Android", 30, 75, paint);
        }
    }
}
```



「注」 メソッドの種類
onDraw と drawText
はどちらもメソッドで
すが、呼び出し方（呼
び出され方）が異なり
ます。通常 onXXX メ
ソッドはシステムから
呼び出されるメソッド
で、処理内容をユーザ
が定義します。たとえ
ば、onDraw メソッド
は画面に描画を行う必
要が生じたときに呼び
だれます。これに対し
「オブジェクト名.メ
ソッド名(引数,...)」
の形式で呼び出すメ
ソッドはユーザが定義
することなしに呼び出
します。

「注」 背景色
背景色が白になってい
る場合は P36 参照。

Chapter 03

ウィジェットとXML

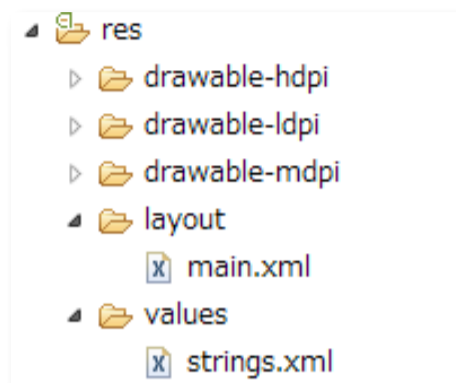
AndroidではテキストビューやボタンなどのGUI部品をウィジェット (Widget) と呼んでいます。ウィジェットは main.xml という XML ファイル中で定義します。

XML (Extensible Markup Language : 拡張可能マークアップ言語) は文書やデータの意味や構造を記述するためのマークアップ言語の一つです。Androidではレイアウトファイル (main.xml)、文字列リソースファイル (string.xml)、マニフェストファイル (AndroidManifest.xml) などがXMLで記述されています。XMLでは利用者が自由に要素や属性を定義でき、プログラムでXML要素を操作できる (具体的には findViewById メソッドで main.xml に設定したウィジェットを取得するなど) というメリットがあります。

この章ではウィジェットを記述するためのXMLの一般的な書き方と、ボタン、テキストビュー、エディットテキスト、チェックボックス、ラジオボタン、スピナー、リストビュー、イメージビューなどの個々のウィジェットの具体的な使い方を説明します。ウィジェットをクリック (タッチ) したときのアクションを監視するものをリスナーと呼びます。このリスナーをウィジェットに組み込む方法を説明します。

Chapter3-1 AndroidのXMLファイル

Androidで使用しているXMLはレイアウトファイル(main.xml)や文字列リソースファイル(string.xml)に記述されています。これらのファイルはres/layoutやres/valuesフォルダに格納されています。



■ main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <Button
        android:id="@+id/button"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="クリックしてね"
    />
</LinearLayout>
```

■ string.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, Test1!</string>
    <string name="app_name">Test1</string>
</resources>
```

1.XMLの書式

XMLでは、文書構造を構成する個々のパーツを「要素」(エレメント:Element)と呼びます。要素は要素を示すタグと内容で構成されます。タグは開始タグと終了タグがあり、その2つのタグの間に「内容」を記述します。この「内容」は定義する文字列であったり別の要素であったりします。開始タグの中にはより細かな指定を属性で行うことができます。属性にはそれぞれ値を指定します。これらの各項目は行を分けて書いても、1行にまとめて書いても同じです。

```
<開始タグ  
  属性=値  
  属性=値>  
  内容  
</終了タグ>
```

たとえばstring.xmlには以下のような要素が記述されています。

```
  開始タグ          内容  
<string name="app_name">Test1</string>  
  属性              終了タグ
```

2.XML宣言(<?xml ?>)

XML宣言はそのファイルがxmlで記述されていることをXMLパーサに指示するためのもので<?xmlで始まり?>で終わります。version属性にXMLバージョン、encoding属性に文字エンコーディングを指定します。現在のXMLバージョンは1.0です。Androidの文字エンコーディングはutf-8です。

■ UTF-8

UTF-8(Unicode Transformation Format)はUnicodeを8ビット単位の可変長コード(1~4バイト)にエンコードする方式でインターネットで広く使われているエンコード方式です。ASCIIコード対応文字はASCIIコードのまま1バイトですが、日本語は3~4バイトになります。

3.XML名前空間(xmlns)

このXMLファイルで使用する要素や属性に関する名前空間(NameSpace)が定義されている場所を指定します。この指定はルート要素にだけ置きます。

「注」
本書ではXMLの各要素のインデントはネストが深くなるごとに2文字の空白を置くことにしています。

```
xmlns:android="http://schemas.android.com/apk/res/android"
```

接頭辞

名前空間URL (定義場所)

このように指定することで、接頭辞の「android」を使って、各要素の属性を「android:id」や「android:layout_width」のように指定します。

4. レイアウト

ボタンやテキストビューなどのウィジェットを配置する方法をレイアウトと呼び、AndroidではLinearLayout,RelativeLayout,FrameLayout,TableLayoutなどがあります。デフォルトのレイアウトはLinearLayoutで、<LinearLayout>で始まり</LinearLayout>で終わります。この2つのタグの間にウィジェットを置きます。LinearLayoutでは宣言されたウィジェットの順序で単純(直線的)にウィジェットを並べます。レイアウトに指定する主な属性は以下です。

■ android:orientation

ウィジェットを並べる方法で"vertical"(垂直)、「horizontal"(水平)を指定します。指定しなければ"horizontal"とみなされます。

■ android:layout_width,android:layout_height

レイアウトの幅と高さを指定します。値には"fill_parent"(可能な限り拡大)または"wrap_content"(表示に必要なサイズ)または具体的な数値を指定します。

「注」 fill_parent と match_parent

API 8 からはfill_parentの代わりにmatch_parentを使用することが推奨されていますが、Eclipse 3.7で生成されるスケルトンでは「fill_parent」が使用されています。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical" >
  <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
```

```
</LinearLayout>
```

ただし「Graphical Layout」(3-18参照)を使用して生成するウィジェットのスケルトンでは「match_parent」が使用されています。本書では「fill_parent」を使用しています。

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
    <requestFocus />
</EditText>
```

5. ウィジェット

ウィジェットとは一般に、デスクトップの好きな位置に置いておくことのできる小さなアプリケーション(カレンダー、ノートパッド、地図、検索など)を指します。AppleやYahoo!などが「ウィジェット」と呼び、GoogleやMicrosoftは「ガジェット」と呼んでいます。widget、gadgetとも元の英語の意味は「小さな道具」という意味です。

Androidではボタンやテキストビューのような予め定義されたUI (UserInterface) 部品を標準ウィジェット(単にウィジェット)と呼んでいます。


ボタンはXMLでは次のように定義します。

```
<Button
    android:id="@+id/button"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text=" クリックしてね "
/>
```

LinearLayoutでandroid:orientation="vertical"の場合は縦に並べていくのでウィジェットのlayout_widthは"fill_parent"(横幅一杯に広げる)、layuot_heightは"wrap_content"(実際に表示される内容の大きさ)とします。

「注」直接文字列と文字列リソース

Eclipse 3.7では文字列リソースを使わずに「android:text="クリックしてね"」のように直接文字列を指定すると警告エラーとなります。警告を回避するにはstring.xmlに以下のようにリソースを定義し、「android:text="@string/msg"」とします。



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3   ↓
4   <string name="hello">Hello World, Widget1!</string>
5   <string name="app_name">Widget1</string>
6   <string name="msg">クリックしてね</string>
7 </resources>
```

6. ウィジェットのID (android:id)

ウィジェットのIDは「android:id」属性に指定します。Javaコードから findViewById メソッドを使ってウィジェットを取得する際のIDを指定します。IDの指定方法には以下の2種類があります。

① android:id="@+id/xxx"

リソースファイル(main.xmlなど)で定義しているウィジェットのユーザが定めたIDを指定します。文字列の先頭にあるアットマーク(@)は、XMLパーサに解析をさせ、id以降の文字列を展開しそれをIDリソースとして識別させるということを指示しています。プラスマーク(+)は、生成されるR.javaファイル内で、リソースのひとつとして追加される必要がある新しいリソースの名称であることを意味しています。

② android:id="@android:id/xxx"

android名前空間でシステムが予め定めているIDを指定します。ListViewなどで使用することがあります。AndroidのリソースIDを参照する場合、プラスマークは不要ですが、「android:id="@android:id/empty"」のようにandroidパッケージネームスペースを追加する必要があります。パッケージネームスペース付きの場合ローカルのリソースクラスからではなく、android.RリソースからIDを参照する必要があります。⇒中級編の「16-8 TwoLineListlem」参照

7. ウィジェットのテキスト (android:text)

「android:text」にはウィジェットに表示するテキストを指定します。文字列リソースを使う場合は「@string/リソース名」を指定します。


```
android:text="@string/hello"
```

直接文字列を指定する場合は "" 内にその文字列を指定します。

```
android:text=" クリックしてね "
```

8. 空タグ

ボタンやテキストビューなどの多くのウィジェットは開始タグと終了タグの間に内容や別のタグを持ちませんこのようなタグを空タグと呼びます。正規な書き方なら、次のように開始タグと終了タグを書きます。

```
<Button
```

```
  属性=値 >
```

```
</Button>
```

しかしこれは煩雑になるので、開始タグの最後を /> で終わることで、終了タグを省略する便法があります。

```
<Button
```

```
  属性=値 />
```

9.R.java

Java からリソースにアクセスするためのクラスが gen フォルダの R.java に自動生成されています。このファイルは自動生成されるもので、ユーザが変更を加えてはいけません。たとえば、以下のようなリソースを定義したとします。

- main.xml に配置したウィジェットの TextView(id は text) と Button(id は button)
- drawable のイメージ ic_launcher.png と white.png
- string.xml で定義した文字列リソースの "hello" と "app_name"

この場合の R.java の定義内容は以下ようになります。リソースの種別ごとに、0x7f020000 番代は drawable リソース、0x7f050000 番代は id 属性で定義されたレイアウトやウィジェットの ID などと内部管理用の ID が割り当てられます。

```
■ R.java
```

```
public final class R {
```

```
    public static final class attr {
```

```
}
public static final class drawable {
    public static final int ic_launcher=0x7f020000; ←イメージファイル
    public static final int white=0x7f020001;
}
public static final class id {
    public static final int button=0x7f050001; ←ウィジェットのID
    public static final int text=0x7f050000;
}
public static final class layout {
    public static final int main=0x7f030000; ←レイアウトファイルmain.xml
}
public static final class string {
    public static final int app_name=0x7f040001; ←文字列リソース
    public static final int hello=0x7f040000;
}
}
```

R.javaで定義される主なリソースIDクラスは以下があります。

■ layout クラス

layout リソースのxml ファイル名。0x7f030000からの通し番号が割り当てられます。

■ id クラス

id属性で定義されたレイアウトやウィジェットのID。0x7f050000からの通し番号が割り当てられます。

■ drawable クラス

drawable リソースのイメージファイル名。0x7f020000からの通し番号が割り当てられます。

■ string クラス

string.xmlに記述されている文字列リソース名。0x7f040000からの通し番号が割り当てられます。

10.R.layout.mainのロード

アプリケーションをコンパイルすると、各XMLレイアウトファイルはViewリソース内にコンパイルされます。アプリケーションコードからレイアウトをロードする必要があり、そのコードはActivity.onCreate()のコールバックメソッドで実装することになります。setContentView()にレイアウトリソースの参照を「R.layout.レイアウトファイル名」の形で引数として渡してそのメソッドを呼び出します。例えば、XMLレイアウトがmain.xmlの場合、以下のようにしてアクティビティにロードします。

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
}
```

11.findViewByIdによるウィジェットの取得

main.xmlで定義されているウィジェットをJava側から操作するためにはfindViewByIdメソッドで、ウィジェットオブジェクトを取得します。main.xmlでボタンのIDを「android:id="@+id/button」と定義していた場合、R.javaではこのIDはJava用に「R.id.button」と定義し直されていて、このIDをfindViewByIdの引数に指定します。

```
Button bt=(Button)findViewById(R.id.button);
```

■キャスト

(Button)のように()内にクラス名(ButtonやTextViewなど)や型(intやfloatなど)を書いたものをキャストと呼びます。findViewByIdで取得するオブジェクトはButtonであったり、TextViewであったりするのでキャストによりそのオブジェクトのクラスまたは型を明示する必要があります。キャストがないとエラーとなります。

Chapter 04

レイアウト

ウィジェットを配置するコンテナをレイアウトと呼びます。Androidで指定できるレイアウトとしてリニアレイアウト、相対レイアウト、フレームレイアウト、テーブルレイアウト、絶対レイアウトの5種類があります。絶対レイアウトはAndroid1.5から非推奨となりましたので、本書では扱いません。

また個々のレイアウトやウィジェットにスタイルを適用するstyles.xml、アプリケーションにスタイルを適用するthemeについても説明します。

LinearLayoutはレイアウト内のウィジェットを垂直方向または水平方向に単純にならべます。LinearLayoutクラスの属性は「3.10 LinearLayoutクラスの属性」で説明しましたのでそこを参照してください。この他にLinearLayout.LayoutParamsクラスの属性として以下があります。

LinearLayout.LayoutParamsの属性	意味
layout_gravity	ウィジェットの配置方法。gravityと同じ値を指定。
layout_weight	ウィジェットを配置する際の比率。1を基準に2なら倍、0.5なら半分

1. 垂直配置

自動生成されたmain.xmlのスケルトンでは「android:orientation="vertical"」でレイアウト内のウィジェットを垂直に配置します。orientationを指定しなければhorizontalと見なされます。

「例題4-1-1」 3つのボタンを垂直配置します。

■ main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text=" ボタン 1 "
    />
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
```

```
        android:text=" ボタン 2 "
```

```
    />
```

```
    <Button
```

```
        android:layout_width="fill_parent"
```

```
        android:layout_height="wrap_content"
```

```
        android:text=" ボタン 3 "
```

```
    />
```

```
</LinearLayout>
```



2. 水平配置

「`android:orientation="horizontal"`」を指定するか `orientation` 属性を指定しないとレイアウト内のウィジェットは水平に配置されます。

「例題4-1-2」3つのボタンを水平配置します。

```
■ main.xml
```

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
```

```
android"
```

```
    android:orientation="horizontal"
```

```
    android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent"
```

```
>
```

```
    <Button
```

```
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
```

```
        android:text=" ボタン 1 "
```

```
    />
```

```
    <Button
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text=" ボタン 2 "
/>
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text=" ボタン 3 "
/>
</LinearLayout>
```



ボタン3のlayout_widthを「fill_parent」にすると最後のボタンの横幅が画面を埋めるサイズに伸びます。最後の要素でないものに「fill_parent」を指定すると、その後のウィジェットは画面から溢れて表示されません。



3. 均等割り付け

各ウィジェットを均等割りするには、各ウィジェットの幅を「android:layout_width="fill_parent"」で画面一杯に割り当て、さらにすべてのウィジェットに「android:layout_weight="1"」で均等割りを指示します。均等割りをする場合の幅の指定を適切な値(画面に納まる範囲)にするとその幅で均等されます。画面一杯で均等割りする場合に「fill_parent」を指定します。「fill_parent」の代わりに「800dp」などと大きな値を指定したり、「0dp」などを指定する方法がありますが、TableLayoutでlayout_spanを使っている場合「0dp」はうまくいきません。

「例題4-1-3」 3つのボタンを水平に均等配置します。

■ main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <Button
        android:layout_width="fill_parent"
        android:layout_weight="1"
        android:layout_height="wrap_content"
        android:text=" ボタン 1 "
    />
    <Button
        android:layout_width="fill_parent"
        android:layout_weight="1"
        android:layout_height="wrap_content"
        android:text=" ボタン 2 "
    />
    <Button
        android:layout_width="fill_parent"
        android:layout_weight="1"
        android:layout_height="wrap_content"
        android:text=" ボタン 3 "
    />
</LinearLayout>
```



Chapter 05

main.xml を使わずに レイアウトする

レイアウトとそこに配置するウィジェットを記述する方法として主に以下のような3種類があります。すでに①の方法を説明してありますので、この章では②、③の方法を説明します。複数のウィジェットを配置する場合などは、main.xmlに記述するより、for文などを用いてJavaプログラムで記述した方が効率的な場合があります。

- ① main.xmlにレイアウトやウィジェットを記述する。
- ② main.xmlにレイアウトを記述せずに、Javaプログラムコード中でレイアウトやウィジェットを作成する。
- ③ main.xmlに記述したレイアウトに、Javaプログラムコードでウィジェットやビューを追加する。

さらに、LayoutInflaterを用いてレイアウトXMLファイルを動的にViewオブジェクトに展開する方法を説明します。レイアウトとは直接関係はありませんが、画面情報や画面設定に関する内容も説明します。

リニアレイアウトを生成し、コンテンツビューに設定するには次のようにします。

```
LinearLayout layout=new LinearLayout(this);  
layout.setOrientation(LinearLayout.HORIZONTAL);  
setContentView(layout);
```

ボタンを生成し、テキストを設定するには次のようにします。

```
Button bt1=new Button(this);  
bt1.setText("1");
```

リニアレイアウト layout にボタン bt1 を配置するには次のようにします。

```
layout.addView(bt1,new LinearLayout.LayoutParams(WC,WC));
```

addView メソッドの第二引数には追加するウィジェットのサイズを指定します。サイズは LinearLayout.LayoutParams を使って指定します。wrap_content と fill_parent を示す定数を次のように定義しておくると便利です。

```
private final int WC=ViewGroup.LayoutParams.WRAP_CONTENT;  
private final int FP=ViewGroup.LayoutParams.FILL_PARENT;
```

「例題5-1」ボタンを2つリニアレイアウトに水平方向で配置します。

■ JLayout1.java

```
package jp.jlayout1;  
  
import android.app.Activity;  
import android.os.Bundle;  
import android.view.ViewGroup;  
import android.widget.*;
```

```
public class JLayout1 extends Activity {
    private final int WC = ViewGroup.LayoutParams.WRAP_CONTENT;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        LinearLayout linearLayout=new LinearLayout(this);
        linearLayout.setOrientation(LinearLayout.HORIZONTAL);
        setContentView(linearLayout);
        Button bt1 = new Button(this);
        bt1.setText("1");
        linearLayout.addView(bt1,new LinearLayout.LayoutParams(WC,WC));
        Button bt2 = new Button(this);
        bt2.setText("2");
        linearLayout.addView(bt2,new LinearLayout.LayoutParams(WC,WC));
    }
}
```



Chapter 06

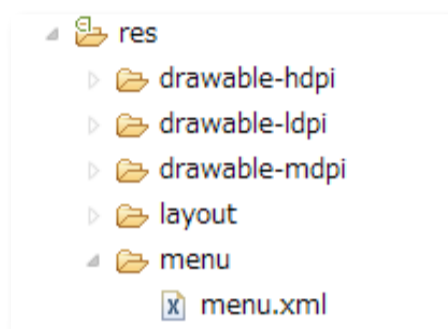
メニュー

メニューは「Menu」ボタンのクリックで、画面下部に表示されます。メニューはmenu.xmlリソースを使って作りますが、Javaコードだけで作ることもできます。メインメニューの下にサブメニューを作ることができます。サブメニューはポップアップ形式で表示されます。サブメニューにはチェックボックスなどを付けることができます。サブメニューをグループ化することで、グループ単位でチェックボックス属性やラジオボタン属性を与えることができます。ウィジェットを長押しすることで、コンテキストメニューを表示することができます。

メニュー表示を行うにはMenuInflaterクラスのオブジェクトとメソッドを使ってメニューリソースをViewオブジェクトに展開します。

1. メニューリソース (menu.xml)

表示したいメニュー・リソースをlayoutとは異なるmenuフォルダにmenu.xmlとして作成します。



メニューは<menu>タグで記述し、各メニュー項目は<item>タグで記述します。<item>タグのicon属性に表示するアイコン、title属性に表示するテキストを指定します。

```
<menu>
  <item android:id="@+id/ID"
        android:icon="@drawable/アイコンのリソース"
        android:title="テキスト"
  />
</menu>
```

2. メニューの登録

メニューの登録はonCreateOptionsMenuメソッド内で行います。まずメニューにメニュー項目を登録するためのメニューインフレーターを取得します。

```
MenuInflater inflater=getMenuInflater();
```

次にメニューリソースのmenu.xmlとonCreateOptionsMenuメソッドのmenu引数(ここにメニューオブジェクトが渡されています)を使ってメニューの登録をします。

「注」 menu.xmlの作成
「res」フォルダ上で右クリックし、「新規」→「フォルダ」から「menu」フォルダを作成します。「menu」フォルダ上で右クリックし、「新規」→「Android XML ファイル」から「menu.xml」ファイルを作成します。

```
inflater.inflate(R.menu.menu,menu);
```

3. 選択された項目の判定

メニューを選択すると `onOptionsItemSelected` メソッドが呼び出されます。このとき `item` 引数を使って選択されたメニュー項目を判定することができます。

```
switch (item.getItemId()) {  
    case R.id.項目1のID: 項目1の処理; return true;  
    case R.id.項目2のID: 項目2の処理; return true;  
    default: return super.onOptionsItemSelected(item);  
}
```

「例題6-1」 google と microsoft のメニュー項目を作ります

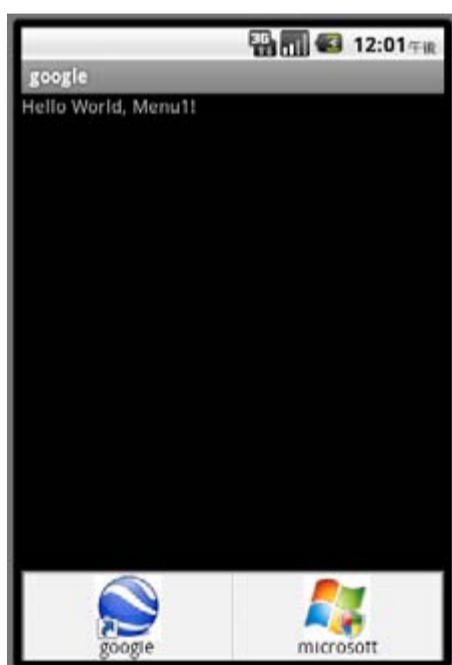
■ menu.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<menu xmlns:android="http://schemas.android.com/apk/res/android">  
    <item  
        android:id="@+id/google"  
        android:icon="@drawable/google"  
        android:title="google"  
    />  
    <item  
        android:id="@+id/microsoft"  
        android:icon="@drawable/microsoft"  
        android:title="microsoft"  
    />  
</menu>
```

■ Menu1.java

```
package jp.menu1;  
  
import android.app.Activity;  
import android.os.Bundle;  
import android.view.*;
```

```
public class Menu1 extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater mi=getMenuInflater();
        mi.inflate(R.menu.menu, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.google:setTitle("google");return true;
            case R.id.microsoft:setTitle("microsoft");return true;
            default:return super.onOptionsItemSelected(item);
        }
    }
}
```



「注」 Android 4.0ではメニュー項目は縦に配置され、アイコンが表示されません。

Chapter 07

トースト、ダイアログ、ログ

カレントのActivityとは別画面にメッセージ表示する方法としてトーストとダイアログがあります。トーストは画面下に表示され、ある時間が過ぎるとフェードアウトしていく別ウィンドウです。Androidで使用できる標準ダイアログとしてAlertDialog、ProgressDialog、DatePickerDialog、TimePickerDialogがあります。ユーザが独自のカスタム・トーストやカスタム・ダイアログを作ることができます。

トーストやダイアログはメッセージを別画面として表示するのに便利な機能ですが、デバッグ情報などの記録をとっておくのには向きません。このような情報はログとして出力します。

トーストは画面下に表示され、ある時間が過ぎるとフェードアウトしていく別ウィンドウです。トーストは以下のようにして表示します。表示されている時間は「Toast.LENGTH_LONG」と「Toast.LENGTH_SHORT」が指定できます。

```
Toast.makeText(this, "メッセージ", Toast.LENGTH_LONG).show();
```

トーストは様々な局面で、ちょっとした情報を表示するのに使われます。

「例題7-1-1」ボタンのクリックでトーストを表示します。その際、クリックした回数を変数 count にカウントして、その値をもとに「5回目」のように表示します。

■ main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <Button
        android:id="@+id/button"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="クリックしてね"
    />
</LinearLayout>
```

■ Toast1.java

```
package jp.toast1;

import android.app.Activity;
import android.os.Bundle;
import android.view.*;
import android.view.View.OnClickListener;
```

```
import android.widget.*;

public class Toast1 extends Activity implements OnClickListener{
    private int count=0;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button bt=(Button)findViewById(R.id.button);
        bt.setOnClickListener(this);
    }
    public void onClick(View view) {
        Toast.makeText(this,(++count)+" 回目 ",Toast.LENGTH_LONG).show();
    }
}
```



「補足」トーストの位置決め

標準のトーストは、画面の下部の中央に表示されます。この位置を `setGravity(Gravity 定数, x位置, y位置)` メソッドで変更することができます。Gravity 定数には TOP、BOTTOM、CENTER、LEFT、RIGHT などを指定でき、これらを「|」演算子で組み合わせて指定することもできます。「Gravity.TOP|Gravity.LEFT」とすると画面左上隅になります。x,y 位置は Gravity 定数で指定した位置からの相対オフセット値になります。

```
Toast toast=Toast.makeText(this,(++count)+" 回目 ",Toast.LENGTH_LONG);
toast.setGravity(Gravity.TOP|Gravity.LEFT,0,0);
toast.show();
```

Chapter 08

タッチイベント

タッチスクリーン(ディスプレイ)は指で触れて操作します。今までボタンやリスト項目などを指でタッチする動作をクリックと呼び、`OnClickListener`で処理していました。タッチスクリーンに指で触れる動作を一般にタッチとかタップと呼び、主な操作方法の呼び方は以下です。

呼び方	動作
タップ	指で軽く叩く操作。マウスのクリックに相当
ダブルタップ	2回叩く操作。マウスのダブルクリックに相当
ロングタッチ	一定時間(たとえば1秒以上)画面に触れてから離す操作
ドラッグ(スライド)	指で押さえながら移動する操作
スクロール	指で押さえながら上下左右に移動する操作
フリック(フリング)	リストなどをスクロールする時に指で軽くはらう操作
ピンチ	2本指でのつまむ操作の総称
ピンチアウト(ピンチオープン)	2本指の間を広げて拡大する時の操作
ピンチイン(ピンチクローズ)	2本指の間を縮めて縮小する時の操作

Chapter8-1

タッチアクションの種類

タッチイベントが発生するとonTouchEventメソッドが呼び出されます。そのとき引数eventを使ってevent.getAction()で発生したイベントの種類(ACTION_DOWN, ACTION_UP, ACTION_MOVE)が取得できます。タッチ位置の座標はevent.getX()とevent.getY()でfloat値で取得できます。

```
public boolean onTouchEvent(MotionEvent event) {
    switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN: break;
        case MotionEvent.ACTION_UP: break;
        case MotionEvent.ACTION_MOVE: break;
    }
    return super.onTouchEvent(event);
}
```

「例題8-1」タッチイベントの種類とタッチ位置のx,y座標値をタイトルバーに表示します。

```
■ Touch1.java
package jp.touch1;

import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;

public class Touch1 extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    @Override
    public boolean onTouchEvent(MotionEvent event) {
```

```
String action="";
switch (event.getAction()) {
    case MotionEvent.ACTION_DOWN:
        action="ACTION_DOWN: ";break;
    case MotionEvent.ACTION_UP:
        action="ACTION_UP: ";break;
    case MotionEvent.ACTION_MOVE:
        action="ACTION_MOVE: ";break;
}
setTitle(action+event.getX()+", "+event.getY());
return super.onTouchEvent(event);
}
}
```



Chapter 09

キーイベント、 フォーカスイベント

Androidのエミュレータ環境では通常のキーボードが付いていますが、実機のハードウェアキーは、主にBackキー、Homeキー、Menuキー、Searchキーなどに限定されます。KeyEventで定義されているKeyCodeマクロは大まかに以下の5つのキーごとに分類されています。

- 専用キー (Backキー、Homeキー、Menuキー、Searchキーなど)
- ゲームパッド用ボタン
- 通常のキーボードのキー
- 十字キー
- メディア再生用キー

これらのキー操作によるイベント処理はdispatchKeyEventイベントハンドラ、onKeyDownイベントハンドラ、onKeyUpイベントハンドラで行います。

EditTextへの入力時などに画面に表示されるキーを、ハードウェアキーの代わりにソフトキーと呼びます。ユーザ専用のソフトキーを作ることができます。

一般に、ビュー(オブジェクト)がイベントを受け取れる状態を「フォーカスを持つ」といいます。ビューのフォーカスの状態が変化したときの処理はonFocusChangeイベントハンドラで行います。

Chapter9-1

キーイベントの種類

キーイベントは `dispatchKeyEvent` イベントハンドラで捕捉することができます。引数の `event` を使って `event.getAction()` でキー操作がダウン (`KeyEvent.ACTION_DOWN`) なのかアップ (`KeyEvent.ACTION_UP`) なのか判定できます。

`event.getKeyCode()` でキー操作されたキーのコードが取得できます。キーダウン専用のイベントハンドラの `onKeyDown` やキーアップ専用のイベントハンドラの `onKeyUp` も使用できます。`onKeyDown`, `onKeyUp` では `keyCode` 引数でキー操作されたキーのコードが取得できます。いずれの場合もキーコードの値は、`KeyEvent.KEYCODE_BACK` (Back キー)、`KeyEvent.KEYCODE_MENU` (Menu キー) などの定数が定義されています。これらのイベントハンドラを複数設定した場合は、`dispatchKeyEvent` が先に呼び出され、その後 `onKeyDown`、`onKeyUp` が呼び出されます。

主な専用キーのキーコードの定義値は以下です。

```
public static final int KEYCODE_HOME    = 3;
public static final int KEYCODE_BACK    = 4;
public static final int KEYCODE_ENTER   = 66;
public static final int KEYCODE_MENU    = 82;
public static final int KEYCODE_SEARCH  = 84;
```

「英単」 `dispatch`: 急送[発送]する。複数のプログラムを同時に実行するマルチタスク OS やマルチプロセッサにおいて、処理すべきプログラム(タスク)の優先順位を決定して、CPU に割り当てること。

「例題9-1」 dispatchKeyEvent、onKeyDown、onKeyUpの呼び出し順序を調べます。Menuキーを押した場合、dispatchKeyEventのACTION_DOWN、onKeyDown、dispatchKeyEventのACTION_UP、onKeyUpの順に発生します。Backキーを押した場合、dispatchKeyEventのACTION_DOWN、onKeyDownまで発生し、プログラムを終了します。Homeキーは捕捉できません。

■ Key1.java

```
package jp.key1;

import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.widget.Toast;

public class Key1 extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        Toast.makeText(this, "KeyDown:" + keyCode, Toast.LENGTH_SHORT).
show();
        return super.onKeyDown(keyCode, event);
    }
    public boolean onKeyUp(int keyCode, KeyEvent event) {
        Toast.makeText(this, "KeyUp:" + keyCode, Toast.LENGTH_SHORT).show();
        return super.onKeyUp(keyCode, event);
    }
    public boolean dispatchKeyEvent(KeyEvent event) {
        String action="";
        switch (event.getAction()) {
            case KeyEvent.ACTION_DOWN:action="ACTION_DOWN:" +event.
getKeyCode();break;
            case KeyEvent.ACTION_UP:action="ACTION_UP:" +event.
```



```
getKeyCode());break;
    }
    Toast.makeText(this,action,Toast.LENGTH_SHORT).show();
    return super.dispatchKeyEvent(event);
}
}
```



著者略歴

河西 朝雄(かさいあさお)

山梨大学工学部電子工学科卒(1974年)。長野県岡谷工業高等学校情報技術科教諭、長野県松本工業高等学校電子工業科教諭を経て、現在は「カサイ・ソフトウェアラボ」代表。

「主な著書」

「入門ソフトウェアシリーズC言語」、「同シリーズJava言語」、「同シリーズC++」、「入門新世代言語シリーズVisualBasic4.0」、「同シリーズDelphi2.0」、「やさしいホームページの作り方シリーズHTML」、「同シリーズJavaScript」、「同シリーズHTML機能引きテクニック編」、「同シリーズホームページのすべてが分かる事典」、「同シリーズiモード対応HTMLとCGI」、「同シリーズiモード対応Javaで作るiアプリ」、「同シリーズVRML2.0」、「チュートリアル式言語入門VisualBasic.NET」、「はじめてのVisualC#. NET」、「C言語用語辞典」ほか

(以上ナツメ社)

「構造化BASIC」、「Microsoft Language シリーズMicrosoft VISUAL C++ 初級プログラミング入門上、下」、「同シリーズVisualBasic初級プログラミング入門上、下」、「C言語によるはじめてのアルゴリズム入門」、「Javaによるはじめてのアルゴリズム入門」、「VisualBasicによるはじめてのアルゴリズム入門」、「VisualBasic6.0入門編、中級テクニック編、上級編」、「Internet Language改訂新版シリーズ ホームページの制作」、「同シリーズJavaScript入門」、「同シリーズJava入門」、「New Language シリーズ標準VisualC++ プログラミングブック」、「同シリーズ標準Java プログラミングブック」、「VB.NET 基礎学習Bible」、「原理がわかるプログラムの法則」、「プログラムの最初の壁」、「河西メソッド：C言語プログラム学習の方程式」、「基礎から学べる VisualBasic2005標準コースウェア」、「基礎から学べる JavaScript 標準コースウェア」、「基礎から学べる C言語標準コースウェア」、「基礎から学べる PHP 標準コースウェア」、「なぞりがきC言語学習ドリル」、「C言語 標準ライブラリ関数ポケットリファレンス [ANSI C,ISO C99対応]」、「C言語 標準文法ポケットリファレンス [ANSI C,ISO C99対応]」、「[[標準] C言語重要用語解説 ANSI C / ISO C99対応」ほか

(以上技術評論社)

Androidプログラミング Bible

初級 基礎編

Android 2.x / 4.x対応

2013年1月15日 初版 第1刷発行

著者＝河西 朝雄

発行者＝河西 朝雄

発行所＝カサイ. ソフトウエアラボ

長野県茅野市ちの813 TEL.0266-72-4778

デザイン＝河西 朝樹

本書の一部または全部を著作権法の定める範囲を超え、無断で複製、複製、転載、あるいはファイルに落とすことを禁じます。

本書に記載された内容は、情報の提供のみを目的としています。したがって、本書を用いた運用は、必ずお客様自身の責任と判断によって行ってください。これらの情報の運用の結果について、発行者および著者はいかなる責任も負いません。

定価＝1,000円(税込)

©2013 河西 朝雄