Android 2.x / 4.x 対応

Android

プログラミングBible

中級 Android 的プログラミング法

河西 朝雄 著

KASAI.SOFTWARELAB

定価 1,500 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 HandsetAlliance」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 Google Play Store(旧名称は Android Market)が提供されていて、2013 年 7 月時点で有料、無料含め 100 万を超えるアプリケーションが提供されています。Google Play Store 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「中級　Android 的プログラミング法」です。34 の章というのはかなり多い章分けですが、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」はグラフィックスやウイジェットを例に Java や XML の一般的なプログラミング法を説明しました。「中級　Android 的プログラミング法」は Android の特徴を生かしたプログラミング法を説明します。Andoroid アプリケーションを構成するコンポーネントとして、アクティビティ、サービス、ブロードキャストレシーバーがあります。これら 3 種類のコンポーネントを起動するのにインテントを使います。これらについて 10 章～14 章で説明します。アプリケーションのコードが実行される前に、Android システムにアプリケーションに関する必要不可欠な情報を伝える必要があります。これらの情報を Java コードではなくマニフェスト（AndroidManifest.xml）に記述することでコードが実行される前に Android システムに情報を伝えることができます。すでに個々のマニフェストの使用例は示していますが、15 章で系統的に説明します。基本ウイジェットについては「初級　基礎編」で説明しましたが、基本ウイジェットを機能強化したウイジェット、小物ウイジェット、高度なビュー系ウイジェットについて 16 章～18 章で説明します。アプリケーション・ウイジェットとはホーム画面に直接貼り付けて利用できる小規模アプリケーションで 19 章で説明します。マルチメデイアを扱うウイジェットとして MediaPlayer と VideoView があり 20 章で説明します。リソースをイメージや文字列としてアプリケーションから常に外部化しておくことにより、それらを独立させて保持することが可能となります。21 章では文字列リソース、カラー状態リストリソース、Drawable リソース、メニューリソース、レイアウトリソース、スタイルリソース、その他のリソースについて説明します。アニメーションリソースについては 22 章で説明します。

ということで本書は以下のような章構成となっています。

10 章　インテントとアクティビティ
11 章　Thread、Handler、Message
12 章　サービス
13 章　ブロードキャストレシーバ
14 章　コンテンツプロバイダ
15 章　マニフェスト
16 章　基本ウイジェットを機能強化したウイジェット
17 章　小物ウイジェット

18 章　高度なビュー系ウイジェット
19 章　アプリケーション・ウイジェット
20 章　マルチメデイア
21 章　リソース
22 章　アニメーション

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。

2014 年 2 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「中級　Android 的プログラミング法」です。他の本の内容は以下です。

☆初級　基礎編

1 章　Java による Android アプリの作り方
2 章　Android グラフィックスによる Java 入門
3 章　ウイジェットと XML
4 章　レイアウト
5 章　main.xml を使わずにレイアウトする
6 章　メニュー
7 章　トースト、ダイアログ、ログ
8 章　タッチイベント
9 章　キーイベント、フォーカスイベント

☆上級　各種処理

23 章　グラフィックス
24 章　SurfaceView
25 章　OpenGL
26 章　ファイル処理
27 章　SQLite
28 章　Gmail
29 章　GoogleMap
30 章　センサー（実機のみ）
31 章　カメラ（実機のみ）
32 章　音声認識（実機のみ）
33 章　音声合成
34 章　ネットワーク通信

目次（中級　Android 的プログラミング法）

10 章　インテントとアクティビティ

Andoroid アプリケーションを構成するコンポーネントとして、アクティビティ、サービス、ブロードキャストレシーバーがあります。これら 3 種類のコンポーネントを起動するのにインテントを使います。この章ではインテントを使ってアクティビティを起動する方法や、各種サービスを起動する方法を説明します。サービス、ブロードキャストレシーバーに関してはそれぞれ別の章で説明します。

10-1　インテントとは

Android では Intent クラスを使って別の Activity に状態を移すことができます。インテントには動作とデータを与えます。動作（Action）にユーザ定義クラスを指定する場合を「明示的インテント」と呼びます。Android が定める Action の種類を指定する場合を「暗黙的インテント」と呼び、以下のような Action があります。

Action	機能
ACTION_VIEW	別画面（別アクティビティ）を表示します。指定する URI が「http:」なら Web ページ、「geo:」なら GoogleMap などになります。最も一般的なアクションです。インテントを使って GoogleMap の表示する方法は 29 章の 29-2 で説明します。
ACTION_SEND	データを送ります。メールの送信などに使います。
ACTION_DIAL	ダイアラーを表示します。
ACTION_SET_WALLPAPER	壁紙の設定をします。
ACTION_PICK	データから項目を選択します。ギャラリーなどで使用します。

「英単」intent：意図，意向,目的、しっかりと向けられている

1.　ACTION_VIEW

たとえば指定した URI の Web ページを表示するには次のようにします。インテントのアクションは「ACTION_VIEW」、データは「Uri.parse("http://www.google.co.jp/")」となります。このインテントを開始するには startActivity メソッドを使います。

```
Uri uri=Uri.parse("http://www.google.co.jp/");
Intent it=new Intent(Intent.ACTION_VIEW,uri);
startActivity(it);
```

「例題 10-1」指定した URI の Web ページを表示します。

・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="wrap_content"
        android:layout_height="wrap_content"
        android:text="他のページへ"
    />
</LinearLayout>
```

・Intent1.java

```
package jp.Intent1;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class Intent1 extends Activity implements OnClickListener {
    @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) {
        Uri uri=Uri.parse("http://www.google.co.jp/");
        Intent it=new Intent(Intent.ACTION_VIEW,uri);
        startActivity(it);
    }
}
```

「注」「Uri.parse("http://www.google.co.jp/search?hl=ja&source=hp&q=AKB48");」などとすれば指定したキーで検索します。

2.　ACTION_SET_WALLPAPER（壁紙の設定）

　壁紙の設定をします。インテントのアクションは「ACTION_SET_WALLPAPER」、データはありません。

```
Intent it=new android.content.Intent(Intent.ACTION_SET_WALLPAPER);
startActivity(it);
```

「補足」壁紙の取得と設定

インテントを使わずに setWallpaper メソッドで壁紙を設定することができます。また getWallpaper メソッドで現在設定されている壁紙を取得することができます。以下はロード時に現在の壁紙のビットマップをキャンバスに表示します。画面タッチで別のイメージを壁紙に設定し、キャンバスにも設定した壁紙を表示します。

・マニフェスト（AndroidManifest.xml）

```
<uses-permission android:name="android.permission.SET_WALLPAPER"/>
```

・WallP.java

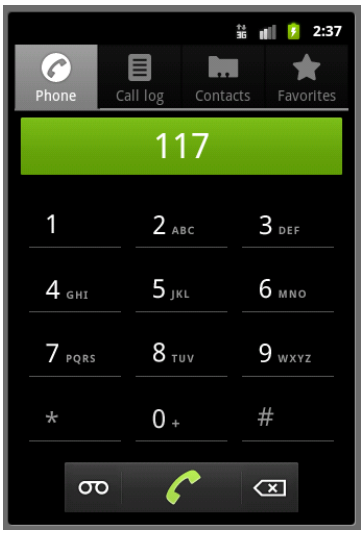
```
package jp.wallp;

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

public class WallP extends Activity {
	private GView gv;
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		gv=new GView(this);
		setContentView(gv);
	}
	public boolean onTouchEvent(MotionEvent event) {
		if (event.getAction()==MotionEvent.ACTION_DOWN){
			Bitmap
bmp=BitmapFactory.decodeResource(getResources(),R.drawable.sakura);
			try {
				setWallpaper(bmp);
				gv.invalidate();
			} catch (Exception e) {}
		}
```

```
            return super.onTouchEvent(event);
        }
        private class GView extends View {
            public GView(Context context) {
                super(context);
            }
            protected void onDraw(Canvas canvas) {
                BitmapDrawable wallpaper=(BitmapDrawable)getWallpaper();
                Bitmap bmp=wallpaper.getBitmap();
                canvas.drawBitmap(bmp,0,0,null);
            }
        }
}
```

3. ACTION_DIAL（ダイアラーの表示）

ダイアラーを表示します。インテントのアクションは「ACTION_DIAL」、データは「Uri.parse("tel:117")」となります。

```
Intent it=new Intent(Intent.ACTION_DIAL,Uri.parse("tel:117"));
startActivity(it);
```



「補足」リストビューの名前を選択してダイアラーを起動します。

・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"
	>
	<ListView
		android:id="@+id/listview"
		android:layout_width="fill_parent"
		android:layout_height="wrap_content"
	/>
</LinearLayout>
```

・Intent2.java

```
package jp.intent2;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
```

```
import android.widget.*;
import android.widget.AdapterView.OnItemClickListener;

public class Intent2 extends Activity {
    private String name[]={"あっちゃん","まゆゆ","たかみな"};
    private String tel[]={"0312341234","0412341234","0512341234"};
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
          android.R.layout.simple_list_item_1);
        ListView lv=(ListView) findViewById(R.id.listview);
        for (int i=0;i<name.length;i++)
            adapter.add(name[i]);
        lv.setAdapter(adapter);
        lv.setOnItemClickListener(new ItemClick());
    }
    class ItemClick implements OnItemClickListener {
        public void onItemClick(AdapterView<?> parent,View view,int position, long
id) {
            Intent it=new Intent(Intent.ACTION_DIAL,
            Uri.parse("tel:"+tel[position]));
            startActivity(it);
        }
    }
}
```


5:15午前
電話
通話履歴
連絡先
お気入り
03-1234-1234
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
*
0 +
#

11 章　Thread、Handler、Message

Java ではプログラムの一連の実行をスレッド（Thread）という小さい処理単位に分割し、必要に応じて実行することで複数のプログラムを同時に並列処理しているように見せることができます。いままでのプログラムでは、ボタンのクリックや画面のタッチなどのイベントの発生を受けて処理（イベントドリブン・プログラム）を行っていましたが、そのようなイベントと関係なく別の仕事を行いたい場合にスレッドを用います。

Android でもスレッドを使用することができます。ただし、Android の GUI はシングルスレッドにしか対応していないため、画面の描画処理を別スレッドで実行することを禁止しています。このような場合はハンドラ（Handler）を介することで裏ルートでのマルチスレッドを実現します。Message はハンドラに渡すメッセージを定義するクラスです。

Handler を使って一定時間ごとの処理を行うことができます。このことを利用してイメージを一定時間ごとに移動する方法を説明します。

11-1　Thread と Handler

Thread を用いて別スレッドを起動する方法と、画面描画処理を Handler に投げる方法を説明します。

1.　Thread クラス

スレッドを用いた処理はThreadクラスとRunnableインターフェースを用いて行います。スレッドは start メソッドで開始します。スレッドが実行されたときに行う処理は run メソッド内に記述します。run メソッドは Runnable インターフェースのメソッドです。Thread を生成し、開始するには以下のようにします。start メソッドが実行されると、run メソッドが呼び出されます。

```
new Thread(new Runnable() {
    public void run() {
        //スレッド内で行う処理内容
    }
}).start();
```

ただし、Android の GUI はシングルスレッドにしか対応していないため、画面の描画処理を別スレッドで実行することを禁止しています。つまりアクティビティと異なるスレッドからアクティビティ画面に描画しようとすると実行時エラーとなります。つまり、上の「スレッド内で行う処理内容」にアクティビティに対する描画処理を置くとエラーとなります。たとえば以下のようにタイトルバーに表示しようとするとエラーとなります。

別スレッドで画面に描画するには SurfaceView を用います。24 章の「24-4」を参照してください。

・Thread1.java

```
package jp.thread1;

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

public class Thread1 extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction()==MotionEvent.ACTION_DOWN){
            new Thread(new Runnable() {
                public void run() {
                    setTitle("別スレッドから GUI にアクセス");
                }
            }).start();
        }
        return super.onTouchEvent(event);
    }
}
```

「補足」スレッドは以下のような形式で作成し、start することもできます。

```
public class Thread1 extends Activity implements Runnable{
    private Thread th=null;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction()==MotionEvent.ACTION_DOWN){
            if (th==null){
                th=new Thread(this);
                th.start();
```

```
            }
        }
        return super.onTouchEvent(event);
    }
    public void run() {
        // 処理
    }
}
```

run メソッド内で一定時間ごとの処理をしたい場合は以下のようにします。スレッドをスリープするには通常の Java のように「th.sleep(ミリ秒)」のようにすることはできず、静的オブジェクトを使って「Thread.sleep(ミリ秒)」とします。「th.sleep(ミリ秒)」を使う場合は run メソッドの前に「@SuppressWarnings("static-access")」を置きます。

```
public void run() {
    while (true) {
        try {
            Thread.sleep(ミリ秒);
        }
        catch (Exception e){}
        // 処理
    }
}
```

2. Handler クラス

このようなエラーを回避するにはアクティビティ内でハンドラ hd を生成しておき、この hd に対し post メソッドを使って Runable オブジェクトを設定します。これによりこの Runable スレッドは、アクティビティと同じスレッドで実行されることになりエラーを回避できます。このような別スレッドからアクティビティ画面を描画する例は「12-5　コールバック」を参照してください。

```
private Handler hd;
hd=new Handler();

hd.post(new Runnable() {
    public void run() {
```

```
            //アクティビティへの描画処理
        }
});
```

Handler クラスのメソッドとして以下があります。

Handler クラスのメソッド	機能
post	メッセージキューに Runnable を追加します。
postDelayed	post と同様ですが指定時間が経過後に実行されるように、メッセージキューに追加します。
sendMessage	メッセージキューにメッセージを追加します。このメッセージは、このハンドラに結び付けられたスレッドの handleMessage で受信されます。
sendMessageDelayed	sendMessage と同様ですが、指定時間が経過後に実行されるように、メッセージキューに追加します。
sendEmptyMessage	sendMessage と同様ですが、what 値だけを含むメッセージを送ります。 「hd.sendMessage(Message.obtain(hd,1));」は 「hd.sendEmptyMessage(1);」と同じです。

「例題 11-1」画面のタッチで、別スレッドを開始し、その中でタイトルバーへの表示を行います。

・Handler1.java

```
package jp.handler1;

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

public class Handler1 extends Activity {
    private Handler hd;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
```

```
        hd=new Handler();
    }
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction()==MotionEvent.ACTION_DOWN){
            new Thread(new Runnable() {
                public void run() {
                    hd.post(new Runnable() {
                        public void run() {
                            setTitle("別スレッドから GUI にアクセス");
                        }
                    });
                }
            }).start();
        }
        return super.onTouchEvent(event);
    }
}
```

12 章　サービス

アクティビティは画面に表示されている表に現れる処理です。これに対してサービスやブロードキャストレシーバはアクティビティのバックグラウンドで動作する表に現れない処理です。サービスは長時間かかる処理をアクティビティと独立して行う仕組みです。また、サービスを呼び出したアクティビティが中断されてもサービス処理を継続することができます。

サービスを起動させる方法は、「インテントを利用する方法」と「インターフェースを用いたバインドを利用する方法」の 2 種類の方法があります。前者は単にサービスを起動するような場合に簡便な方法です。後者はサービスを利用するためにインターフェースを提供しなければならないなど、前者に比べて処理が複雑になりますが、アクティビティとサービスの間でより柔軟な連携が行えます。また後者ではコールバック機能を使ってサービスで発生したイベントをアクティビティに通知することもできます。

ユーザが作成するサービスとは別にシステムが予め提供するサービスがあり、このサービスを getSystemService で取得して、ユーザが利用することができます。

12-1　インテントによるサービスの起動

インテントによるサービスの起動方法を以下に説明します。

1.　Service クラス

サービスは Service クラスを継承して作ります。onStart メソッドはサービス開始時に呼び出され、ここにサービスの内容を記述します。onBind メソッドはバインドを行う際に呼び出されますが、ここではバインドを行わないので null を返すだけの実装とします。

```
public class MyService extends Service {
    @Override
    public void onStart(Intent intent, int startId) {
        サービス内容
    }
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
}
```

このサービスは、マニフェストに以下のように指定します。

```
<service android:name="MyService">
</service>
```

2.　サービスの開始と終了

このサービスを開始するにはインテントと startService メソッドを使って以下のようにします。

```
Intent it = new Intent(this,MyService.class);
startService(it);
```

このサービスを終了するにはインテントと stopService メソッドを使って以下のようにします。

```
Intent it=new Intent(this,MyService.class);
stopService(it);
```

「例題 12-1」画面のタッチでサービスを呼び出します。アクティビティの終了でサービスを終了します。サービスはサービスの開始と終了を Toast メッセージで表示するだけのものです。アクティビティを Service1.java、サービスを MyService.java とします。

・マニフェスト（AndroidManifest.xml）
下線部を追加。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
・
・
    <application android:icon="@drawable/ic_launcher"
android:label="@string/app_name">
        <activity android:name=".Service1"
・
・
        </activity>
        <service android:name="MyService">
        </service>
    </application>
</manifest>
```

・Service1.java

```
package jp.service1;

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

public class Service1 extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    public void onDestroy() {
```

```
        super.onDestroy();
        Intent it=new Intent(this,MyService.class);
        stopService(it);
    }
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction()==MotionEvent.ACTION_DOWN){
            Intent it=new Intent(this,MyService.class);
            startService(it);
        }
        return super.onTouchEvent(event);
    }
}
```

・MyService.java

```
package jp.service1;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.widget.Toast;

public class MyService extends Service{
    @Override
    public void onStart(Intent intent, int startId) {
        Toast.makeText(this,"サービス開始",Toast.LENGTH_SHORT).show();
    }
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
    public void onDestroy() {
        Toast.makeText(this,"サービス終了",Toast.LENGTH_SHORT).show();
    }
}
```


5:58午前
Service1
Hello World, Service1!
サービス開始


5:58午前
Google
ホーム画面へのアプリのドラッグ
ランチャーでアプリが振動するまで押し続けます。
2 / 6
サービス終了

13 章　ブロードキャストレシーバ

時刻変更が行われた、バッテリーの状態に変化が生じたなどの、システム全体に影響を与える事項が発生した場合は稼働中のアプリケーション全てに対して通知が必要となります。このような場合にブロードキャストで通知を行います。broadcast とは不特定多数のすべてのユーザーに対してデータを送信することです。ブロードキャストを受信するには BroadcastReceiver クラスの onReceive メソッドで行います。

13-1　ブロードキャストの送信と受信

ユーザがブロードキャストを送信するにはマニフェストに記述したブロードキャストを行うためのインテントアクション名を指定したインテントを生成し、putExtra で送信データを設定し、sendBroadcast で送信します。

```
Intent it=new Intent("アクション名");
it.putExtra("データ識別文字列","データ");
sendBroadcast(it);
```

ブロードキャストされたデータを受信するには BroadcastReceiver クラスの onReceive メソッドで行います。受信したデータは送信時に指定した"データ識別文字列"使って取得します。

```
public void onReceive(Context context, Intent it) {
    Bundle bd=it.getExtras();
    String txt=bd.getString("データ識別文字列");
}
```

「例題 13-1」画面のタッチでブロードキャスト送信を行い、ブロードキャストレシーバーで受信したデータを表示します。ブロードキャストを行うためのインテントアクション名を「BroadCast」とし、ブロードキャストを受信するクラスを TextReceiver とします。

・マニフェスト（AndroidManifest.xml）

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
・
・
    <application android:icon="@drawable/ic_launcher"
android:label="@string/app_name">
        <activity android:name=".BCast1"
・
・
        </activity>
        <receiver android:name=".TextReceiver">
            <intent-filter>
```

```
                <action android:name="BroadCast"/>
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

・BCast1.java

```
package jp.bcast1;

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

public class BCast1 extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction()==MotionEvent.ACTION_DOWN){
            Intent it=new Intent("BroadCast");
            it.putExtra("TEXT","テスト送信");
            sendBroadcast(it);
        }
        return super.onTouchEvent(event);
    }
}
```

・TextReceiver.java

```
package jp.bcast1;

import android.content.*;
import android.os.Bundle;
import android.widget.Toast;
```

```
public class TextReceiver extends BroadcastReceiver{
	@Override
	public void onReceive(Context context, Intent it) {
		Bundle bd=it.getExtras();
		String txt=bd.getString("TEXT");
		Toast.makeText(context,txt, Toast.LENGTH_SHORT).show();
	}
}
```

14 章　コンテンツプロバイダ

Android では標準のコンテンツプロバイダを android.provider パッケージで公開し、アプリケーションから利用できるようになっています。代表的なものに、オーディオ、ビデオ、イメージデータ、発着信履歴（CallLog）、コンタクトリスト（Contacts）などがあります。10 章では ACTION_PICK アクションを使って、ギャラリー、ビデオ・コンテンツ、オーディオ・コンテンツ、電話帳などの一覧を取得・表示する方法を説明しました。ACTION_PICK アクションもこのコンテンツプロバイダを利用してデータを取得・表示しています。ここでは、ACTION_PICK アクションからでなく直接コンテンツプロバイダに接続してデータを取得する方法を説明します。

また、独自のコンテンツプロバイダを作り、このコンテンツプロバイダ経由でファイルにアクセスすることもできます。これについては「26 章　ファイル処理」で説明します。

14-1　コンテンツプロバイダへのアクセス法

コンテンツプロバイダへのアクセスは ContentResolver オブジェクトを使います。

1.　ContentResolver オブジェクト

android.provider パッケージで公開されているコンテンツプロバイダにアクセスするには getContentResolver()で ContentResolver オブジェクトを取得します。このオブジェクトに対し query メソッドを適用し、指定したコンテンツプロバイダ URI が示すデータベースへのカーソルを取得します。query の引数は 5 つあり、先頭の URI 以外は null が指定できます。null の場合はデフォルト処理が行われます。

```
Cursor c=getContentResolver().query(
    コンテンツプロバイダの URI,
    返すべきカラムのデータ名,
    返す行を選別するフィルタ,
    クエリパラメータ,
    返却された行のソート順);
startManagingCursor(c);
```

2.　ブラウザの履歴

ブラウザの履歴は android.provider.Browser クラスで取得します。プロバイダへの URI として android.provider. Browser.BOOKMARKS_URI と android.provider. Browser. SEARCHES_URI があります。これらの URI を指定して取得したデータはそれぞれ以下のサブクラスを使ってアクセスします。

Browser のサブクラス	意味
Browser.BookmarkColumns	ブラウザのブックマークと履歴。
Browser.SearchColumns	ブラウザの検索履歴。

Browser.BookmarkColumns の定数として以下があります。

Browser.BookmarkColumns の定数	意味
DATE	最後に訪問した日付。
TITLE	履歴のタイトル。
URL	履歴の URI。

「例題 14-1」ブラウザの履歴をリストビューに表示します。

・マニフェスト（AndroidManifest.xml）

```
<uses-permission
android:name="com.android.browser.permission.READ_HISTORY_BOOKMARKS" />
```

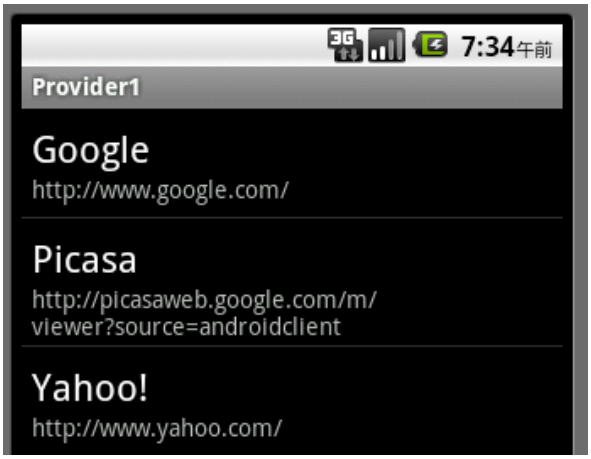
・main.xml

```
<?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"
    >
    <ListView
        android:id="@+id/listview"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    />
</LinearLayout>
```

・Provider1.java

```
package jp.provider1;

import android.app.Activity;
import android.database.Cursor;
import android.os.Bundle;
import android.provider.Browser;
import android.widget.*;

public class Provider1 extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Cursor c=getContentResolver().query(Browser.BOOKMARKS_URI,
null,null,null,null);
```

```
        startManagingCursor(c);
        ListAdapter adapter=new
SimpleCursorAdapter(this,android.R.layout.simple_list_item_2,c,
            new String[] {Browser.BookmarkColumns.TITLE,
                        Browser.BookmarkColumns.URL},
            new int[]{android.R.id.text1,android.R.id.text2});
        ListView lv=(ListView)findViewById(R.id.listview);
        lv.setAdapter(adapter);
    }
}
```



15 章　マニフェスト

アプリケーションのコードが実行される前に、Android システムにアプリケーションに関する必要不可欠な情報を伝える必要があります。これらの情報を Java コードではなくマニフェスト（AndroidManifest.xml）に記述することでコードが実行される前に Android システムに情報を伝えることができます。10 章~14 章でインテント、サービス、ブロードキャストレシーバ、コンテンツプロバイダを使用する上でのマニフェストの記述方法について個々に説明しました。この章ではマニフェストについて系統的にまとめて解説します。

15-1　マニフェストの主な役割

以下のようなプロジェクトを作成したとします。

この場合に、デフォルトで生成されるマニフェスト（AndroidManifest.xml）は以下のような内容です。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="jp.manif1"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="8" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".Manif1"
            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>
```

　マニフェストの主な役割は以下です。
·アプリケーションの Java パッケージ名を指定します。パッケージ名がアプリケーションを一意に識別する名前を提供します。

·アプリケーションを構成するコンポーネントが何かを記述します。各コンポーネントのクラスの名前およびその機能(例えばIntentメッセージをハンドリングできるコンポーネントの選択)を公開します。これらの宣言により、Android はどんなコンポーネントがあるのか、どういった条件でそれらが起動されるのかということを知ることができます。主なコンポーネントとして以下があります。
<activity>　アクティビティ
<service>　サービス
<receiver>　ブロードキャストレシーバ
<provider>　コンテンツプロバイダ

·ホストとなるアプリケーションのコンポーネントを決定します。

·保護された API 部品にアクセスするため、および他のアプリケーションと相互作用するために、アプリケーションに付与すべき許可の選択を宣言します。

·同じように、このアプリケーションのコンポーネントと相互作用するために、外部に付与されるべき許可を宣言します。

·アプリケーション実行時の分析およびその他の情報を提供する Instrumentation クラスを列挙します。この宣言は、アプリケーションの開発時やテスト時のみマニフェストに存在します。つまりアプリケーションの公開前に削除されます。

·アプリケーションが必要とする Android API の最低限のレベルを宣言します。

·アプリケーションがリンクする必要があるライブラリをリストします。

「英単」launcher：ランチャー
　ロケットやミサイルなどの発射機・発射装置。コンピューターで頻繁に利用するアプリケーションソフトを、少ない手順で起動するための機能。

16 章　基本ウイジェットを機能強化したウイジェット

3 章では Button、CheckBox、EditText、ImageView、ListView、RadioButton、RadioGroup、Spinner、TextView といった基本ウイジェットについて説明しました。これらの基本ウイジェットを機能強化したウイジェットがあります。オートコンプリート機能を持つテキストビュー、マルチオートコンプリート機能を持つテキストビュー、チェック付きテキストビュー、拡張機能付きリストビュー、イメージボタン、トグルボタン、リストビューに 2 項目のテキストビューを展開するリストアイテムなどがあります。

16-1　ウイジェットの種類

すでに 3 章で基本ウイジェットについて説明しました。ウイジェットは「android.widget」パッケージで定義されています。ListView や Spinner にデータを提供する Adapter、4 章で説明したレイアウト、7 章で説明した Toast も「android.widget」パッケージのクラスです。ウイジェットを機能別に分類すると、レイアウト、基本ウイジェット、基本ウイジェットを機能強化したウイジェット、小物ウイジェット、高度なビュー系ウイジェット、マルチメディア系ウイジェットになります。

1.　レイアウト

クラス名	意味	解説箇所
AbsoluteLayout	絶対レイアウト	Android1.5 から非推奨
AbsoluteLayout.LayoutParams	レイアウトパラメータ	Android1.5 から非推奨
FrameLayout	フレームレイアウト	4-4
FrameLayout.LayoutParams	レイアウトパラメータ	4-4
LinearLayout	リニアレイアウト	4-1
LinearLayout.LayoutParams	レイアウトパラメータ	4-1
RelativeLayout	相対レイアウト	4-3
RelativeLayout LayoutParam	レイアウトパラメータ	4-3
TableLayout	テーブルレイアウト	4-5
TableLayout.LayoutParams	レイアウトパラメータ	4-5
TableRow	テーブル要素	4-5
TableRow.LayoutParams	レイアウトパラメータ	4-5

2.　アダプター

クラス名	意味	解説箇所
ArrayAdapter	配列アダプタ	3-14
AdapterView	Adapter に設定されたデータ（文字列やイメージ）を View に表示。	3-17
SimpleAdapter	Map インターフェースを持つ List インターフェースに格納されたデータを layout 上に展開。	3-17
SimpleCursorAdapter	Cursor データを layout 上に展開。	3-17

3. 基本ウイジェット

クラス名	意味	解説箇所
Button	ボタン	3-2
CheckBox	チェックボックス	3-4
EditText	エディットテキスト	3-3、3-12
ImageView	イメージビュー	3-8、3-13
ListView	リストビュー	3-7、3-14
RadioButton	ラジオボタン	3-5
RadioGroup	ラジオボタングループ	3-5
Spinner	スピナー	3-6、3-14
TextView	テキストビュー	3-11

4. 基本ウイジェットを機能強化したウイジェット

クラス名	意味	解説箇所
AutoCompleteTextView	オートコンプリート機能を持つテキストビュー	16-4
CheckedTextView	チェック付きテキストビュー	16-6
ExpandableListView	拡張機能付きリストビュー	16-7
ImageButton	イメージボタン	16-2
MultiAutoCompleteTextView	マルチオートコンプリート機能を持つテキストビュー	16-5
ToggleButton	トグルボタン	16-3
TwoLineListItem	リストビューに2項目のテキストビューを展開するリストアイテム	16-8

5. 小物ウイジェット

クラス名	意味	解説箇所
AnalogClock	アナログクロック	17-1
Chronometer	クロノメータ	17-4
DatePicker	日付ピッカー	17-2
DigitalClock	ディジタルクロック	17-1
PopupWindow	ポップアップウインドウ	17-12

ProgressBar	プログレスバー	17-5
QuickContactBadge	連絡先データベースへのクイックコンタクト	17-13
RaitingBar	レィティングバー	17-6
SeekBar	シークバー	17-7
SlidingDrawer	ビューの展開と圧縮を行うプルタブ	17-11
TimePicker	時刻ピッカー	17-3
Toast	トースト	7-1
ZoomButton	ズームボタン	17-8
ZoomButtonsControl	ズームボタンコントロール	17-9
ZoomControls	ズームコントロール	17-10

6. 高度なビュー系ウイジェット

クラス名	意味	解説箇所
Gallery	ギャラリー	18-1
GridView	グリッドビュー	18-2
HorizontalScrollView	水平スクロールビュー	18-3
ImageSwitcher	イメージスイッチャー	18-5
OverScroller	オーバースクローラー	18-11
RemoteViews	リモートビュー	19-2
Scroller	スクローラー	18-10
ScrollView	スクロールビュー	18-4
TabWidget	タブウイジェット	18-12
TextSwitcer	テキストスイッチャー	18-6
ViewAnimator	ビューアニメーター	18-7
ViewFlipper	ビューフリッパー	18-8
ViewSwitcher	ビュースイッチャー	18-9

7. マルチメディア系ウイジェット

クラス名	意味	解説箇所
MediaPlayer	メディアプレーヤー	20-1
VideoView	ビデオビュー	10-10、20-7

「注」AbsListView、AbsSpinner は ExpandableListView、Gallery、GridView、ListView、Spinner などのベースクラスで直接使用することはないので本書では説明しません。

「注」CompoundButton は CheckBox、RadioButton、Switch、ToggleButton などのベースクラスで直接使用することはないので本書では説明しません。

「注」Adapter として AlphabetIndexer、CursorTreeAdapter、Filter、FilterQueryProvider、HeaderViewListAdapter、HeterogeneousExpandableList、ResourceCursorAdapter、ResourceCursorTreeAdapter、SimpleCursorTreeAdapter、SimpleExpandableListAdapter などがありますが本書では説明しません。

「注」API 11 以後では CalendarView、EdgeEffect(API 14)、ListPopupWindow、NumberPicker、PopupMenu、RemoteViewsService、SearchView、ShareActionProvider(API 14)、Space(API 14)、StackView、Switch(API 14)などのウイジェットがありますが本書では説明しません。

17章　小物ウイジェット

基本ウイジェットほどの使用頻度はありませんが、ちょっとした使い勝手のよい小物ウイジェットがあります。

時間や時計を扱うウイジェットとしてアナログクロック、ディジタルクロック、日付ピッカー、時刻ピッカーがあります。

進捗状況などを表示したり設定するウイジェットとしてクロノメータ、プログレスバー、レィティングバー、シークバーがあります。

マップなどのイメージにズームをかけるウイジェットとしてズームボタン、ズームボタンコントロール、ズームコントロールがあります。

その他にポップアップウインドウ、連絡先データベースへのクイックコンタクト、プルタブビューの展開と圧縮を行うプルタブがあります。

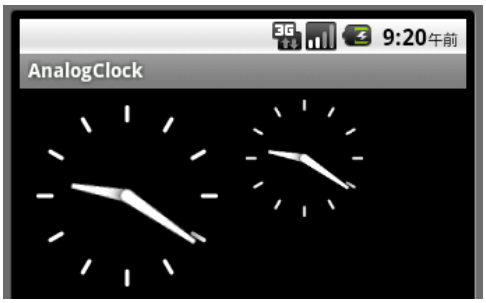
,

17-1　AnalogClock/DigitalClock

AnalogClock はシステム時間をアナログ形式の時計で表示します。表示するだけで時間の設定などは行えません。

・main.xml

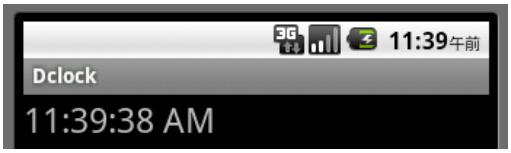
```
<AnalogClock
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
/>
<AnalogClock
    android:layout_width="100dp"
    android:layout_height="100dp"
/>
```



DigitalClock はシステム時間をデジタル形式の時計で表示します。表示するだけで時間の設定などは行えません。

・main.xml

```
<DigitalClock
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="24sp"
/>
```



18章　高度なビュー系ウイジェット

ビューを各種レイアウトで表示するための高度なビュー系ウイジェットがあります。

複数のアイテムを表示し、スクロールすることができるウイジェットとしてギャラリー、グリッドビューがあります。

ビューのスクロールを行うウイジェットとして水平スクロールビュー、スクロールビュー、スクローラー、オーバースクローラーがあります。

複数のイメージやテキストを切り替え表示するウイジェットとしてイメージスイッチャー、テキストスイッチャー、ビューアニメーター、ビューフリッパー、ビュースイッチャーがあります。

この他にタブでビューを切り替えるタブウイジェットがあります。

18-1　Gallery

画面上部に複数のアイテムを表示し、横スクロールすることができます。現在選択されているアイテムはビューの中心に配置されます。Gallery クラスは AdapterView クラスのサブクラスです。AdapterView クラスはアダプターを使って別に用意されたデータを子要素として持つビューです。

Gallery にアイテムを提供するには BaseAdapter クラスを使用します。このクラスの実装すべきメソッドは getCount、getItem、getItemId、getView です。getCount、getItem、getItemId メソッドは Adapter の単純なクエリー用に実装が必要なメソッドです。getView メソッドが実際にアイテムを提供するメソッドです。ここでは Drawable リソースの配列からアイテムを適用し、幅と高さをセットし、アイテムのスケールをセットし、アイテムの境界のスタイルを設定します。

ギャラリーアイテムの境界のスタイルは「res/values/」フォルダに Styleable リソースとしての attrs.xml を作成し、ここに「android:galleryItemBackground」を定義します。

選択されたアイテムの番号（0 スタート）は onItemClick メソッドの position 引数で取得できます。

「例題 18-1」3 つのイメージを Gallery で表示します。

・main.xml

```
<Gallery
    android:id="@+id/gallery"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
/>
```

・res/values/attrs.xml

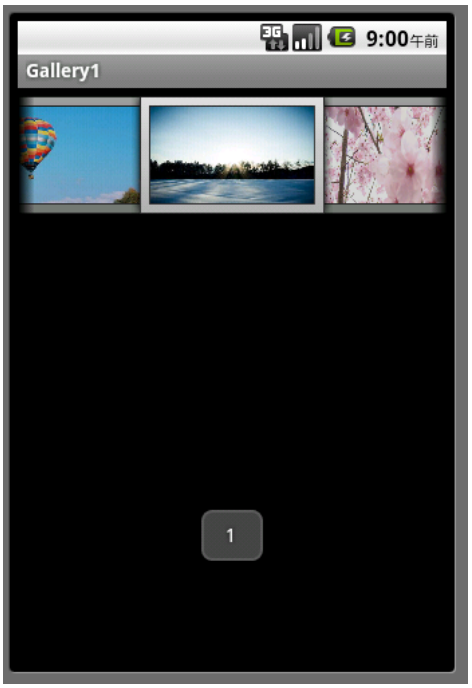
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="galleryback">
        <attr name="android:galleryItemBackground" />
    </declare-styleable>
</resources>
```

・Gallery1.java

```
package jp.gallery1;

import android.app.Activity;
```

```
import android.content.Context;
import android.content.res.TypedArray;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.*;
import android.widget.AdapterView.OnItemClickListener;

public class Gallery1 extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Gallery gal=(Gallery) findViewById(R.id.gallery);
        gal.setAdapter(new ImageAdapter(this));
        gal.setOnItemClickListener(new OnItemClickListener() {
            public void onItemClick(AdapterView<?> parent, View v, int position, long
id) {
                Toast.makeText(getApplicationContext(),"" +
position,Toast.LENGTH_SHORT).show();
            }
        });
    }
    public class ImageAdapter extends BaseAdapter {
        private int back;
        private Context context;
        private int[] id={
            R.drawable.balloon,
            R.drawable.hikari,
            R.drawable.sakura
        };
        public ImageAdapter(Context c) {
            context=c;
            TypedArray ta=obtainStyledAttributes(R.styleable.galleryback);

back=ta.getResourceId(R.styleable.galleryback_android_galleryItemBackground,0);
```

```
            ta.recycle();
        }
        public int getCount() {
            return id.length;
        }
        public Object getItem(int position) {
            return position;
        }
        public long getItemId(int position) {
            return position;
        }
        public View getView(int position, View convertView, ViewGroup parent) {
            ImageView iv=new ImageView(context);
            iv.setImageResource(id[position]);
            iv.setLayoutParams(new Gallery.LayoutParams(225,150));
            iv.setScaleType(ImageView.ScaleType.FIT_XY);
            iv.setBackgroundResource(back);
            return iv;
        }
    }
}
```



「注」this と getApplicationContext()

onItemClick メソッドはリスナーの内部クラスのメソッドとして定義されています。このため Toast の第 1 引数は通常は「this」を指定していましたが、内部クラスでは「this」を指定できないので、getApplicationContext()を使って

「Toast.makeText(getApplicationContext(), 」のようにします。もし「this」を使いたいなら「Gallery1Activity.this」のように「this」の前にクラス名を指定します。

「注」HVGA（320×480）でギャラリーのサイズを同じにするには

「iv.setLayoutParams(new Gallery.LayoutParams(225,150));」の 225→150、150→100→に変更。

「注」getApplicationContext()の使い方は「例題 13-7」参照。

「参考」Gallery をイメージとタイトルで構成します。イメージとタイトルのためのレイアウトを res/layout/item.xml に定義し、ArrayAdapter<BindData>クラスを使ってイメージとタイトルを Gallery に提供します。

イメージリソース ID とタイトルを格納するクラス BindData とそれらを格納するホルダクラス ViewHolder を定義し、 getView メソッドで Gallery に提供します。

・main.xml、attrs.xml

Gallery1 と同じ。

・res/layout/item.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:gravity="center"
    >
    <ImageView
        android:id="@+id/imageview"
        android:layout_width="150dp"
        android:layout_height="100dp"
    />
    <TextView
        android:id="@+id/textview"
```

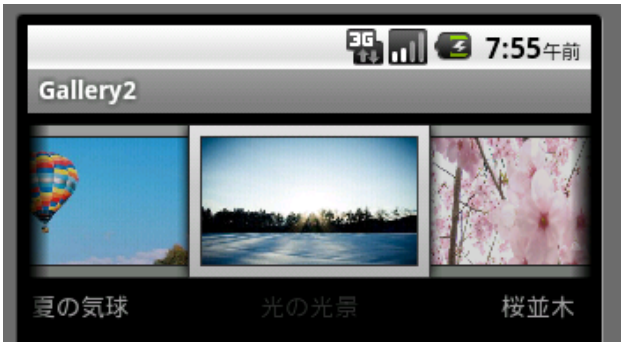
```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
    />
</LinearLayout>
```

・Gallery2.java

```
package jp.gallery2;

import android.app.Activity;
import android.content.Context;
import android.content.res.TypedArray;
import android.os.Bundle;
import android.view.*;
import android.widget.*;
import android.widget.AdapterView.OnItemClickListener;

public class Gallery2 extends Activity {
    public class BindData {
        int iconId;
        String title;
        BindData(int id, String s) {
            iconId=id;
            title=s;
        }
    }
    private BindData[] mDatas = {
        new BindData(R.drawable.balloon, "夏の気球"),
        new BindData(R.drawable.hikari, "光の光景"),
        new BindData(R.drawable.sakura, "桜並木"),
    };
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Gallery gal=(Gallery) findViewById(R.id.gallery);
        gal.setAdapter(new MyAdapter(this, R.layout.item, mDatas));
```

```
        gal.setOnItemClickListener(new OnItemClickListener() {
            public void onItemClick(AdapterView<?> parent,View v,int position,long
id) {
                Toast.makeText(getApplicationContext(),
""+position,Toast.LENGTH_SHORT).show();
            }
        });
    }
    static class ViewHolder {
        TextView textView;
        ImageView imageView;
    }
    public class MyAdapter extends ArrayAdapter<BindData> {
        private LayoutInflater inflater;
        private int id;
        private int back;
        public MyAdapter(Context context,int layoutId,BindData[] objects) {
            super(context,0,objects);
            TypedArray ta=obtainStyledAttributes(R.styleable.galleryback);

back=ta.getResourceId(R.styleable.galleryback_android_galleryItemBackground,0);
            ta.recycle();

inflater=(LayoutInflater)context.getSystemService(Context.LAYOUT_INFLATER_SER
VICE);
            id=layoutId;
        }
        @Override
        public View getView(int position,View convertView,ViewGroup parent) {
            ViewHolder holder;
            if (convertView==null) {
                convertView=inflater.inflate(id,parent,false);
                holder = new ViewHolder();
                holder.textView=(TextView)convertView.findViewById(R.id.textview);

holder.imageView=(ImageView)convertView.findViewById(R.id.imageview);
```

```
                convertView.setTag(holder);
            } else {
                holder=(ViewHolder)convertView.getTag();
            }
            BindData data= getItem(position);
            holder.textView.setText(data.title);
            holder.imageView.setImageResource(data.iconId);
            holder.imageView.setScaleType(ImageView.ScaleType.FIT_XY);
            holder.imageView.setBackgroundResource(back);
            return convertView;
        }
    }
}
```



19 章　アプリケーション・ウイジェット

　アプリケーション・ウイジェットとはホーム画面に直接貼り付けて利用できる小規模アプリケーションです。普通のプログラムのように画面全体を占有するのではなく、画面の一部に表示されます。アプリケーション・ウイジェットはプロバイダーの一種である AppWidgetProvider クラスを継承して作り、「プロバイダー」、「サービス」、「ブロードキャストレシーバ」を利用して動作します。

19-1　アプリケーション・ウイジェットの構成要素

AnalogClock ウイジェットをアプリケーション・ウイジェットとして登録する例を元にしてアプリケーション・ウイジェットの構成要素を説明します。

1.　マニフェスト

デフォルトの<activity>・・・</activity>は削除しブロードキャストレシーバの<receiver>・・・</receiver>に置き換えます。
<intent-filter>に「"android.appwidget.action.APPWIDGET_UPDATE"」を指定し更新アクションを設定します。
<meta-data>にアプリケーション・ウイジェットのホーム画面上での形状を記述した「@xml/app_widget」を指定し、ホーム画面に表示されるリモートビューを決めます。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="jp.appw1"
        android:versionCode="1"
        android:versionName="1.0">
    <application android:icon="@drawable/ic_launcher" android:label="My アナログ時計">
        <receiver android:name="AppW1">
            <intent-filter>
                <action
android:name="android.appwidget.action.APPWIDGET_UPDATE"/>
            </intent-filter>
            <meta-data android:name="android.appwidget.provider"
android:resource="@xml/app_widget"/>
        </receiver>
    </application>
</manifest>
```

「注」正式には「android:label="My アナログ時計"」は「android:label="@string/app_name"」のままにして strings.xml の「app_name」の定義を「<string name="app_name">My アナログ時計</string>」とします。

2. main.xml

アプリケーション・ウイジェットに載せるレイアウトを記述します。

```
<?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"
    >
    <AnalogClock
        android:id="@+id/AnalogClock"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
    />
</LinearLayout>
```

「注」アプリケーション・ウイジェットに載せるレイアウトは以下をサポートします。

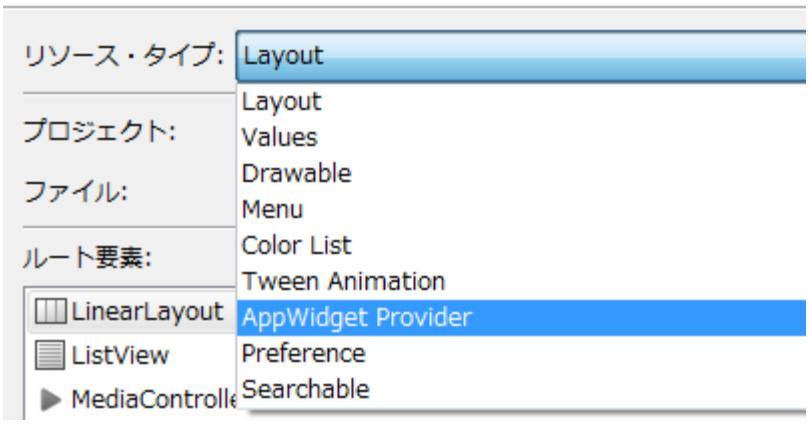
・FrameLayout
・LinearLayout
・RelativeLayout

配置できるウイジェットは以下をサポートします。

・AnalogClock
・Button
・Chronometer
・ImageButton
・ImageView
・ProgressBar
・TextView

3.　res/xml/app_widget.xml

アプリケーション・ウイジェットのホームスクリーン上での形状を記述した XML ファイルを res/xml フォルダに「app_widget.xml」として作成します。リソース・タイプに「AppWidget Provider」を選択します。




```
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:updatePeriodMillis="0"
    android:minWidth="72dp"
    android:minHeight="72dp"
    android:initialLayout="@layout/main">
</appwidget-provider>
```

通常、AppWidget は画面の縦横 4 分の 1 サイズが基本となっています。このマス目をいくつか組み合わせたサイズで設計されるようになっています。このことを考慮して minWidth と minHeight にアプリケーションウイジェットの幅と高さを指定します。
updatePeriodMillis はアプリケーション・ウイジェットの更新間隔をミリ秒で与えます。更新処理をしない場合は「0」を指定し、システムからの onUpdate 呼び出しを停止します。この値は 1800000 ミリ秒（30 分）未満は設定できません。これより小さい値を指定しても、1800000 と見なされます。つまり 30 分未満の更新周期で onUpdate を呼び出すような指定はこの updatePeriodMillis ではできません。
initialLayout にアプリケーション・ウイジェット本体を構成するレイアウトを指定します。

4.　AppW1.java

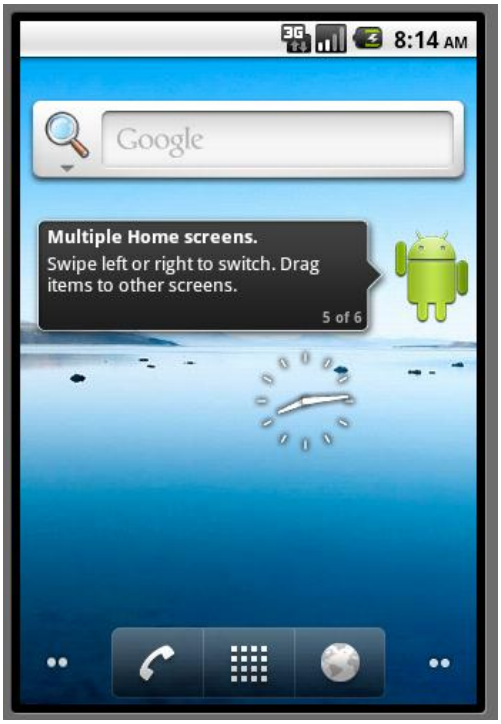
AppWidgetProvider クラスのサブクラスとして作成します。このクラスでは更新処理を行うために onUpdate メソッドを実装する必要があります。今回の例ではウイジェットの AnalogClock が自動的にシステムにより更新されますので、空のメソッドになっています。

```
package jp.appw1;

import android.appwidget.AppWidgetManager;
import android.appwidget.AppWidgetProvider;
import android.content.Context;

public class AppW1 extends AppWidgetProvider {
    @Override
    public void onUpdate(Context context, AppWidgetManager appWidgetManager,
int[] appWidgetIds) {
        super.onUpdate(context, appWidgetManager, appWidgetIds);
    }
}
```

5.　アプリケーション・ウイジェットの登録

ホーム画面を長押しすると「ホーム画面に追加」メニューがでますので、「ウイジェット」を選択します。

登録されているアプリケーション・ウイジェットの一覧が表示されますので今回作った「My アナログ時計」を選択します。マニフェストの「android:label="My アナログ時計"」で指定した名前が登録されています。



「注」Android 4.0 では Home 画面の  から「ウイジェット」を選択します。

20 章　マルチメデイア

マルチメデイアを扱うウイジェットとして MediaPlayer と VideoView があります。MediaPlayer はサウンドの演奏を行い、VideoView は動画の再生を行います。これらは「android.widget」パッケージで定義されています。ウイジェットの分類ではありませんが、この他にマルチメデイアを扱うクラスとして「android.media」パッケージの SoundPool、ToneGenerator、MediaRecorder、「android.webkit」パッケージの WebView があります。この章ではこれらのクラスの使用方法を説明します。

20-1　サウンドの再生

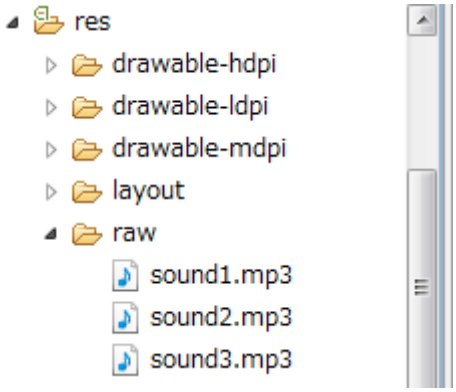
1.　音楽ファイル形式の種類

Android の仕様では MP3/AAC/3GP/MID/WMA/IMY/OTA/OGG などに対応することになっていますが、実機に搭載されたメディアプレーヤーにより再生できるフォーマットは異なります。代表的なフォーマットとして以下があります。

音楽ファイル形式	特徴
MP3 (MPEG Audio Layer-3)	映像データ圧縮方式の MPEG-1 で利用される音声圧縮方式。
WMA (Windows Media Audio)	Microsoft 社が開発した Windows 標準の音声圧縮方式。
3GP (Third Generation Partnership)	3GP プロジェクト(3GPP)が定めた標準規格のファイルフォーマット。
OGG	Xiph.org Foundation が開発した、ライセンスフリーな音声圧縮方式。

2.　MediaPlayer

サウンドファイル（~.mp3）をリソースフォルダの res/raw に配置しておきます。



サウンドの再生は MediaPlayer クラスを使用して次のように行います。create メソッドの第 1 引数には「getApplicationContext()」を指定します。this や getContext()では引数の型が合いません。seekTo(0)で先頭位置に設定します。

```
MediaPlayer mp;
mp=MediaPlayer.create(getApplicationContext(),R.raw.sound1);
mp.seekTo(0);
mp.start();
```

再生を停止するには次のようにします。

```
mp.stop();
mp.release();
mp=null;
```

「例題 20-1」リストビューに登録された音楽ファイルを選択して演奏します。
res/raw フォルダにサウンドファイルとして sound1.mp3～sound3.mp3 を配置しておきます。

・main.xml

```
<?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" >
    <ListView
        android:id="@+id/listview"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    />
</LinearLayout>
```

・Sound1.java

```
package jp.sound1;

import android.app.Activity;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.widget.*;
import android.widget.AdapterView.OnItemClickListener;

public class Sound1 extends Activity {
    private MediaPlayer mp=null;
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ArrayAdapter<String> adapter=new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1);
        adapter.add("序曲");
        adapter.add("勇者の行進");
        adapter.add("明日への道");
        adapter.add("演奏停止");
        ListView lv=(ListView)findViewById(R.id.listview);
        lv.setAdapter(adapter);
        lv.setOnItemClickListener(new ItemClick());
    }
    class ItemClick implements OnItemClickListener {
        public void onItemClick(AdapterView<?> parent, View view,int position, long
id) {
            int mid[]={R.raw.sound1,R.raw.sound2,R.raw.sound3};
            if (mp!=null){
                mp.stop();
                mp.release();
                mp=null;
            }
            if (position!=3){
                mp=MediaPlayer.create(getApplicationContext(),mid[position]);
                mp.seekTo(0);
                mp.start();
            }
        }
    }
}
```

3.　再生位置の取得

曲の長さは「mp.getDuration()」、再生位置は「mp.getCurrentPosition()」で取得できます。単位はミリ秒です。MediaPlayer クラスには現在の再生状態を問い合わせるメソッドがありませんので、独自のハンドラを使って１秒ごとに再生状態を調べます。

・main.xml

```
<?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" >
    <ProgressBar
        android:id="@+id/pbar"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        style="?android:attr/progressBarStyleHorizontal"
    />
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="演奏開始"
    />
    <Button
        android:id="@+id/button2"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="演奏停止"
    />
</LinearLayout>
```

・Sound2.java

```
package jp.sound2;

import android.app.Activity;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.*;

public class Sound2 extends Activity implements Runnable{
    private MediaPlayer mp=null;
    private Handler hd=new Handler();
    private ProgressBar pbar;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        pbar=(ProgressBar) findViewById(R.id.pbar);

        Button bt1=(Button) findViewById(R.id.button1);
        bt1.setOnClickListener(new OnClickListener(){
            public void onClick(View v) {
                mp=MediaPlayer.create(getApplicationContext(),R.raw.sound1);
                mp.seekTo(0);
                mp.start();
                pbar.setMax(mp.getDuration());
                hd.post(Sound2.this);
            }
```

```
        });
        Button bt2=(Button) findViewById(R.id.button2);
        bt2.setOnClickListener(new OnClickListener(){
            public void onClick(View v) {
                if (mp!=null){
                    mp.stop();
                    mp.release();
                    mp=null;
                    hd.removeCallbacks(Sound2.this);
                }
            }
        });
    }
    public void run() {
        setTitle((mp.getCurrentPosition()/1000)+"/"+(mp.getDuration()/1000)+"秒");
        pbar.setProgress(mp.getCurrentPosition());
        hd.postDelayed(this,1000);
    }
}
```

21章　リソース

リソースをイメージや文字列としてアプリケーションから常に外部化しておくことにより、それらを独立させて保持することが可能となります。またリソースの外部化により、異なる言語や画面サイズといった各種のデバイス設定をサポートできるようになります。リソースをプロジェクトの res フォルダに、タイプや設定によりグループ化したさまざまなサブフォルダを用意してまとめておくことをお勧めします。

この章では文字列リソース、カラー状態リストリソース、Drawable リソース、メニューリソース、レイアウトリソース、スタイルリソース、その他のリソースについて説明します。アニメーションリソースについては 22 章で説明します。

21-1　リソースタイプ

1.　グループ化

リソースにはイメージリソース、レイアウトリソース、文字列リソースなど色々な種類があります。これらは res フォルダの特定のサブフォルダにグループ化して配置する必要があります。主なリソースフォルダとして以下があります。

表 21-1　リソースフォルダ

リソースフォルダ	意味
anim	Tween アニメーションを定義する XML ファイル。
color	カラーの状態リストを定義する XML ファイル。
drawable	ビットマップファイル(.png、jpg、gif)、 リサイズ可能なビットマップ（9png）、状態リスト Drawable、外形 Drawable、アニメーション Drawable、その他の Drawable。
layout	レイアウトを定義する XML ファイル
menu	オプションメニュー、コンテキストメニュー、サブメニューなどのアプリケーションメニューを定義する XML ファイル。
raw	そのままの形式で任意に保存するファイル。
values	文字列、数値、カラーなどの単純な値を含む XML ファイル。arrays.xml(タイプ化配列)、colors.xml(カラー値)、dimens.xml(ディメンション値)、strings.xml(文字列)、styles.xml(スタイル)。
xml	実行時に Resources.getXML()を呼び出すことにより読み込みが可能な任意の XML ファイル。

アプリケーションが特定のデバイス設定をサポートするために代替リソースを提供する必要があります。例えば異なる画面密度用には代替の Drawable リソース、異なる言語用には代替の文字列リソースを含める必要があります。実行時に Android は現在のデバイス設定を検知し、アプリケーションに適切なリソースをロードします。

表 21-1 で示す「リソースフォルダ名」に対し「リソースフォルダ名－個別の設定名」で示す個別リソースフォルダを作り、異なる環境に対し適切なリソースを選択させることができます。「drawable」の「drawable-ldpi」、「drawable-mdpi」、「drawable-hdpi」、「drawable-xhdpi」がそうです。個別の設定名として表 21-2 があります。

表 21-2　個別の設定名

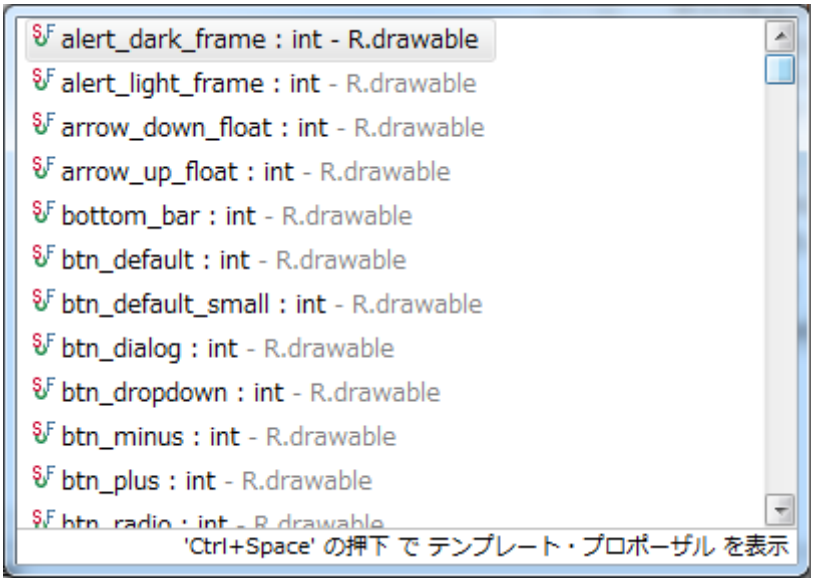
対象	個別の設定名	意味
MCC と MNC	mcc310、mcc310-mnc、004mcc、208-mnc00	モバイル国コード(mobile country code:MCC)で、オプションとしてデバイスの SIM カードからのモバイルネットワークコード（ mobile network code:MNC)が後ろに付加されます。例えば、mcc310 は U.S.のすべてのキャリアで、mcc310-mnc004 は U.S.の Verizon です。
言語と地域	en、fr、en-rUS、fr-rFR、fr-rCA	言語は、2 文字で定義された ISO 639-1 言語コードで、オプションとして 2 文字の ISO 3166-1-alpha-2 の地域コードが後ろに付加されます(小文字で"r"を前に付けて)。
画面サイズ	small、normal、large	small:低密度の QVGA 画面で利用可能な領域を基準とした画面です。normal:従来の中密度の HVGA 画面を基準とした画面です。large:中密度の VGA 画面で利用可能な領域を基準とした画面です。
画面アスペクト	long、notlong	long: WQVGA、WVGA、FWVGA などの長い画面。notlong: QVGA、HVGA、および VGA などの長くない画
画面オリエンテーション	port、land	port: デバイスがポートレートのオリエンテーション(垂直)。land:デバイスがランドスケープのオリエンテーション(水平)。
ドックモード	car、desk	car: デバイスが車載ドックにある。desk: デバイスがデスクドックにある。
ナイトモード	night、notnight	night: 夜間。notnight:日中。
画面ピクセル密度	ldpi、mdpi、hdpi、xhdpi、nodpi	ldpi: 低密度画面、およそ 120dpi。mdpi:中密度画面(従来の HVGA で)、およそ 160dpi。hdpi:高密度画面、およそ 240dpi。xhdpi:超高密度画面、およそ 320dpi。nodpi: デバイスの密度に合わせて拡大縮小したくないビットマップリソースにこれを使用します。
タッチ画面タイプ	notouch、stylus、finger	notouch:タッチ画面がないデバイス。stylus:タッチペンの使用に適した抵抗方式タッチ画面のデバイス。finger:タッチ画面のあるデバイス。
キーボードの使用	keysexposed、keyssoft	keysexposed:利用可能なキーボードのあるデバイス。keyssoft:ソフトウェアキーボードが見えるかどうかにかかわらず、それが有効なデバイス。

主なテキスト入力方式	nokeys、qwerty、12key	nokeys:入力するハードウェアキーがないデバイス。qwerty:ハードウェア qwerty キーボードがあるデバイス。12key:ハードウェア 12-key キーボードがあるデバイス。
ナビゲーションキーの使用	navexposed、navhidden	navexposed:ユーザが利用可能なナビゲーションキーがある。navhidden: 利用可能なナビゲーションキーがない。
主な非タッチナビゲーション方式	nonav、dpad、trackball、wheel	nonav:タッチスクリーン以外にナビゲーションする機器がないデバイス。dpad:ナビゲーション用に方向を変えるパッド(d-pad)があるデバイス。trackball:ナビゲーション用にトラックボールがあるデバイス。wheel:ナビゲーション用に方向を変えるホイールがあるデバイス。
システムバージョン（API レベル)	v3、v4、v7	デバイスによりサポートされる API レベルです。例えば、v1 は API レベル 1(Android 1.0 以上のデバイス)で、v4 は API レベル 4(Android 1.6 以上のデバイス)になります。

2. システムリソース

システムリソースは「android.R.」と入力すると、以下のような項目が表示されます。すでに「android.R.layout.simple_list_item_1」、「android.R.id.text1」などのシステムリソースは「初級　基礎編」の「3-14　ArrayAdapter」で紹介してあります。

さらに「android.R.drawable.」と入力すると、以下のようなシステムアイコンの ID 一覧が表示されます。



「例題 21-1-1」android.R.drawable.ic_popup_reminder というシステムアイコンを表示します。

・main.xml

```
<ImageView
    android:id="@+id/image"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
/>
```

・Icon2.java

```
package jp.icon2;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ImageView;

public class Icon2 extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ImageView image=(ImageView)findViewById(R.id.image);
        image.setImageResource(android.R.drawable.ic_popup_reminder);
    }
}
```

「例題 21-1-2」システムアイコンの一覧をリストビューに表示します。システムアイコンのデータは以下のようにして得られます。

```
Field[] fields = android.R.drawable.class.getFields();
```

このデータからアイコンの ID と ID 名を取得するには以下のようにします。

```
    int n=fields.length;
    int img[]=new int[n];
    String text[]=new String[n];
    for (int i=0;i<n;i++){
        try {
            img[i]= fields[i].getInt(null);
        } catch (IllegalAccessException e) {}
        text[i]=fields[i].getName();
    }
```

この img[]と text[]データをリストビューに表示します。

・main.xml

```
<ListView
    android:id="@+id/listview"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
/>
```

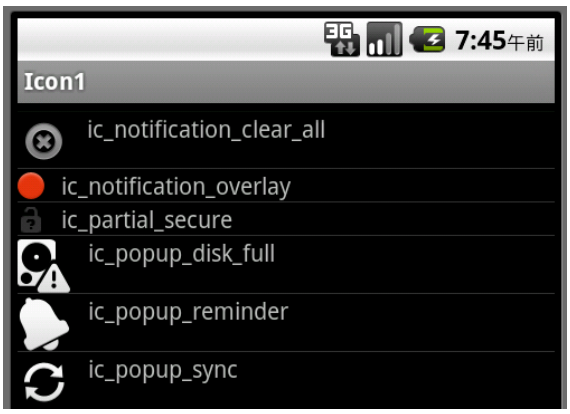
・res/layout/list.xml

```
<?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"
	>
	<ImageView
		android:id="@+id/image"
		android:paddingTop="2dp"
		android:layout_height="wrap_content"
		android:layout_width="wrap_content"
	/>
	<TextView
		android:id="@+id/text"
		android:paddingLeft="10dp"
		android:layout_width="wrap_content"
		android:layout_height="wrap_content"
	/>
</LinearLayout>
```

・Icon1.java

```
package jp.icon1;

import android.app.Activity;
import android.os.Bundle;
import android.widget.*;
import java.util.*;
import java.lang.reflect.Field;

public class Icon1 extends Activity {
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);
		Field[] fields = android.R.drawable.class.getFields();
```

```
            int n=fields.length;
            int img[]=new int[n];
            String text[]=new String[n];
            for (int i=0;i<n;i++){
                try {
                    img[i]= fields[i].getInt(null);
                } catch (IllegalAccessException e) {}
                text[i]=fields[i].getName();
            }
            List<HashMap<String,Object>> list=new
ArrayList<HashMap<String,Object>>();
            Map<String, Object> map;
            for (int i=0;i<img.length;i++){
                map=new HashMap<String,Object>();
                map.put("img",img[i]);
                map.put("text",text[i]);
                list.add((HashMap<String,Object>)map);
            }
            ListView lv=(ListView)findViewById(R.id.listview);
            SimpleAdapter adapter=new SimpleAdapter(this,list,
                R.layout.list,new String[] {"img","text"},
                new int[] {R.id.image,R.id.text});
            lv.setAdapter(adapter);
        }
}
```



22 章　アニメーション

アニメーションのタイプは大きく 2 つに分類できます。Tween アニメーション(Animation クラス)と Frame アニメーション(AnimationDrawable クラス)です。Tween アニメーションは 1 つのイメージを連続に変化させるタイプで、移動、フェード、回転、拡大・縮小、それらを組み合わせて変化させます。Frame アニメーションは順番にイメージを並べて表示してアニメーションにするタイプでパラパラ漫画のような表示を行います

22-1　Tween アニメーション

XML で定義されたフェード、伸縮、移動、回転などといった変化を実行するアニメーションです。Tween アニメーションを行うにはアニメーションの内容を res/anim フォルダに anim.xml として定義します。アニメーション定義は<set>要素内に<alpha>、<scale>、<translate>、<rotate>などのアニメーション要素を記述します。<set>のアニメーション要素は複数指定できます。

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false">
    <rotate
        android:fromDegrees="0"
        android:toDegrees="360"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="4000"
    />
</set>
```

このアニメーション定義ファイルをイメージビューiv に設定するには以下のようにします。Tween アニメーションは「android.view.animation.Animation」クラスを使って制御します。

```
Animation anim=AnimationUtils.loadAnimation(this, R.anim.anime);
iv.startAnimation(anim);
```

1.　<set>

アニメーション要素(<alpha>、<scale>、<translate>、<rotate>）や他の <set> 要素を保持するコンテナです。AnimationSet クラスを表します。

属性	意味
interpolator	Interpolator リソースへのリファレンス。
shareInterpolator	子要素に同じ Interpolator を共有させる（true)、させない（false)。

2. <alpha>

アニメーションのフェードイン、フェードアウトします。AlphaAnimation クラスを表します。以下の属性で不透明度を 0.0～1.0 の float 値で指定します。0.0 は透明、1.0 は不透明です。

属性	意味
fromAlpha	開始の不透明度。
toAlpha	終了の不透明度。
duration	アニメーション時間（ミリ秒）。

透明度を 0.0～1.0 まで 4000 ミリ秒の間で変化させます。

```
<alpha
    android:fromAlpha="0.0"
    android:toAlpha="1.0"
    android:duration="4000"
/>
```

3. <scale>

アニメーションの拡大・縮小を行います。ScaleAnimation クラスを表します。以下の属性で拡大・縮小の倍率と中心位置を float 値で指定します。倍率の「1.0」は変化しない（等倍）を意味します。pivotX と pivotY が（0,0）の場合は左上隅を中心に拡大・縮小はすべて下と右に伸縮します。

属性	意味
fromXScale	開始の X 倍率。
toXScale	終了の X 倍率。
fromYScale	開始の Y 倍率。
toYScale	終了の Y 倍率。
pivotX	拡大縮小の中心 X 座標。
pivotY	拡大縮小の中心 Y 座標。
duration	アニメーション時間（ミリ秒）。

縮尺を 1.0～0.0 まで中心を軸にして 4000 ミリ秒の間で縮小させます。

```
<scale
    android:fromXScale="1.0"
    android:toXScale="0.0"
```

```
    android:fromYScale="1.0"
    android:toYScale="0.0"
    android:pivotX="50%"
    android:pivotY="50%"
    android:duration="4000"
/>
```

4. <translate>

アニメーションの X 方向、Y 方向の水平移動を行います。TranslateAnimation クラス を表します。以下の属性で移動量を float 値またはパーセンテージで指定します。通常の位置に対する相対ピクセル("5"など)、要素の幅に対する相対パーセンテージ("5%"など)、親の幅に対する相対パーセンテージ("5%p"など)のいずれか一方で表現します。

属性	意味
fromXDelta	開始の X オフセット値。
toXDelta	終了の X オフセット値。
fromYDelta	開始の Y オフセット値。
toYDelta	終了の Y オフセット値。
duration	アニメーション時間（ミリ秒)。

対象となるイメージの左端から右端まで 4000 ミリ秒の間で平行移動します。

```
<translate
    android:fromXDelta="0%"
    android:toXDelta="100%"
    android:duration="4000"
/>
```

5. <rotate>

アニメーションの回転を行います。RotateAnimation クラス を表します。 以下の属性で回転角度と回転中心座標を float 値で指定します。pivotX、pivotY はパーセンテージ指定もできます。

属性	意味
fromDegrees	開始位置の角度（度数)。
toDegrees	終了位置の角度（度数)。
pivotX	回転の中心となる X 座標。
pivotY	回転の中心となる Y 座標。
duration	アニメーション時間（ミリ秒)。

イメージの中央を原点として 0 度～360 度まで 4000 ミリ秒の間で回転します。

```
<rotate
    android:fromDegrees="0"
    android:toDegrees="360"
    android:pivotX="50%"
    android:pivotY="50%"
    android:duration="4000"
/>
```

「例題 22-1」イメージを回転しながら縮小します。イメージは画面中央に配置します。

・res/anim/anime.xml

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false"
    >
    <rotate
        android:fromDegrees="0"
        android:toDegrees="360"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="4000"
    />
    <scale
        android:fromXScale="1.0"
```

```
        android:toXScale="0.0"
        android:fromYScale="1.0"
        android:toYScale="0.0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="4000"
    />
</set>
```

・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="アニメ開始"
    />

    <ImageView
        android:id="@+id/image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/sakura"
    />
</LinearLayout>
```

・Tween1.java

```
package jp.tween1;

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

```
import android.view.View;
import android.view.View.OnClickListener;
import android.view.animation.*;
import android.widget.*;

public class Tween1 extends Activity implements OnClickListener{
	@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) {
		ImageView iv=(ImageView) findViewById(R.id.image);
		Animation anim=AnimationUtils.loadAnimation(this, R.anim.anime);
		iv.startAnimation(anim);
	}
}
```

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．ソフトウエアラボ」代表。

「主な著書」

「入門ソフトウエアシリーズ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 標準コースウエア」、「基礎から学べる JavaScript 標準コースウエア」、「基礎から学べる C 言語標準コースウエア」、「基礎から学べる PHP 標準コースウエア」、「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,ISOC99 対応]」ほか（以上技術評論社）

Android プログラミング Bible

中級　Android 的プログラミング法

2014 年 2 月 1 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

定価＝1,500 円（税込）