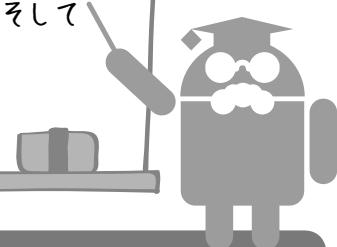


第6章

ためしてわかる Androidのしくみ

著：飯塚康至

ここでは、簡単なAndroidアプリケーションを実際に作成しながら、Androidアプリケーションの構成、作成方法、そして実行の方法について確認していきます。



6-1-1 Android アプリケーションの作成と実行

本講座で初めてとなるAndroidアプリケーションを実際に作成し、次の3点を学んでいきましょう。

- ①Androidアプリケーションの作成から実行までの流れ
- ②Androidアプリの基本構造
- ③Androidアプリが実行されてから終了するまでのライフサイクル

初めてのAndroidアプリを作ってみよう

それでは、初めてのAndroidアプリケーションを作成してみましょう。Androidアプリケーションを作成するには、Eclipse上で「Androidアプリケーションプロジェクト」(Android Application Project)を作成します。プロジェクトとは、Eclipse上でプログラムを管理するための大きな単位です。「File」メニューから「New」→「Android Application Project」を選択します(図1)。

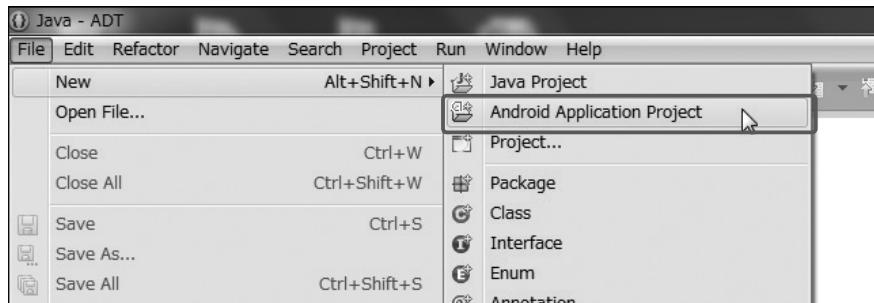


図1: Android Application Project

すると、次のように「New Android Application」のダイアログボックスが表示されます(図2)。

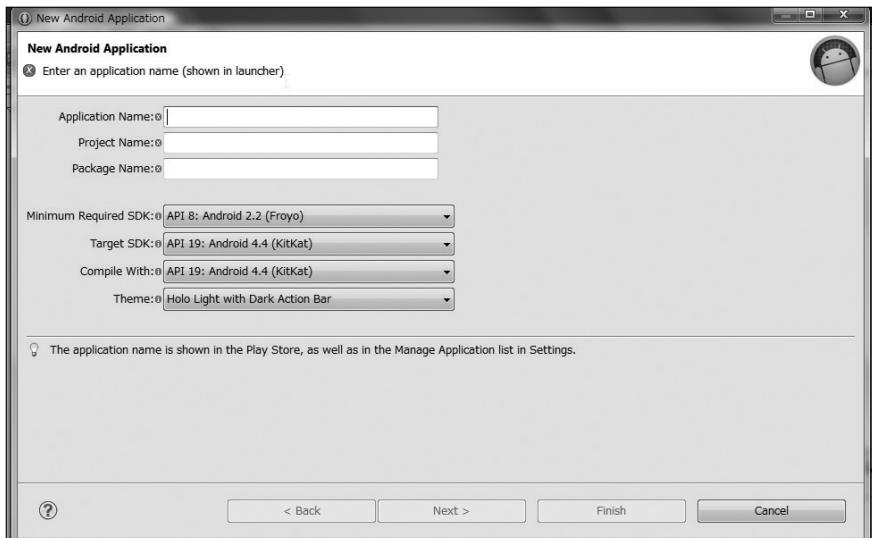


図2: New Android Application

各項目の意味は次の通りです。

- Ⓐ Application Name: アプリケーションランチャーに表示されるアプリケーション名
- Ⓑ Project Name: Eclipse上でAndroidのプロジェクトを区別するための名前
- Ⓒ Package Name: 世界中のAndroidアプリケーションと区別するための名前
- Ⓓ Minimum Required SDK: アプリケーションが動作する最低レベルのAPIレベル
- Ⓔ Target SDK: 作成するアプリケーションが動作する最高レベルのAPIレベル
- Ⓕ Compile With: ビルド時に利用するAPIレベル
- Ⓖ Theme: 適用するテーマ(全体の見た目)

まず、アプリケーションの名前をApplication Nameのところに入力してみましょう。アプリケーション名は日本語でも大丈夫ですが、ここでは半角で「Hello」という名前を入力してみます。すると、「Project Name」にHelloが自動で入力され、「Package Name」のところは、「com.example.hello」と入力されます(図3)。

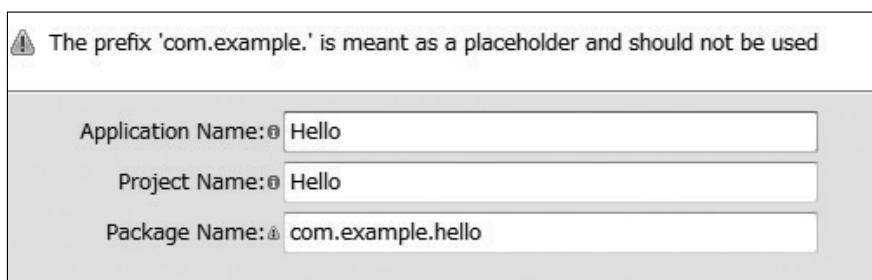


図3: アプリケーション名の指定

図3のように、ダイアログボックスの左上とパッケージ名(Package Name)の部分に、黄色いアイコンが表示されています。パッケージ名とは、世界中のAndroidアプリケーションと自分のAndroidアプリケーションを区別するため名前です。com.example.helloのままだと、他の誰かのアプリケーションと重複してしまうかもしれません。そこで、パッケージ名が重複しないように修正してくださいと、黄色いアイコンが表

示されているのです。一般的には、所属する会社や組織のURLを逆に並べたものをパッケージ名として利用します。ここでは次のようにパッケージ名をつけましょう。

`jp.techinstitute.重複しない文字列.hello`

「重複しない文字列」のところは、例えば山田太郎さんのアプリケーションの場合、taroyamadaでもいいですし、大学や専門学校などでアプリケーションを作成する場合は、学籍番号を利用してもかまいません。ただし、パッケージ名を付けるときは次の点に注意しましょう。

1. パッケージ名は2つ以上の文字列を「.」(ピリオド)で繋げる。
2. 各文字列はJavaの仕様上、数字から始めることができないので、たとえば学籍番号が14a001の場合は、学籍番号の前に適当なアルファベットをつけるようにしましょう。(例:j14a001など)。
3. パッケージ名にはJavaの言語仕様上「-」(ハイフン)は利用できないので-は「_」(アンダースコア)などに置き換えるようにしましょう。
4. パッケージ名はすべて小文字アルファベットで記述するようにしましょう。

パッケージ名を変更すると、次のように黄色いアイコンが消えます。

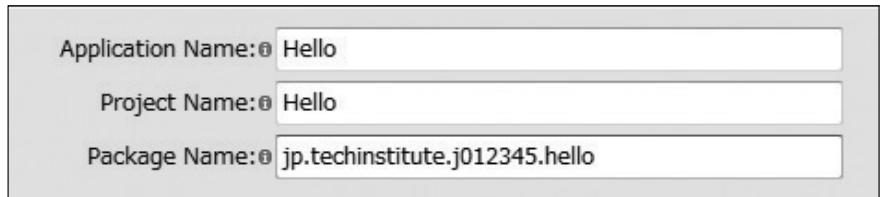


図4: パッケージ名の指定

今回は、アプリケーションが動作する最低限のAPIレベル(OSのバージョン)である「Minimum Required SDK」を、次のように4.0に設定しましょう(図5)。

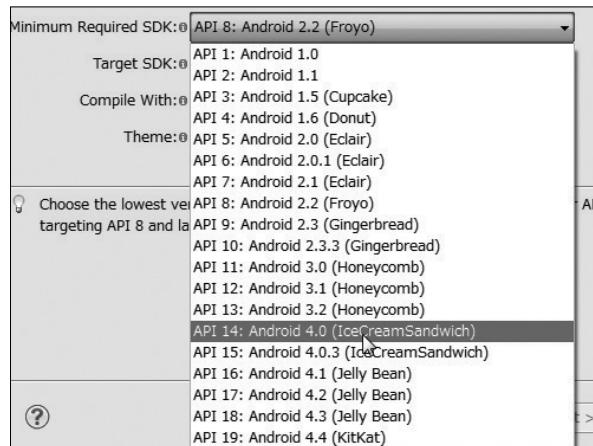


図5: Minimum Required SDKの設定

その他の項目はそのまま、「Next>」ボタンをクリックします。すると次のような画面が表示されますが(図6)、ここではそのまま「Next>」をクリックします。

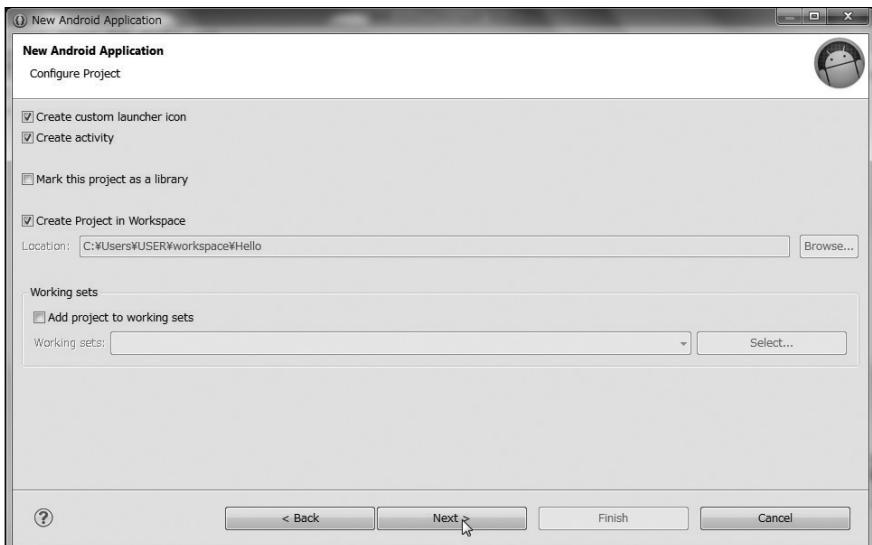


図6:何も変更せずNext>をクリックする画面

すると、アプリケーションアイコンを設定する画面になります(図7)。

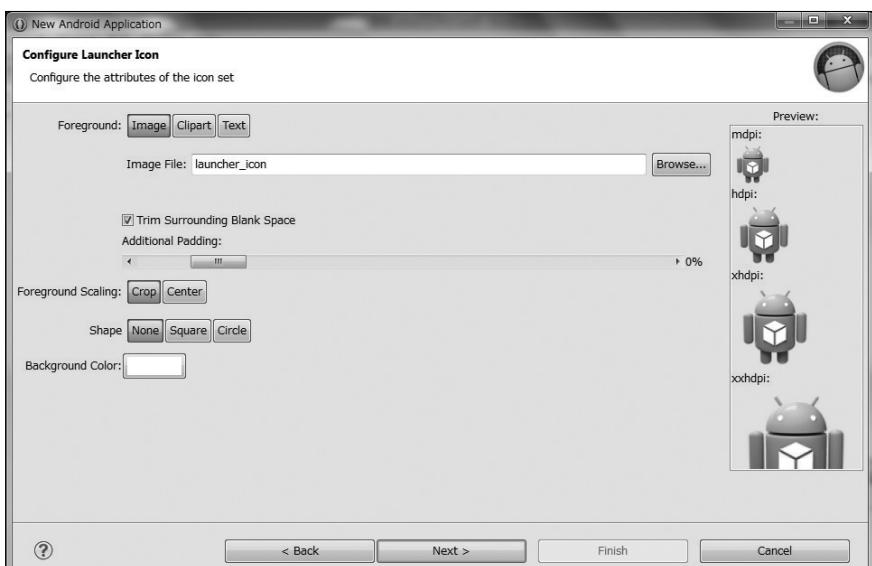


図7:アプリケーションアイコン画面

ここでは、アプリケーションで利用するアイコンを設定することができます。アイコンには画像やクリップアート(イラスト)、文字列を設定することができます(図8)。

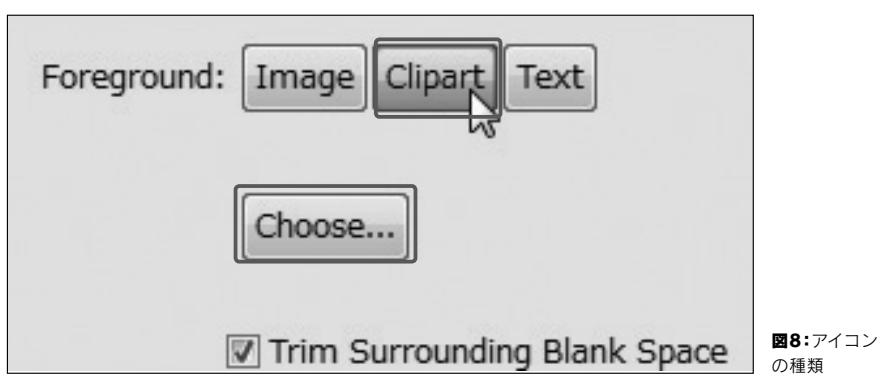


図8:アイコンの種類

たとえば「Clipart」をクリックすると、次のようにデフォルトで定義されている画像の一覧が表示されます(図9)。

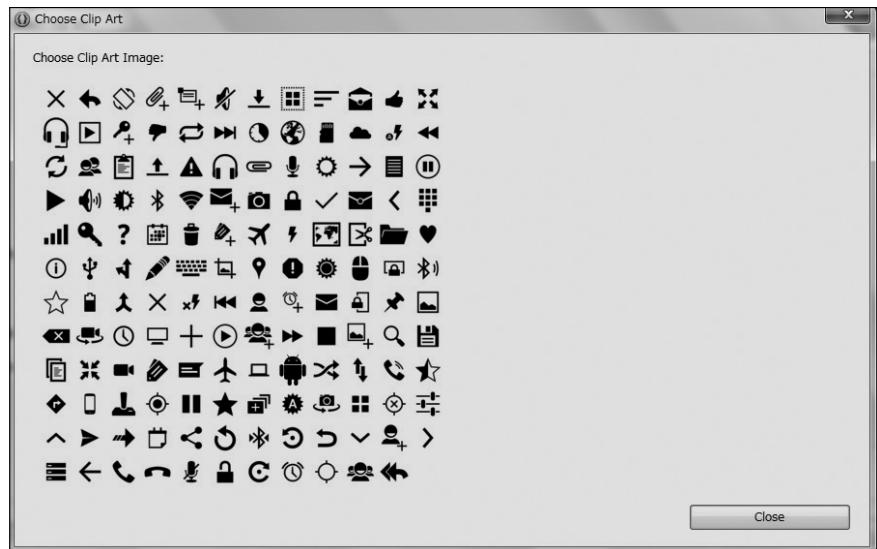


図9:クリップアート一覧

どのクリップアートを選択しても良いのですが、初めてのAndroidアプリケーションなので、マスコットキャラクターのドロイド君(通称)を選択してみましょう(図10a)。

クリップアートを選択すると、アイコンの選択画面が次のように変更されます(図10b)。

Androidのマスコットである緑色のロボットは、「Android Robot」や「ドロイド君」となど呼ばれていますが、グーグルによると正式な名称は決められていないのだそうです。

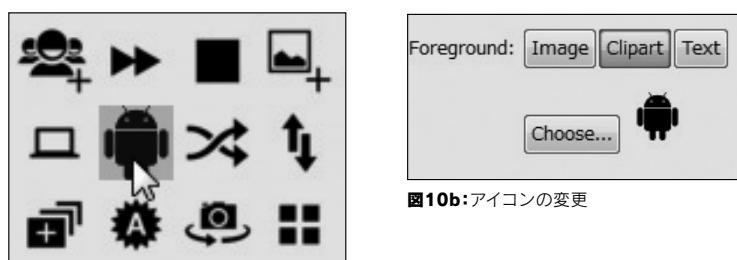


図10: クリップアートの選択

また、アイコンの色や形状も変更することができます。ダイアログ右下を見ると「Foreground Scaling」や「Shape」といった項目が並んでいます(図11)。

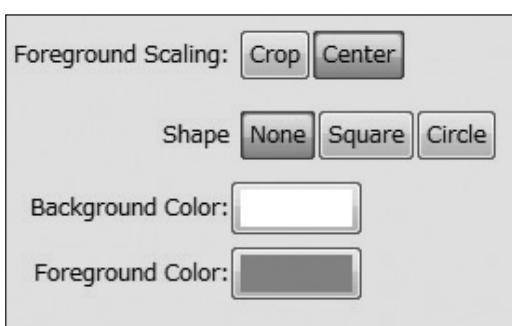


図11:アイコンの色や形状の設定項目

Foreground Scalingは、アイコン画像をどのように扱うか決めるところで、「Crop」を選択するとアイコン領域の大きさからはみ出た部分は切り取られ、「Center」を選択するとアイコン領域の大きさに収まるように縮小されます。Cropを選択すると、ダイアログ右側の「Preview」に、次のように表示されます(図12)。



図12:Cropを選択した場合

また、Centerを選択した場合は次のように表示されます(図13)。

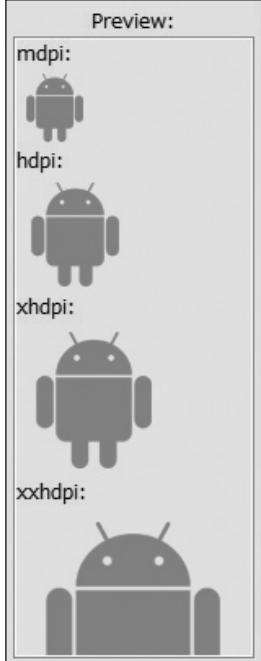


図13:Centerを選択した場合

ここで、Previewに表示されている「mdpi」や「hdpi」について解説しましょう。「DPI」とはdots per inchの略で、1インチの中にどのくらいのドットがあるかを示す単位です。Android端末には、低解像度から高解像度までさまざまな解像度の端末があるので、同じ大きさの画像だと画面上の見た目が異なってしまいます。そのため、それぞれの端末でアイコン画像が同じように見えるように、解像度に応じて大きさの違うアイコンを複数用意することになっています。Androidでは、次のような解像度が定義されています(表1)。

表記	意味	解像度
mdpi	中解像度	160(基本解像度)
hdpi	高解像度	240(1.5倍)
xhdpi	超高解像度	320(2倍)
xxhdpi	超超高解像度	480(3倍)

表1:解像度の種類

この「Configure Launcher Icon」ダイアログでアイコンを定義することによって、各解像度に対応したアイコンが自動で生成されます。

また、アイコンの形は「Shape」の項目で設定できます。「None」は画像のみで、「Square」に設定すると、次のように四角いアイコンになります(図14)。「Circle」を指定すると、次のように丸いアイコンになります(図15)。

また、「Background Color」と「Foreground Color」を指定することで、背景色やアイコンの色を変更できます。実際にForeground Colorをクリックすると、次のようなカラーパレットが表示され、ここで自由に色を指定することができます(図16)

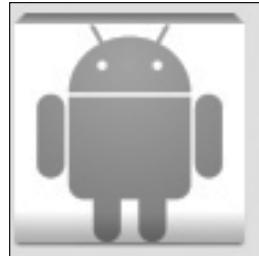


図14:Squareを指定した場合



図15:Circleを指定した場合

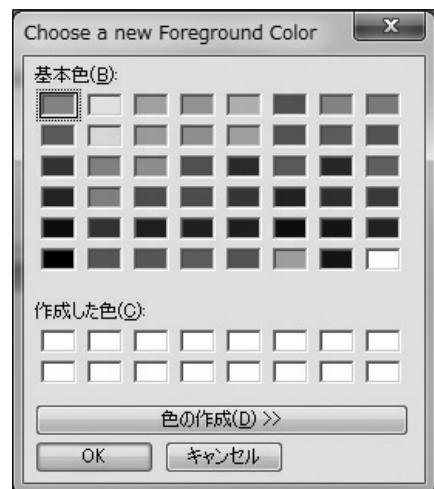


図16:カラーパレット

自分の好みに応じてアイコンの形状や色を変更してみましょう。変更したら「Next>」をクリックします。



図17:Activity指定ダイアログボックス

そうすると、「Activity」(アクティビティ)を指定するダイアログボックスが表示されます(図17)。Activityの詳細はこれから学習していきますが、簡単に説明すると、ActivityとはAndroidアプリケーションの基本となるクラスです。ここでは、「Empty Activity」を選択し「Next >」をクリックします。

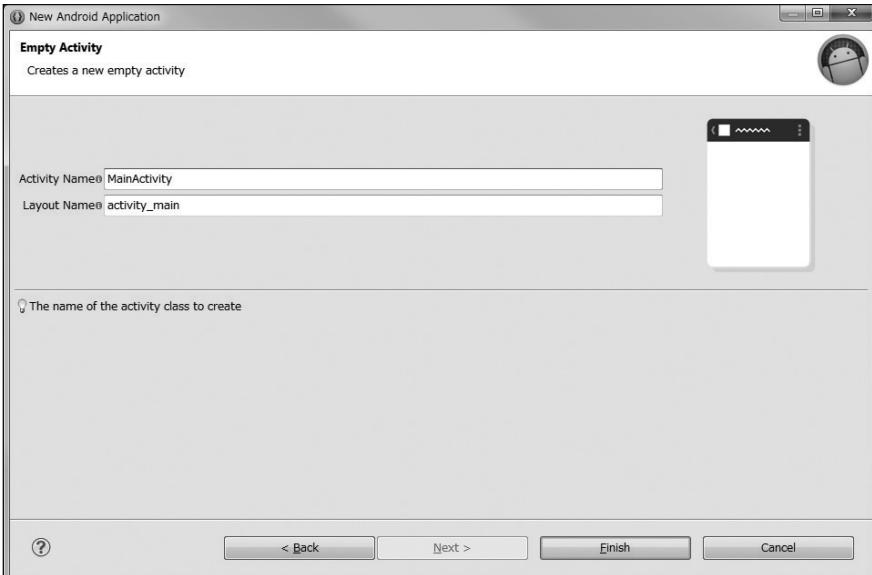


図18:Empty Activity設定画面

最後にEmpty Activityのダイアログボックスが表示されます。ここではそのまま「Finish」ボタンをクリックします(図18)。「Finish」をクリックすると、Eclipseの左側に表示されている、「Package Explorer」に、「Hello」というAndroidプロジェクトが追加されます。プロジェクト名のHelloを展開し、次のようにフォルダー・ファイルが配置されていることを確認しましょう(図19)。

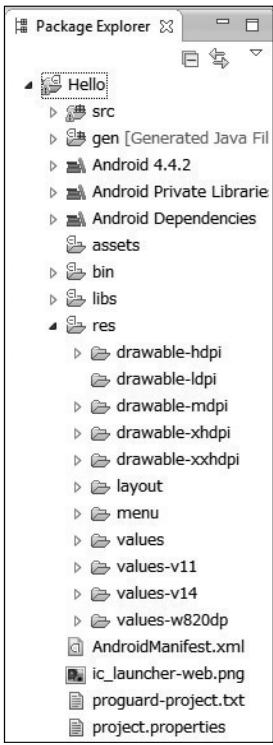


図19:追加されたプロジェクト

また、Eclipse中央にAndroidのレイアウトを編集する画面が表示されることも確認しましょう（図20）。



図20:Androidレイアウト編集画面

Androidプロジェクトの理解に入る前に、Androidアプリケーションの実行方法について確認しておきましょう。Androidアプリケーションを実行するには、次の2つの方法があります。

- 1.エミュレーターを作成し、パソコンの内部で実行する
- 2.実機をUSBケーブルで接続してアプリケーションを実行する

ここでは実機を接続して、実機上で作成したAndroidプロジェクトを実行してみます。Androidプロジェクトを実行するには、Package Explorer上に表示されているプロジェクト名Helloを右クリックし、表示されたドロップダウンメニューから、

「Run As」→「Android Application」を選択します(図23)。

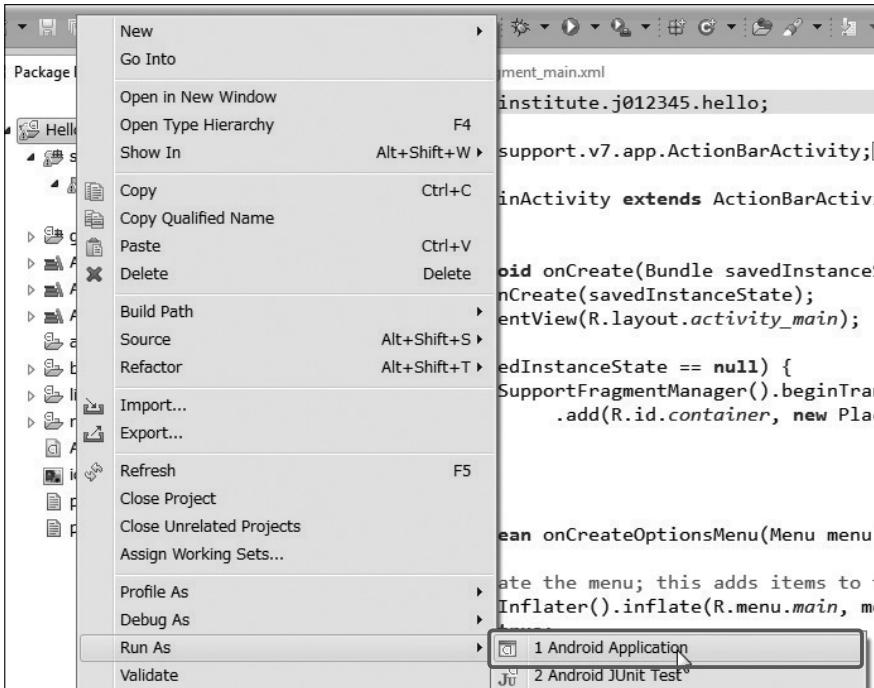


図23:Androidプロジェクトの実行

すると、接続されたAndroid端末にアプリケーションが転送され、次のように実行されます(図24)。

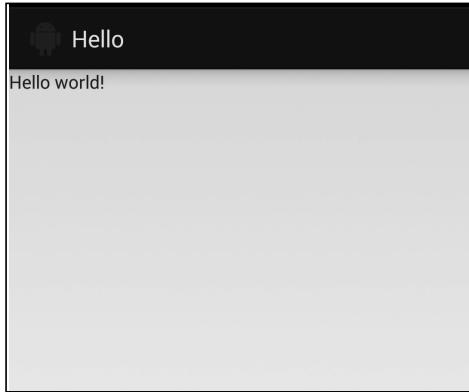


図24:実行例



Android端末を認識させるには

Android端末とパソコンにUSB接続して、開発機として認識させるには、

- ①端末側のUSBデバッガを有効にする
 - ②各端末用のADBドライバーをインストールする
- の2点が必要になります。

①はAndroid OSのバージョンによって設定方法が異なります。Android 4.2以上の場合には「設定」→「端末情報」→「ビルト番号」を7回タップします。すると「設定」→「開発者向けオプション」を使用することができるようになります。4.2以前のバージョンは、デフォルトで開発者向けオプションは表示されています。

「開発者向けオプション」を開くと次のように「USBデバッグ」という項目があるのでチェックを入れ、有効にします。

②は端末メーカーからUSBドライバーが公開されているので「端末名 ADBドライバー」で検索し、各公式サイトの指示に従って設定するようにしましょう。

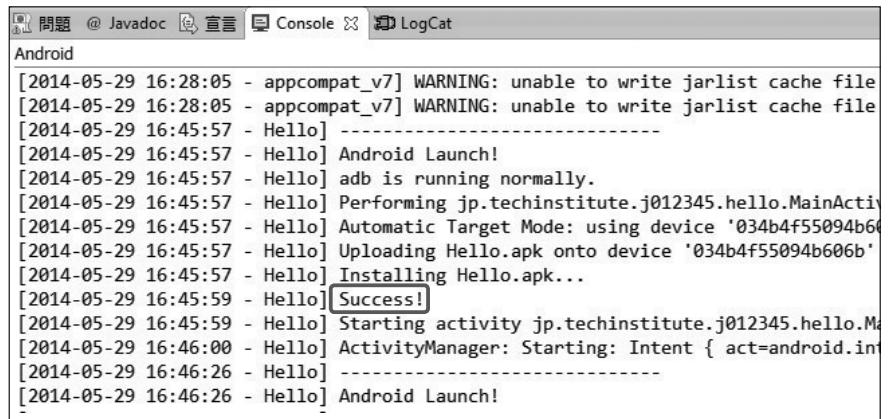


図21:開発者向けオプション



図22:USBデバッグ

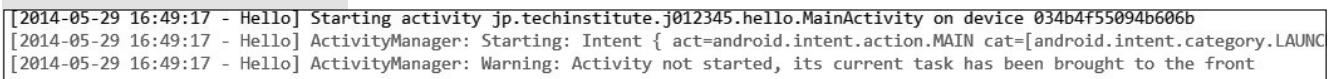
AndroidプロジェクトからAndroidアプリケーションを作成し、USB接続された端末で実行するまでの流れは、Eclipse下部のコンソールに表示されます。Android端末にアプリケーションを転送し、アプリケーションの起動に成功すると、コンソールに次のように「Success!」と表示されます(図26)。



```
[2014-05-29 16:28:05 - appcompat_v7] WARNING: unable to write jarlist cache file
[2014-05-29 16:28:05 - appcompat_v7] WARNING: unable to write jarlist cache file
[2014-05-29 16:45:57 - Hello] -----
[2014-05-29 16:45:57 - Hello] Android Launch!
[2014-05-29 16:45:57 - Hello] adb is running normally.
[2014-05-29 16:45:57 - Hello] Performing jp.techinstitute.j012345.hello.MainActivity
[2014-05-29 16:45:57 - Hello] Automatic Target Mode: using device '034b4f55094b606b'
[2014-05-29 16:45:57 - Hello] Uploading Hello.apk onto device '034b4f55094b606b'
[2014-05-29 16:45:57 - Hello] Installing Hello.apk...
[2014-05-29 16:45:59 - Hello] Success!
[2014-05-29 16:45:59 - Hello] Starting activity jp.techinstitute.j012345.hello.MainActivity
[2014-05-29 16:46:00 - Hello] ActivityManager: Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] }
[2014-05-29 16:46:26 - Hello] -----
[2014-05-29 16:46:26 - Hello] Android Launch!
```

図26:起動成功

何度かアプリケーションの実行を行ってみましょう。すると、次のように赤いエラーメッセージがコンソールに表示される場合があります(図27)。



```
[2014-05-29 16:49:17 - Hello] Starting activity jp.techinstitute.j012345.hello.MainActivity on device 034b4f55094b606b
[2014-05-29 16:49:17 - Hello] ActivityManager: Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] }
[2014-05-29 16:49:17 - Hello] ActivityManager: Warning: Activity not started, its current task has been brought to the front
```

図27:起動失敗

詳細は後述の「Activity」のライフサイクルと「LogCat」の項で説明しますが、「戻る」ボタンと「ホーム」ボタンでは動作が異なります。「戻る」ボタンを押した場合、アプリケーションは完全に終了しますが、「ホーム」ボタンを押した場合はホーム画面が表示されるだけで、アプリケーションが完全に終了したわけではないのです。そのため、「ホーム」ボタンを押してアプリケーションを終了したつもりで、再度Eclipseからアプリケーションを実行しようとすると、うまくいかない場合があるのです。

EclipseからAndroidアプリケーションを実行したときに、端末上で以前に起動したアプリケーションが実行中の場合は、Eclipseは実行中のアプリケーションを終了させ、新しいアプリケーションをインストールして実行しようとします。このとき、うまく以前に起動したアプリケーションを終了できない場合があります。

この場合、「Its current task has been brought to the front」とメッセージが表示されます。エラーメッセージが表示された場合は端末の「戻る」ボタンを利用して以前のアプリケーションを終了し(このとき「ホーム」ボタンで終了しないようにしましょう)、再度、EclipseからAndroidアプリケーションの実行を行ないます。



実行時にダイアログボックスが表示される場合

接続している端末のバージョンによっては、作成したアプリを端末で実行しようとした際に、次のようなダイアログボックスが表示される場合があります。

このダイアログが表示された場合は、リスト内に表示された端末を選択し、[OK]をクリックします。ダイアログボックスのリスト内に端末名が表示されない場合は、何度かUSBケーブルを抜き差しするとうまく認識することがあるので試してみましょう

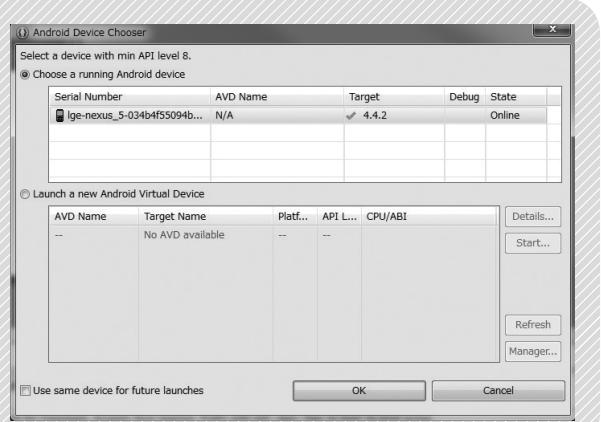


図25:デバイス選択ダイアログ



6-1-2 Android アプリケーションの基本構造

プログラムの改造1

Androidアプリケーションの基本構造を理解するために、「Hello」プロジェクトを改造してみましょう。プロジェクトを生成した際、Eclipseには次のように「activity_main.xml」というファイルが表示されていました(図28)。



図28:activity_main.xml

この画面は、アプリケーションを実行したときに表示されていた画面と同じに見えます。本当にこの画面が表示されていたのか、確認してみましょう。左側の「Palette」から「Form Widgets」内のボタン(Button)と「Time & Date」内のアナログ時計(AnalogClock)をドラッグ&ドロップして、次のように配置してみましょう(図29)。

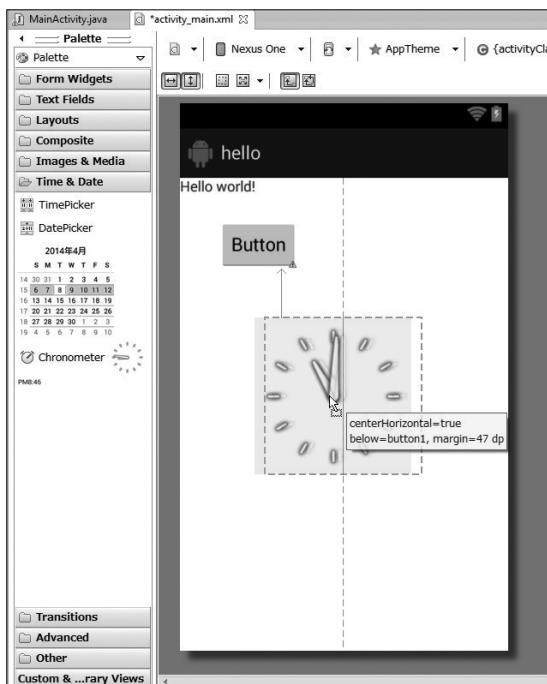


図29:ボタンと時計の配置

再度、Androidアプリケーションを実行すると、編集した画面と同じようにボタンとアナログ時計が表示されました(図30)。

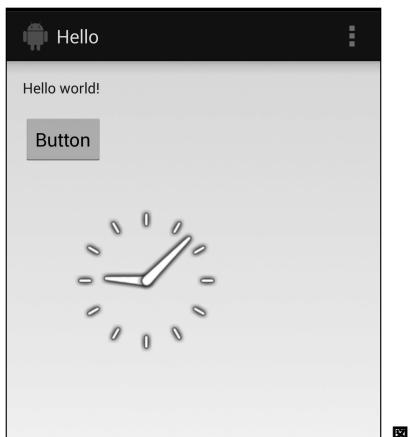


図30:実行例

どうやらAndroidアプリケーション実行時には、この画面を表示しているようです。では、この画面はどこでセットしているのでしょうか。

そのことを確認するためにPackage ExplorerからHelloプロジェクトを開き、「src」フォルダーを開き、「MainActivity.java」をダブルクリックして開いてみましょう(図31)。

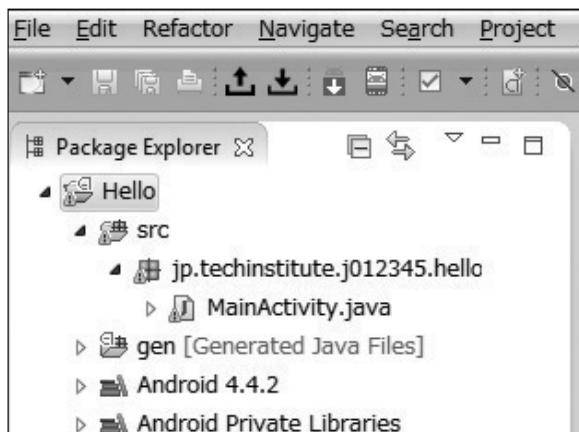


図31:ソースコードの表示

すると、次のようにJavaのプログラムが表示されます(図32)。

```
1 package jp.techinstitute.j012345.hello;
2
3 import android.app.Activity;
4
5 public class MainActivity extends Activity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        setContentView(R.layout.activity_main);
11    }
12}
```

図32:MainActivity.java

この中に、画面を定義していたファイルである「activity_main.xml」という言葉があるかを探してみましょう。activity_main.xmlという言葉は見つかりませんが、「onCreate()」メソッドの中に「setContentView(R.layout.activity_main)」と

いう記述があることがわかります。

Package Explorerを確認すると、「res」フォルダー内の「layout」フォルダーオン中に、「activity_main.xml」というファイルがあります(図33)。

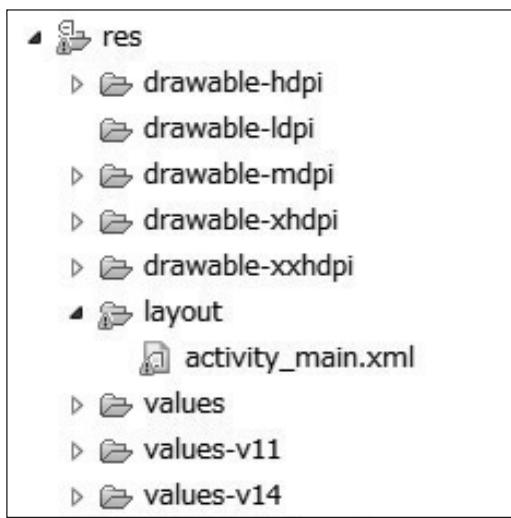


図33:activity_main.xmlの場所

以上からonCreate()は、アプリケーションが起動するときに実行されるメソッドのようで、「setContentView」は、「View」(見た目)を「Content」(アプリケーションの内容)にセットしなさいという命令ではないかと考えられます。

「R.layout」がどのようなものは後述するとして、この理解が正しいか、「Button」という部品(文字通り「ボタン」を表示)を利用して、部品だけを画面に表示できるか確認してみたいと思います。ソースコードを次のように修正します。

```
MainActivity.java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Button btn = new Button(this);
    setContentView(btn);
}
```

Buttonにはimport文が必要です。Buttonに赤波線が出た場合は、次のようにマウスカーソルをポイントして表示されるクリックフィックスより、「Import～」を選択してimport文を設定するようにしましょう(図34)。



図34:インポート文の追加

修正後のプログラム上部3行目付近の「+」にマウスカーソルをポイントすると、次のようにButtonが追加されていることがわかります(図35)。



3④ **import android.support.v7.app.ActionBarActivity;**
14 **import android.support.v7.app.ActionBar;**
15 **import android.support.v4.app.Fragment;**
16 **import android.os.Bundle;**
17 **import android.view.LayoutInflater;**
18 **import android.view.Menu;**
19 **import android.view.MenuItem;**
20 **import android.view.View;**
21 **import android.view.ViewGroup;**
22 **import android.widget.Button;**
23 **import android.os.Build;**
24
25 }
26

Press 'F2' for focus

図35:インポート文の確認



import文が必要な理由

Buttonなどの一般的な名称だと、同じ名前のプログラムがたくさん作成される可能性があります。その場合、どのButtonを利用したいのか、プログラムが判断できなくなってしまうことが想定されます。

各プログラム部品を一意に判断するための名前を、「完全修飾名」と言います。Buttonの場合は、「android.widget.Button」が

例1

```
android.widget.Button btn = new android.widget.Button(this);
```

例2

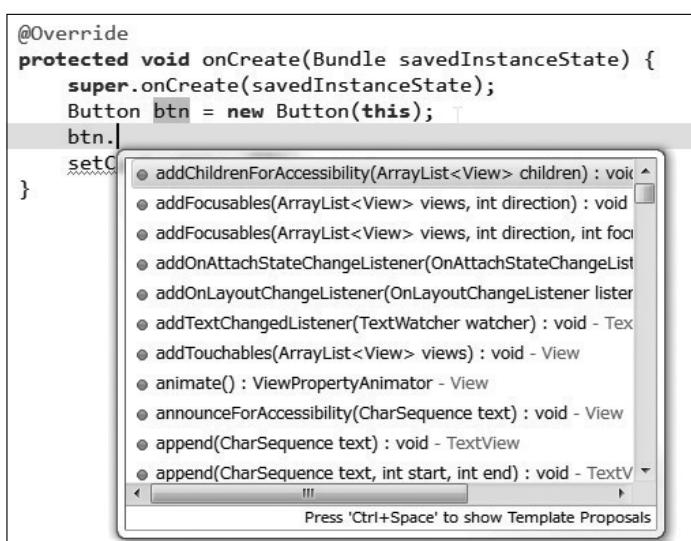
```
Button btn = new Button(this);
```

完全修飾名となります。

インポート文を記述しない場合は例1のように記述することもできますが、何度も記述する場合は面倒です。インポート文を記述することで、手間を省略して、Buttonという名前でプログラム部品を利用できるようになります(例2)。

リストが消えてしまった場合は、btnのところでCtrl+Spaceキーを押すと、再表示できます。

Buttonのインスタンスを生成する場合、このAndroidアプリケーションにボタンを貼り付けたいですから、「MainActivity」を示す「this」を引数に与えます。Buttonにはさまざまな性質や動作が設定されていて、Buttonのインスタンスであるbtnに続けて「.」(ピリオド)を入力すると、Buttonが持っている性能や機能を確認することができます(図36)。



```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    Button btn = new Button(this);  
    btn.  
    setC  
}  
setC  
addChildrenForAccessibility(ArrayList<View> children) : void  
addFocusables(ArrayList<View> views, int direction) : void  
addFocusables(ArrayList<View> views, int direction, int focusSearchFlags) : void  
addOnAttachStateChangeListener(OnAttachStateChangeListener listener) : void  
addOnLayoutChangeListener(OnLayoutChangeListener listener) : void  
addTextChangedListener(TextWatcher watcher) : void - TextView  
addTouchables(ArrayList<View> views) : void - View  
animate() : ViewPropertyAnimator - View  
announceForAccessibility(CharSequence text) : void - View  
append(CharSequence text) : void - TextView  
append(CharSequence text, int start, int end) : void - TextView
```

Press 'Ctrl+Space' to show Template Proposals

図36:Buttonの性質や機能

Buttonにはたくさん性質や機能が存在しますが、ここでは「setText()」を利用してボタンのラベルを設定してみましょう。ソースコードに次のようなコードを追加します。

MainActivity.java

```
～略～  
    Button btn = new Button(this);  
    btn.setText("Hello");// 追加  
    setContentView(btn);  
～略～
```

編集したMainActivity.javaを保存し、実行してみましょう(図37)。

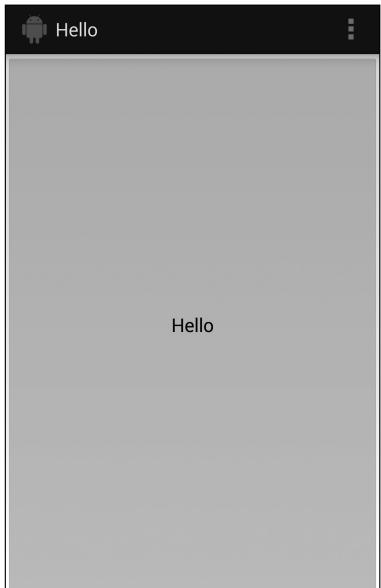


図37:ボタンの表示

画面一杯にボタンが表示されるので、ボタンのように見えないかもしれません。押すと色が変わることから、これがボタンであることがわかります。

しかし、ボタンを押しても何の反応もありません。「ボタンを押したら○○する」といった動作が、何もプログラミングされていないためです。このような、「△△したら○○する」といったプログラムを、「イベント駆動型のプログラム」といいます。イベント駆動型のプログラムを記述するには、「リスナーインターフェース」を利用します。

インターフェースを利用したイベントの記述の方法には、いくつかのパターンがあります。ここでは、Androidアプリケーション開発で比較的よく利用される「匿名インターフェース」を利用した記述で、ボタンを押したときに動作するイベント駆動型のプログラムを記述してみましょう。

とはいっても、初めてAndroidを勉強する人には難しい話なので、形としては覚えていきましょう。次のようにイベントを設定します。

1. 「MainActivity.java」の「btn.setText("Hello");」の次の行に、「btn.」とピリオドまで打ち込む。するとドロップダウンリストが表示されます(図38)。

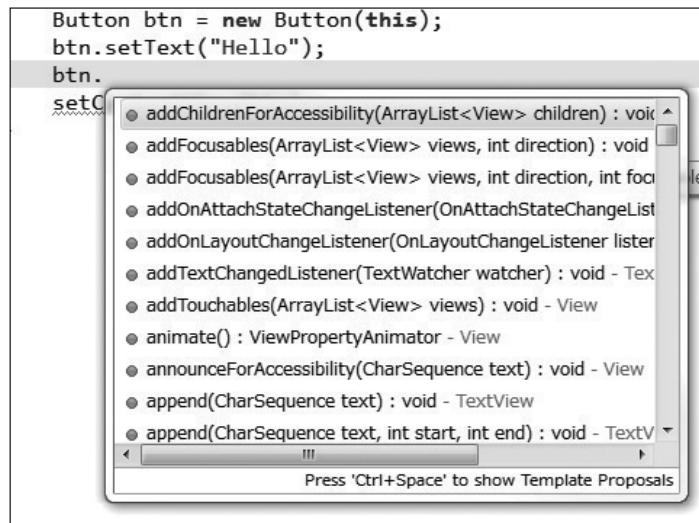


図38: ドロップダウンリストの表示

2. そのまま「seton」まで打ち込み、エンターキーを押します。すると「btn.
setOnTouchListener(l);」と入力されます。

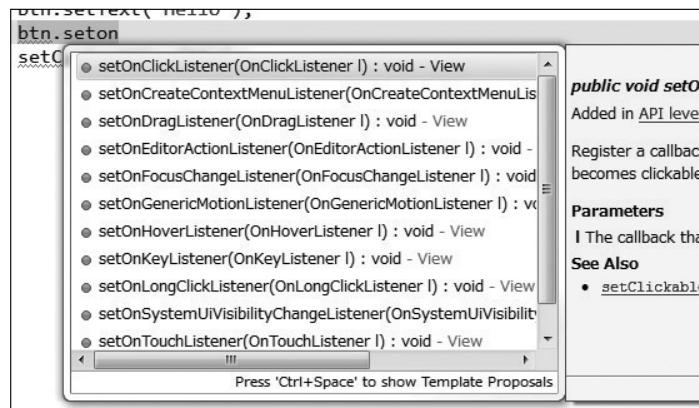


図39: setOnTouchListener

MainActivity.java
btn.setOnClickListener(l);

3. 次に()の中の「l」を範囲選択し、リスナー(監視者)のインスタンスを設定するため、「new View.」と入力します。すると、次のようなドロップダウンリストが表示されるので、一番先頭に表示されている「View.OnClickListener()」を選択し、エンターキーで確定します(図40)。

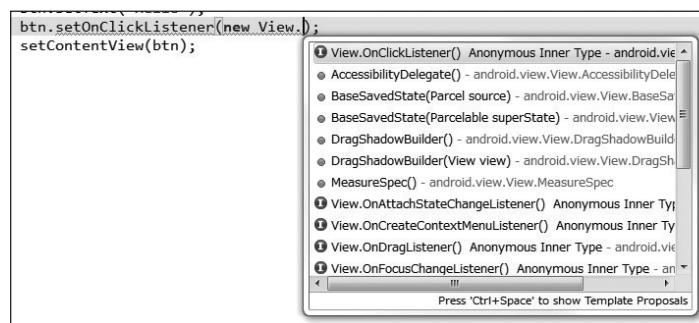


図40: リスナーの設定

MainActivity.java

```
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
    }
});
```

setOnClickListenerは、クリックしたときのリスナー（監視者）をセットするための命令で、()の中にリスナー（監視者）のインスタンスを設定します。

4.この中の「onClick()」メソッドが、ボタンが押されたときに実行される場所になります。引数の「View v」が画面上で押された対象になります。

慣れるまではこの記述を難しく感じると思いまが、頻繁に出てくる記述方法なので、形で覚えてしまいましょう。また、すべて自分で入力せずに、Eclipseのコード補完機能を利用して入力するようにしましょう。

さて、ボタンを押すとonClick()メソッドの中が実行されるわけですから、onClick()メソッドの中を次のように変更し、ボタンを押したときにラベルの表示が変わるようにしてみましょう。

MainActivity.java

```
@Override
public void onClick(View v) {
    Button b = (Button) v;
    b.setText("こんにちは");
}
```

ここで、「View v」の「v」がユーザーに指で押される対象になります。Androidでは、画面に配置できる部品を「View」（ビュー）というクラスで表現します。画面に配置した部品は、Buttonだけでなくアナログ時計など、さまざまな部品があります。これらを総称してViewと呼んでいます。押されたものはonClick()メソッドの引数として渡ってきますので、次のように記述して、Buttonに戻します。

記述例

`Button b = (Button) 「ここに onClick() メソッドの引数名を入れる」`

このような、型を変換する記述を「キャスト」といいます。ここまでできたら、一度動作を確認してみましょう。アプリケーションを実行し、画面いっぱいに表示されたボタンをタップすると「Hello」が「こんにちは」に変わることを確認しましょう（図41）

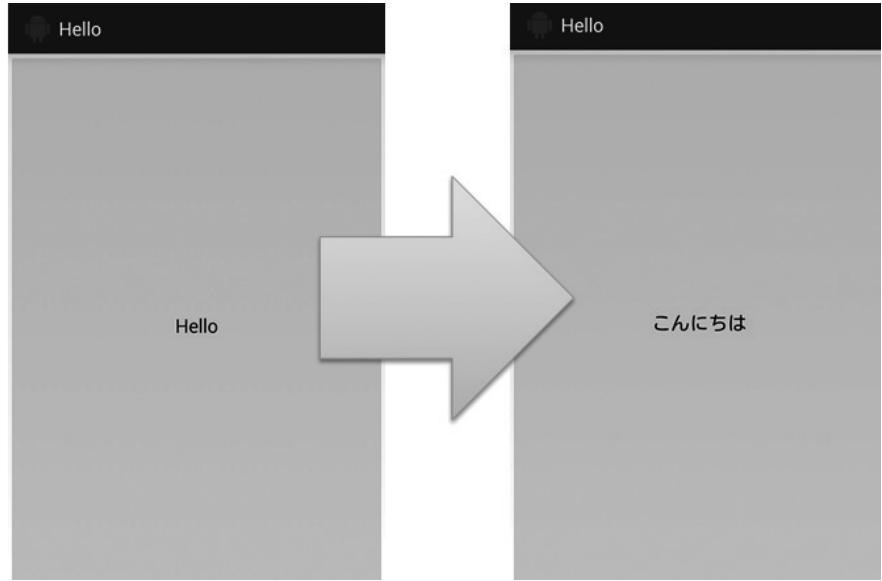


図41:改造1の実行結果

確認できたら、プログラムを元の形に戻しておきましょう。

MainActivity.java

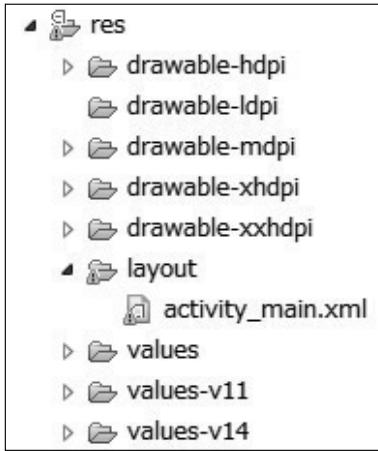
```
～略～  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
～略～
```

プログラムの改造2

プログラムの改造1では、Buttonのインスタンスを作成して画面に貼り付けました。このように、プログラムだけで画面を作成することもできますが、複雑な画面を作成するのは大変です。また、国際的なアプリケーションを作成する場合、Buttonのラベルをそれぞれの国の言語に変更する処理を入れる必要があり、処理が複雑になってします。

そこでAndroidプロジェクトでは、画面の定義をプログラムから切り離して記述できるようになっています。この、プログラムから切り離した画面の定義や音楽ファイル、画像など、プログラム以外のものを「リソース」と呼びます。Androidプロジェクトでは、リソースをresフォルダーの中に格納することになっています(図42)。

図42:resフォルダーの中身



リソースを格納するフォルダーは「カテゴリ名-オプション名」といった形で定義します。よく使うリソースフォルダーは、プロジェクト生成時に自動で生成されます。

フォルダーネ名	内 容
animator/	アニメーションプロパティーを定義したXMLファイルを格納します。
anim/	連続的に変化するアニメーションを定義したXMLファイルを格納します。
color/	色を定義したXMLファイルを格納します。
drawable/	画像ファイルなどクラスBitmapで扱えるファイルを格納します。
layout/	画面の見た目を定義するXMLファイルを格納します。
menu/	メニューの項目を定義するXMLファイルを格納します。
raw/	音楽ファイルなどファイルストリームとして利用するファイルを格納します。
values/	名前と値を定義したXMLファイルを格納します。
xml/	その他プログラム等で利用したいXMLファイルを格納します。

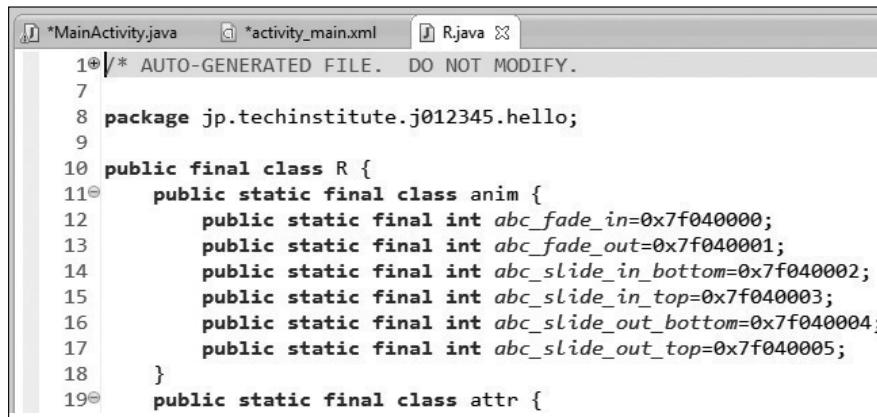
表2:リソースフォルダーの種類

resフォルダー内に格納するファイル名には、半角アルファベット、数字、アンダーバーとピリオドしか利用できません。この決まりに従わないファイルが配置された場合は、エラーとなります(図43)。

```
[2014-06-09 10:02:03 - Hello] Error in an XML file: aborting build.  
[2014-06-09 10:03:41 - Hello] res\layout\A.xml: Invalid file name: must contain only [a-z0-9_.]  
[2014-06-09 10:03:41 - Hello] C:\Users\USER\workspace\Hello\res\layout\A.xml:1: error: Error parsing XML: no element found
```

図43:命名規則違反

プログラムからリソースにアクセスする場合は、どのようにすればいいのでしょうか。たとえば音楽ファイルを再生するなど、resフォルダー内に格納したリソースを、プログラムの中で取得して操作する場合があります。Androidではリソースへのアクセスを簡単にするために、「R」というクラスが用意されています。Rクラスはgenフォルダー内に自動生成されます。



```
*MainActivity.java *activity_main.xml R.java
1 * AUTO-GENERATED FILE. DO NOT MODIFY.
2
3 package jp.techinstitute.j012345.hello;
4
5 public final class R {
6     public static final class anim {
7         public static final int abc_fade_in=0x7f040000;
8         public static final int abc_fade_out=0x7f040001;
9         public static final int abc_slide_in_bottom=0x7f040002;
10        public static final int abc_slide_in_top=0x7f040003;
11        public static final int abc_slide_out_bottom=0x7f040004;
12        public static final int abc_slide_out_top=0x7f040005;
13    }
14    public static final class attr {
```

図44:自動生成されたRクラス

resフォルダー内にファイルが追加、修正されると、自動的にRクラスが更新されて、Activity等のプログラム内でリソースが利用できるようになります。改造前の「setContentView()」が、「setContentView(R.layout.activity_main)」と書かれていたのは、Rクラスを利用してresフォルダー内のlayoutフォルダーに格納された「activity_main.xml」にアクセスしているということになります(図45)。



図45:Rとリソースの関係

resフォルダー内にエラーが発生すると、Rクラスが自動生成されず、Activityがコンパイルエラーになる場合があります。確認のために、activity_main.xmlを右クリックし、「Refactor」から「Rename」を選択して、ファイル名を「Activity_main.xml」に変更してみましょう。

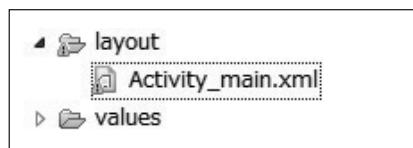


図46:ファイル名をActivity_main.xmlに変更

すると、Rが生成されずに消えてしまい、MainActivity.javaに赤いバツ印がつようになります(図47)。

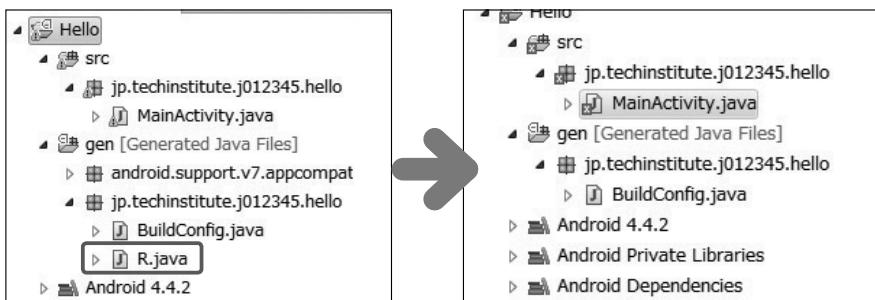


図47:Rが自動生成されず、genフォルダーからRが消えてしまう

そのため、MainActivity内の記述が間違っていない場合でも、コンパイルエラーになってしまふので注意しましょう。

なお、以前のRが残っている場合は、「Project」メニューから「Clean」を選択します(図48)。

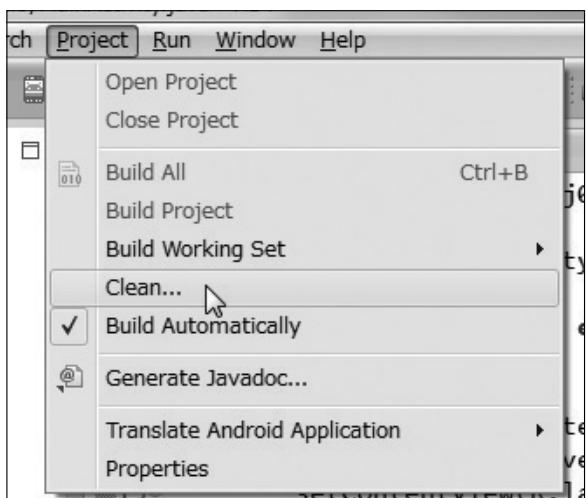


図48:プロジェクトのClean

その後、表示されたダイアログボックスで「OK」をクリックしましょう(図49)。

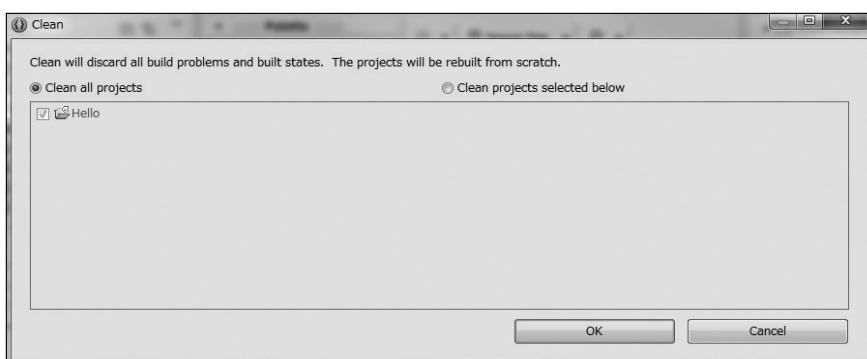


図49:Cleanの実行

Cleanを実行することで、プロジェクトをいちから再構成します。今回はエラーを確認するために利用しましたが、resフォルダー内のエラーを修正してもRが自動生成されないときは、Cleanを実行すると直る場合があるので覚えておきましょう。

確認ができたら、同じ手順でActivity_main.xmlの名称を変更してactivity_main.xmlに戻し、Activityの記述を修正しましょう。

MainActivity.java

```
setContentView(R.layout.Activity_main); // 変更前  
↓  
setContentView(R.layout.activity_main); // 変更後
```

```
>MainActivity.java [x] activity_main.xml  
1 package jp.techinstitute.j012345.hello;  
2  
3 import android.app.Activity;  
4  
5 public class MainActivity extends Activity {  
6  
7     @Override  
8     protected void onCreate(Bundle savedInstanceState) {  
9         super.onCreate(savedInstanceState);  
10        setContentView(R.layout.activity_main);  
11    }  
12}  
13  
14
```

図50:確認後のソースコードの修正

では、画面レイアウトを表現していたactivity_main.xmlを確認してみましょう。先ほど追加したボタンとアナログ時計が配置されています。下部のactivity_main.xmlタブをクリックすると、activity_main.xmlの内部を見ることができます。

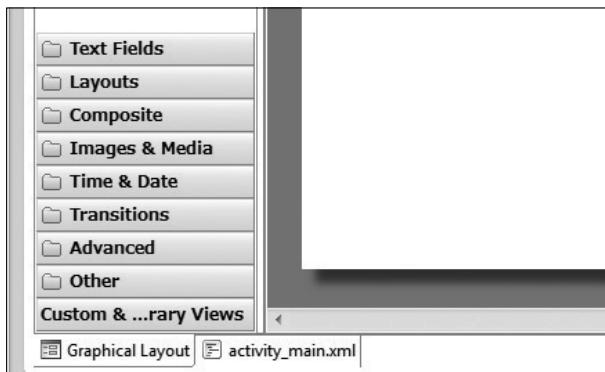


図51:activity_mainの下部のタブ

```
1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
2     xmlns:tools="http://schemas.android.com/tools"  
3     android:layout_width="match_parent"  
4     android:layout_height="match_parent"  
5     android:paddingBottom="@dimen/activity_vertical_margin"  
6     android:paddingLeft="@dimen/activity_horizontal_margin"  
7     android:paddingRight="@dimen/activity_horizontal_margin"  
8     android:paddingTop="@dimen/activity_vertical_margin"  
9     tools:context="jp.techinstitute.j012345.hello.MainActivity$PlaceholderFragment" >  
10  
11     <TextView  
12         android:id="@+id/textView1"  
13         android:layout_width="wrap_content"  
14         android:layout_height="wrap_content"  
15         android:text="@string/hello_world" />  
16  
17     <Button  
18         android:id="@+id/button1"  
19         android:layout_width="wrap_content"  
20         android:layout_height="wrap_content"  
21         android:layout_alignLeft="@+id/textView1"  
22         android:layout_below="@+id/textView1"  
23         android:layout_marginTop="16dp"  
24         android:text="Button" />  
25  
26     <AnalogClock  
27         android:id="@+id/analogClock1"  
28         android:layout_width="wrap_content"  
29         android:layout_height="wrap_content"  
30         android:layout_alignLeft="@+id/button1"
```

図52:activity_main.xml

画面レイアウトのファイルは、「XML」(eXtensible Markup Language)という形式で記述されています。図53が、配置した部品のIDを示しています。

```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/textView1"
    android:layout_below="@+id/textView1"
    android:layout_marginTop="16dp"
    android:text="Button" />

<AnalogClock
    android:id="@+id/analogClock1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/button1"
    android:layout_below="@+id/button1"
    android:layout_marginLeft="42dp"
    android:layout_marginTop="35dp" />
```

図53:IDの確認

activity_main.xmlのButtonのIDが「`android:id="@+id/button1"`」である場合は、Activityからは「`findViewById`」メソッドを利用して、「`R.id.button1`」といった記述で画面に配置したButtonにアクセスすることができます。

それでは、Rを利用した形式にプログラムを改造してみましょう。MainActivity.javaの`onCreate()`メソッドを、次のように修正します。

```
MainActivity.java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Button btn = (Button) findViewById(R.id.button1);
    btn.setText("Hello");
    btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View arg0) {
            Button b = (Button) arg0;
            b.setText("こんにちは");
        }
    });
}
```

アプリケーションを実行すると、次のように表示されます(図54)。

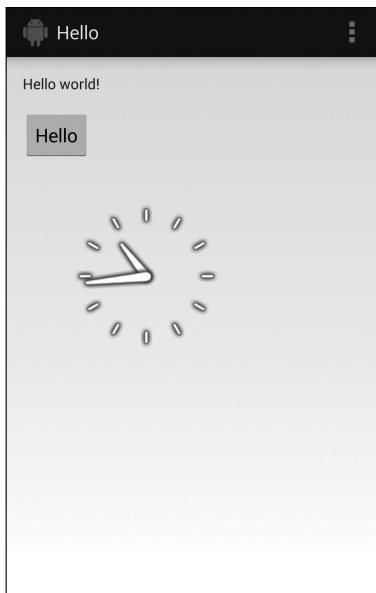


図54:改造したアプリケーション

そして、ボタンをクリックすると、ラベルが「Hello」から「こんにちは」に切り替わることが確認できるでしょう。。

まとめ

1. Androidのプログラムは起動時にActivityのonCreate()メソッドが実行される
2. onCreate()メソッドの中で、setContentView()を実行すると画面の見た目(View)をセットすることができる
3. 画面はactivity_main.xmlといった外部ファイルでも作成することができるし、プログラムでも作成することができる。
4. 画面に配置したボタンなどのViewに対しイベントに対応したプログラムを作成することができる。

6-1で、「Button」をタッチしたときにボタンのラベルを変更するアプリケーションを作成しました。ここでは、ボタンをタッチしたときに音楽ファイルを再生するように改造してみます。



6-2-1 音楽ファイルの再生

ここでは、前説で作成したアプリケーションを使用し、次のことを学びます。

- ①Androidで音楽ファイルを再生する方法
- ②Activityのライフサイクル
- ③ログの出力と見方

AIPを使ってカンタンに音楽を再生

Androidでは、「MediaPlayer」と呼ばれるクラスが用意されています。MediaPlayerを利用すると、MP3などの音楽ファイルを簡単に扱うことができます。このような便利な機能を提供してくれるクラスのことを「API」(Application Programming Interface)といいます。

まず準備として、音楽ファイルを用意しましょう。自作の音楽ファイル以外の音楽ファイルを利用する場合は、著作権やライセンスに十分注意しましょう。

用意した MP3ファイルは、ひとまずmusic.mp3という名前でディスクトップなどに配置しましょう。音楽ファイルはresフォルダーの「raw」というカテゴリのフォルダー内に格納することになっていますが、rawフォルダーは自動生成されないので独自に作る必要があります。次のようにresフォルダーを右クリックし、「New」→「Folder」を選択します(図1)。

すると、次のように「New Android Application」のダイアログボックスが表示されるので、「Folder」に「raw」と入力して「Finish」をクリックします(図2)。

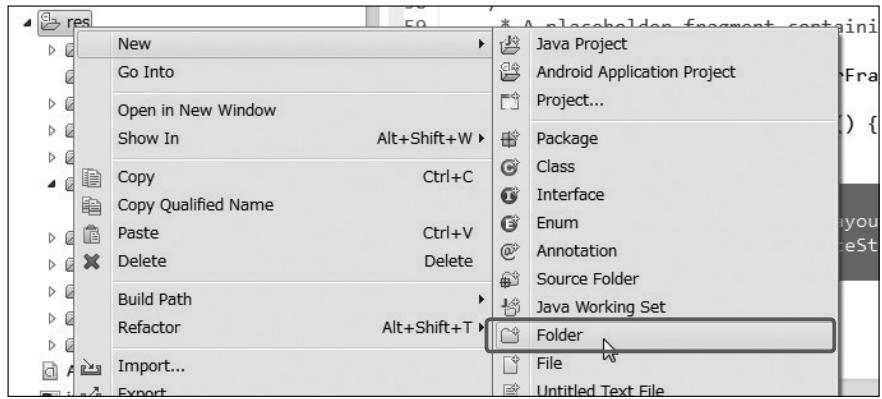


図1:新規フォルダーの作成

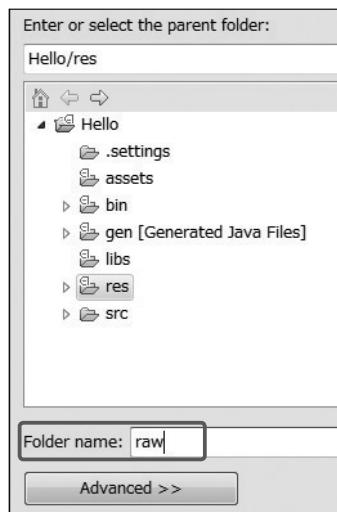


図2:新規フォルダーダイアログ

作成したrawフォルダーに、デスクトップに保存したmusic.mp3をドラッグ&ドロップで配置しましょう(図3)。



図3:音楽ファイルの配置

オプションのダイアログボックスが表示されますが、ファイルをコピーのま「ok」をクリックします(図4)。

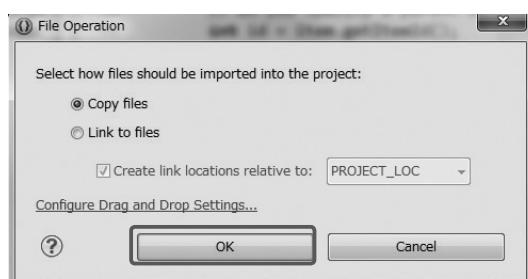


図4:ファイル操作ダイアログ

MediaPlayerを利用するため、フィールド変数「mPlayer」を次のように定義します。

MainActivity.java

```
public class MainActivity extends ActionBarActivity {
    MediaPlayer mPlayer;
~略~
```

次に、Buttonを押したときに音楽が再生されるように、処理を追加します。

MainActivity.java

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Button btn = (Button) findViewById(R.id.button1);
    btn.setText("Hello");
    mPlayer = MediaPlayer.create(this, R.raw.music); // 追加①
    btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View arg0) {
            Button b = (Button) arg0;
            b.setText("こんにちは");
            mPlayer.start(); // 追加②
        }
    });
~略~
```

MediaPlayerのインスタンスを取得する場合は、MediaPlayerの「Create()」メソッドを利用します。第1引数に、対象の音楽を再生するこのアプリケーションを表す「this」を指定し、第2引数にRクラスを利用して、rawフォルダーに格納したmusic.mp3を「R.raw.music」のように指定します。

MediaPlayerのインスタンスを生成することができたら、「start()」メソッドを利用して音楽ファイルを再生することができます。

プログラムを実行し、音楽ファイルを再生してみましょう。うまく音楽が再生されましたか？

このように、Androidアプリケーションでは豊富に用意されたプログラム部品であるAPIを利用し、比較的簡単に高度な機能を実現することができます。



6-2-2 Activity のライフサイクルと LogCat

ただし、今作成したアプリケーションは、音楽ファイルを再生することができますが、止める機能がありません。音楽ファイル再生中に、試しに戻るボタンや、ホームボタンを押してみましょう。音楽は再生されたままの状態が続き、電話がかかってきても、音楽ファイルが再生されたままの状況です。

そこで、戻るボタンやホームボタンを押したときに、音楽が止まるような処理を入れていきましょう。Androidでは、`onCreate()`はアプリケーション(Activity)が起動するときに実行されるというものです。同様に、あるタイミングで実行されるメソッドが、次のように定義されています。

メソッド名	内 容
<code>onCreate()</code>	Activity起動時に呼び出されます
<code>onStart()</code>	<code>onCreate</code> の後もしくはリストアの後に呼び出されます
<code>onResume()</code>	Activityが電話の着信後などポーズ状態から復帰してフォアグラウンドになるに呼び出されます
<code>onPause()</code>	他のActivityが起動する場合や自分自身が終了するときに呼び出されます
<code>onStop()</code>	自身のActivityが長い間バックグラウンドにいると呼び出されます
<code>onDestroy()</code>	自身のActivityが終了するときに呼び出されます
<code>onRestart()</code>	バックグラウンドから復帰するときに呼び出されます

表1: ライフサイクルに対応したメソッド名

Activityが起動してから終了するまでに、これらのメソッドが呼び出されます。この起動から終了までの流れを、「Activityのライフサイクル」といいます。

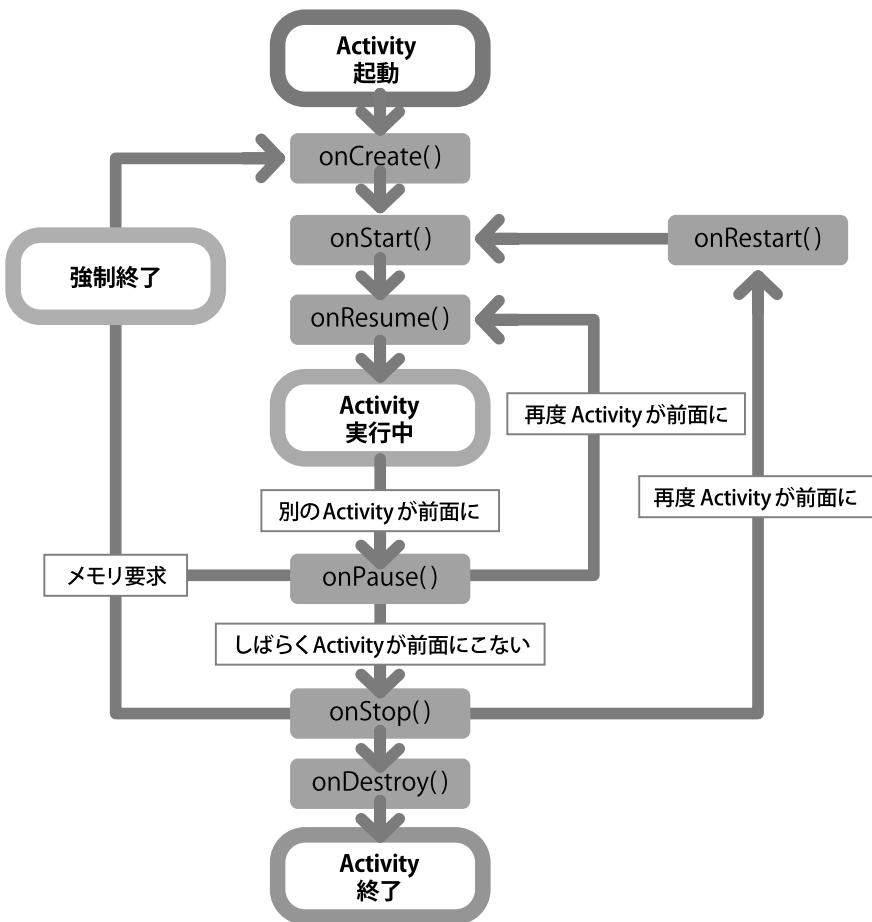


図5:Activityのライフサイクル

戻るボタンやホームボタンを押したとき、電話の着信があったときなどには、ライフサイクルの流れに従って、これらのメソッドが動作します。動作を確認するために、これらのメソッドをオーバーライド(再定義)してみましょう。メソッドをオーバーライドするには「Source」メニューより「Override/Implements Method…」を選択します(図6)。

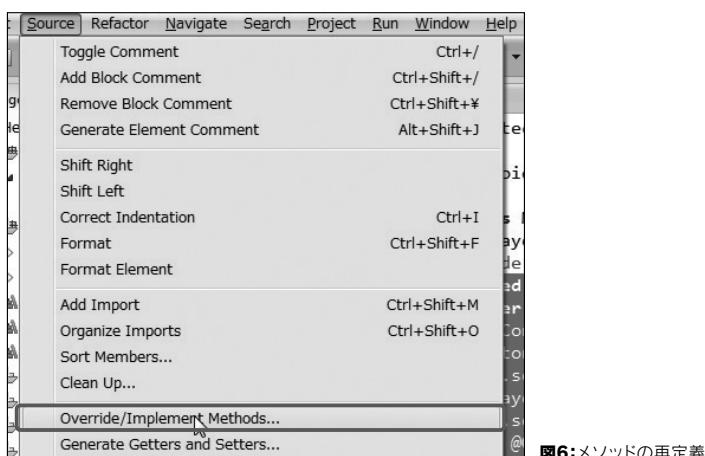


図6:メソッドの再定義

すると、次のようなダイアログボックスが表示されます(図7)。

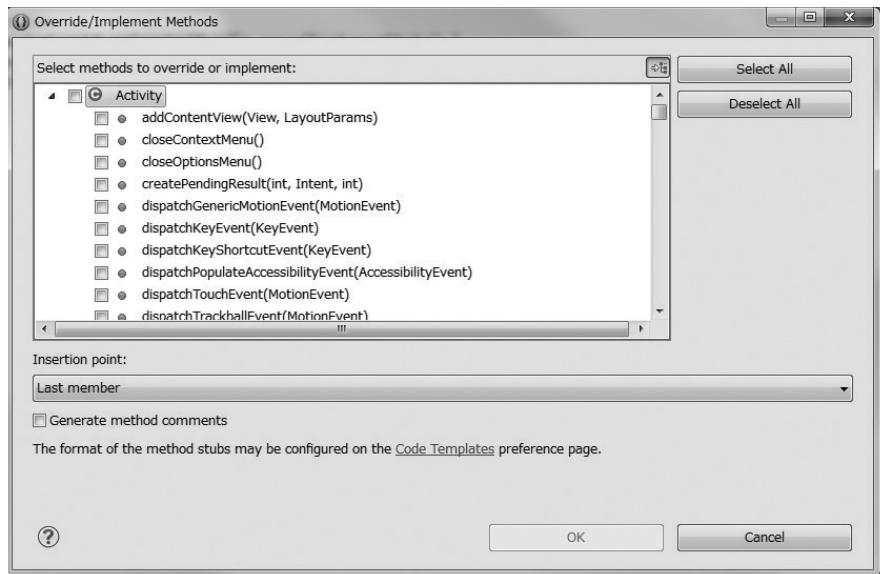


図7:メソッド再定義のダイアログ

「onDestroy」、「onPause」、「onRestart」、「onResume」、「onStop」、「onStart」に、順にチェックを入れ、「Last member」を選択し、OKボタンをクリックします。すると、次のようにメソッドが自動挿入されます(図8)。

```

76@    @Override
77     protected void onStop() {
78         // TODO Auto-generated method stub
79         super.onStop();
80     }
81
82@    @Override
83     protected void onDestroy() {
84         // TODO Auto-generated method stub
85         super.onDestroy();
86     }
87
88@    @Override
89     protected void onPause() {
90         // TODO Auto-generated method stub
91         super.onPause();
92     }
93
94@    @Override
95     protected void onResume() {
96         // TODO Auto-generated method stub
97         super.onResume();
98     }
99
100@   @Override
101     protected void onStart() {
102         // TODO Auto-generated method stub
103         super.onStart();
104     }

```

図8:オーバーライドされたメソッド

定義されたメソッドの中に、ボタンのラベルを変更するなどの処理を入れれば、動作を確認することができそうですが、「onDestroy」メソッドなど、アプリケーション終了時に実行されるメソッドなどは、画面上で確認するのは難しそうです。

Androidには、プログラム中の任意の位置でメッセージを出力できる機能があります。この機能のことを「ログ」といいます。ログを利用して、メソッドが実行されるかどうかを確認してみましょう。Androidでは次のようにログを記述します。

Log.d(タグ名, 出力するメッセージ);

「Log」にはインポート文が必要です。Logに赤波線が引かれたら、マウスカーソルをポイントしてインポートを選択しましょう。

「d」はデバッグという意味で、出力のレベルを決める部分です。d以外に次のよ

うにレベルが存在します。

出力レベル	意味
d	デバッグルベルのログを出力する
e	エラーレベルのログを出力する
i	情報レベルのログを出力する
w	警告レベルのログを出力する
v	上記に当てはまらないログを出力する

表2:ログの出力レベル

タグ名と書かれている第1引数は、ログにつける目印で、任意の文字列を設定できます。ログは自分が作成したアプリケーション以外に、端末にインストールされているさまざまなアプリケーションから出力されます。タグを利用してすることで、自分が作成したアプリケーションのログを絞り込むことができます。



変数の出力

出力するメッセージは文字列で指定します。int型の整数の変数の中身などを出力したい場合は、""+iといった形で、文字列の連結形式にして出力するようにします。

ここでは単純に、"onPauseが実行されました"などのメッセージを出力するようにしていきます(図9)。

```

77@   @Override
78    protected void onStop() {
79      super.onStop();
80      Log.d("TAG", "onStopが実行されました");
81    }
82
83@   @Override
84    protected void onDestroy() {
85      super.onDestroy();
86      Log.d("TAG", "onDestroyが実行されました");
87    }
88
89@   @Override
90    protected void onPause() {
91      super.onPause();
92      Log.d("TAG", "onPauseが実行されました");
93    }
94
95@   @Override
96    protected void onResume() {
97      super.onResume();
98      Log.d("TAG", "onResumeが実行されました");
99    }
00
01@   @Override
02    protected void onStart() {
03      super.onStart();
04      Log.d("TAG", "onStartが実行されました");
05    }

```

図9:ログの出力

忘れずにonCreate()にもLogの出力を入れるようにしましょう(図10)。

```
19  * @Override  
20  protected void onCreate(Bundle savedInstanceState) {  
21      super.onCreate(savedInstanceState);  
22      Log.d("TAG", "onCreateが実行されました");  
23      setContentView(R.layout.activity_main);  
24      Button btn = (Button) findViewById(R.id.button1);  
25      btn.setText("Hello");  
26      mPlayer = MediaPlayer.create(this, R.raw.music);  
27      btn.setOnClickListener(new View.OnClickListener() {
```

図10:onCreate()へのログの追加

ログの出力を確認するには「LogCat」というツールを利用します。Eclipseの「Window」メニューから「Show View」→「Other…」を選択します(図11)。

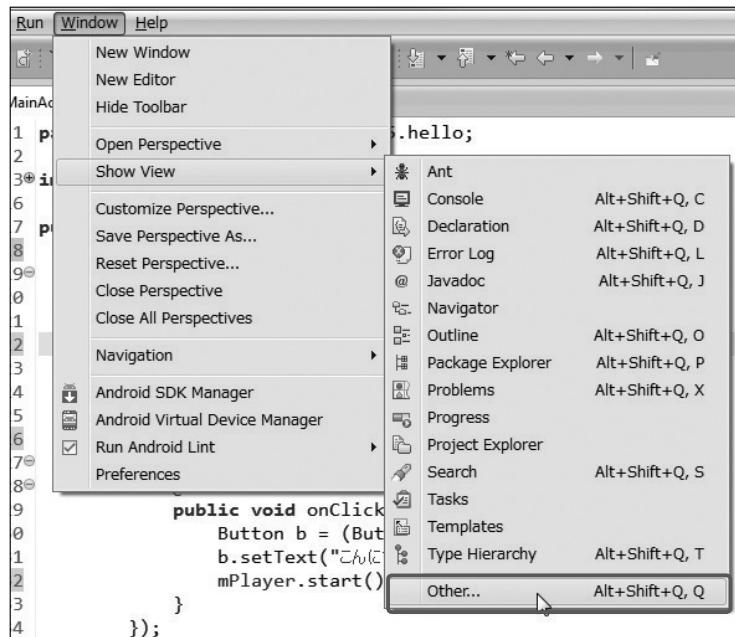


図11:VLogCat
を探すメニュー

すると、次のようなダイアログボックスが表示されます(図12)。

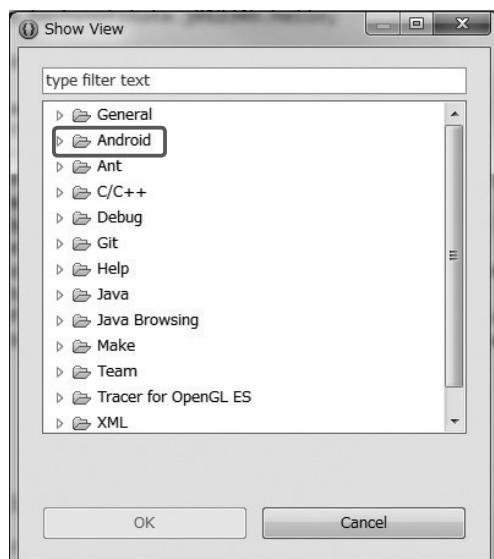


図12:Viewのダイアログボックス

Andriodフォルダーの中のLogCatを選択し、OKをクリックします(図13)。

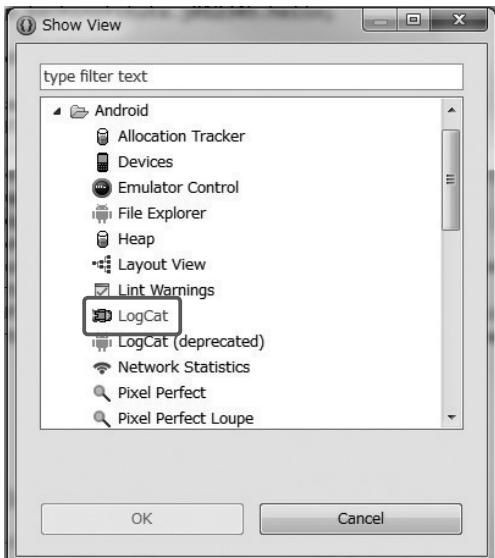


図13:LogCatの指定

すると、コンソールが表示されているところにLogCatのタブが追加され、端末から出力されるログを確認することができます(図14)。左側のSaved Filtersのところでは出力を切り替えることができ、TAGをクリックすると出力を絞り込むことができます。

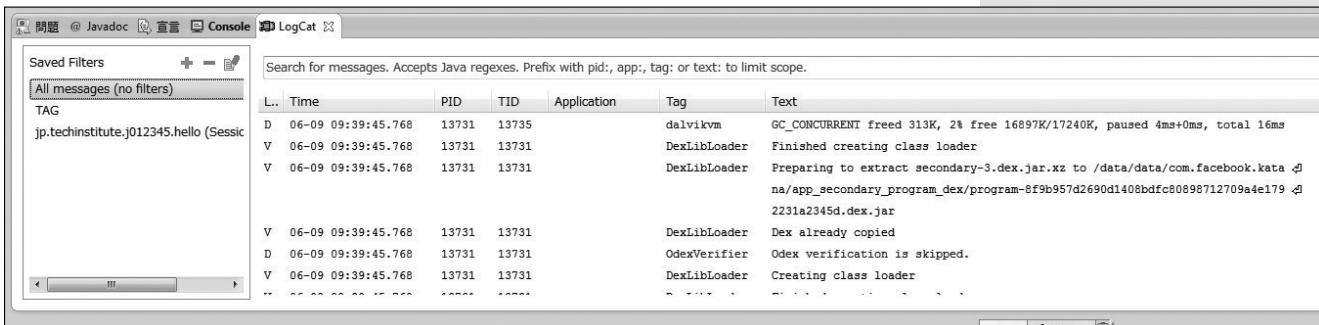


図14:LogCat

それでは、アプリケーション起動時のログを確認してみましょう。アプリケーション起動時は次のように表示されます(図15)。

L..	Time	PID	TID	Application	Tag	Text
D	06-09 11:35:07.755	22206	22206	jp.techinst...	TAG	onCreateが実行されました
D	06-09 11:35:07.805	22206	22206	jp.techinst...	TAG	onStartが実行されました
D	06-09 11:35:07.805	22206	22206	jp.techinst...	TAG	onResumeが実行されました

図15:起動時のログ

onCreate→onStart→onResumeと実行されていることがわかります。次に、戻るボタンを押してみましょう(図16)。

D	06-09 11:36:25.685	22206	22206	jp.techinst...	TAG	onPauseが実行されました
D	06-09 11:36:26.175	22206	22206	jp.techinst...	TAG	onStopが実行されました
D	06-09 11:36:26.175	22206	22206	jp.techinst...	TAG	onDestroyが実行されました

図16:戻るボタンを押した時

戻るボタンを押した場合はonPauseからonDestroyまで実行されてアプリケーションが終了していることがわかります。再度アプリケーションを立ち上げホームボタンを押してみましょう(図17)。

L..	Time	PID	TID	Application	Tag	Text
D	06-09 11:37:59.245	22206	22206	jp.techinst...	TAG	onPauseが実行されました
D	06-09 11:37:59.735	22206	22206	jp.techinst...	TAG	onStopが実行されました

図17:ホームボタンを押した時

ホームボタンを押してもアプリケーションは完全には終了せずに、onStopまで実行されて、onDestroyが実行されていないことがわかります。

この状態で再度、アプリケーションランチャーよりアプリケーションを実行すると、次のように表示され、アプリケーションが再開されていることがわかります。

D	06-09 11:39:17.785	22206	22206	jp.techinst...	TAG	onRestartが実行されました
D	06-09 11:39:17.785	22206	22206	jp.techinst...	TAG	onStartが実行されました
D	06-09 11:39:17.785	22206	22206	jp.techinst...	TAG	onResumeが実行されました

図18:ホームボタンをおした後に再度起動

このように、動作によってはonDestroyまで進まない場合があります。そのため、音楽ファイルを止める処理をonDestroyに入れると、音が止まらない場合が出てきます。どのような場合でも音楽の再生を止めるには、終了過程で必ず実行されるonPause()に音楽ファイルを止める処理を入れましょう。音楽ファイルを止めるには、MediaPlayerのstop()メソッドを記述すればよく、onPause()メソッドを次のように記述します。

```
MainActivity.java
@Override
protected void onPause() {
    super.onPause();
    mPlayer.stop();
    Log.d("TAG", "onPause が実行されました ");
}
```

実際にアプリケーションを実行して、動作を確認してみましょう。



6-2-3 例外のLog出力と確認

ところで、プログラミング上の記述ミスはEclipseで赤破線や赤丸で表示され、すぐに修正可能ですが、アプリケーション実行時のエラーはどのように確認すればよいのでしょうか。実際に試してみるために、ソースを次のように修正し、アプリケーション実行時にエラーとなるようにしてみましょう。

MainActivity.java

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    Log.d("TAG", "onCreateが実行されました");  
    setContentView(R.layout.activity_main);  
    Button btn = (Button) findViewById(R.id.button1);  
    btn.setText("Hello");  
    //mPlayer = MediaPlayer.create(this, R.raw.music); //コメントアウト  
    btn.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View arg0) {  
            Button b = (Button) arg0;  
            b.setText("こんにちは");  
            mPlayer.start();  
        }  
    });  
~略~
```

ここでは、MediaPlayerを生成せずに音楽ファイルを再生しようとしています。そのため、アプリケーションを実行してボタンをタップすると、異常終了のダイアログボックスが表示されます。このような実行時のエラーのことを、Javaでは「例外」(Exception)といいます(図19)。

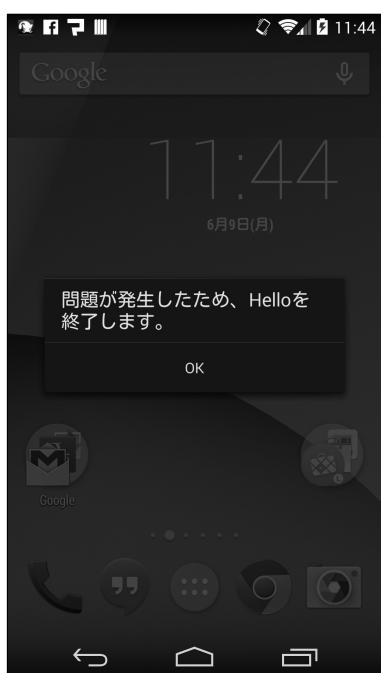


図19:例外時のダイアログボックス

このような例外が発生した場合は、例外の内容や原因がLogCatに次のように表示されます。LogCatのログを見て、例外の原因を確かめるクセをつけるようにしましょう（図20）。

```
D 06-09 11:44:16.225 24191 24191 jp.techinst... AndroidRun... Shutting down VM
W 06-09 11:44:16.225 24191 24191 jp.techinst... dalvikvm     threadid=1: thread exiting with uncaught exception (group=0x415a7ba8)
E 06-09 11:44:16.225 24191 24191 jp.techinst... AndroidRun... FATAL EXCEPTION: main
E 06-09 11:44:16.225 24191 24191 jp.techinst... AndroidRun... Process: jp.techinstitute.j012345.hello, PID: 24191
E 06-09 11:44:16.225 24191 24191 jp.techinst... AndroidRun... java.lang.NullPointerException A
E 06-09 11:44:16.225 24191 24191 jp.techinst... AndroidRun... at jp.techinstitute.j012345.hello.MainActivity$1.onClick(MainActivity.java :32) B
E 06-09 11:44:16.225 24191 24191 jp.techinst... AndroidRun... at android.view.View.performClick(View.java:4438)
E 06-09 11:44:16.225 24191 24191 jp.techinst... AndroidRun... at android.view.View$PerformClick.run(View.java:18422)
E 06-09 11:44:16.225 24191 24191 jp.techinst... AndroidRun... at android.os.Handler.handleCallback(Handler.java:733)
E 06-09 11:44:16.225 24191 24191 jp.techinst... AndroidRun... at android.os.Handler.dispatchMessage(Handler.java:95)
E 06-09 11:44:16.225 24191 24191 jp.techinst... AndroidRun... at android.os.Looper.loop(Looper.java:136)
E 06-09 11:44:16.225 24191 24191 jp.techinst... AndroidRun... at android.app.ActivityThread.main(ActivityThread.java:5017)
E 06-09 11:44:16.225 24191 24191 jp.techinst... AndroidRun... at java.lang.reflect.Method.invokeNative(Native Method)
E 06-09 11:44:16.225 24191 24191 jp.techinst... AndroidRun... at java.lang.reflect.Method.invoke(Method.java:515)
E 06-09 11:44:16.225 24191 24191 jp.techinst... AndroidRun... at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.j ava:779)
```

図20:例外発生時のログ

図20のAのところが、例外の種類（原因）です。今は「NullPointerException」という例外が発生しています。例外の原因がわからない場合は、このException名を検索エンジンで検索するようにしましょう。Bのところが、例外が発生したファイルと行数で、MainActivity.javaの32行目で例外が発生したと示されています。動作が確認できたら、先ほどコメントアウトした行を元に戻しておきましょう。



6-2-4 独自のViewを利用してみよう

ここまで、音楽ファイルの再生を通じて、ActivityのライフサイクルやLogの出力について見てきました。ここで独自の「View」を考えてみたいと思います。前節ではButtonのインスタンスを生成し、setContentView()メソッドを利用して画面に貼り付けました。そういったButtonやレイアウトファイルなど、画面に貼り付けることができるAndroidの部品のことをViewといいます。

すでに用意されているButtonなどは、そのままインスタンスを生成し画面に貼り付けることができましたが、独自にViewを作成して利用することもできます。Viewを作成するために、まずクラスを定義しましょう。

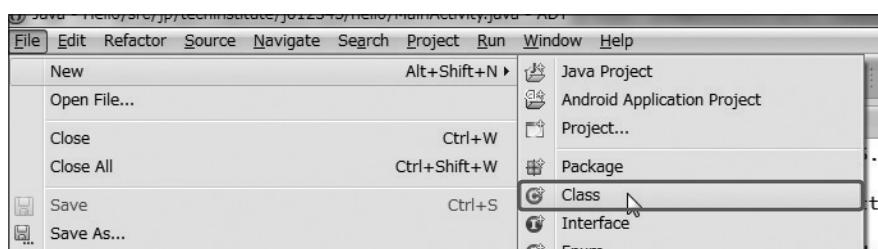


図21:新規クラスの作成

「File」メニューから「New」→「Class」を選択します(図21)。すると新規クラスのダイアログボックスが表示されます(図22)。

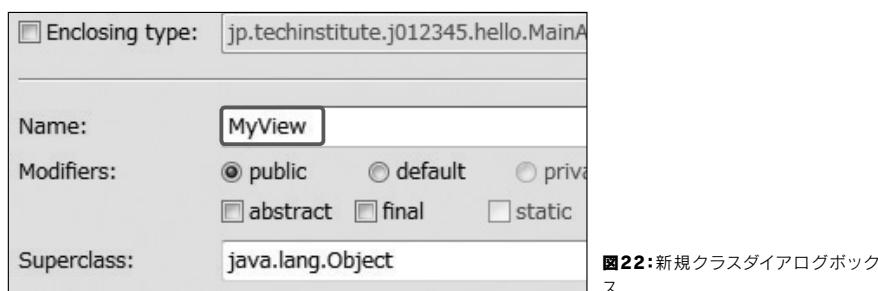


図22:新規クラスダイアログボックス

「Name」欄に「MyView」と入力し、「Finish」ボタンをクリックします。Eclipseの編集領域に、クラスが次のように開きます(図23)。

```
1 package jp.techinstitute.j012345.hello;
2
3 public class MyView {
4
5 }
```

図23:作成されたクラス

画面に表示する部品であることを示すためにViewを継承します。Viewはインポート文が必要です。Viewに赤波線が引かれた場合、マウスカーソルをポイントしてインポートを実行しましょう(図24)。

The screenshot shows a Java code editor with the following code:

```
1 package jp.techinstitute.j012345.hello;
2
3 public class MyView extends View {
4
5 }
6
```

A quick fix dialog is open, showing the message "View cannot be resolved to a type" and three options:

- Import 'View' (android.view)
- Create class 'View'
- Fix project setup...

図24:Viewのインポート

すると、次はMyViewに赤波線が表示されます。Viewにはデフォルトコンストラクタが定義されていないため、Viewに定義されているコンストラクタを実装する必要があるからです。同様にマウスカーソルをポイントし、一番上部のコンストラクタを実装します。

The screenshot shows the same Java code as before, but now the constructor "View()" has a red underline. A quick fix dialog is open, showing the message "Implicit super constructor View() is undefined for default constructor" and three options:

- Add constructor 'MyView(Context)'
- Add constructor 'MyView(Context, AttributeSet)'
- Add constructor 'MyView(Context, AttributeSet, int)'

図25:コンストラクタの実装

以上の作業の結果は、次のようなソースコードになります。

```
MyView.java
package jp.techinstitute.j012345.hello;

import android.content.Context;
import android.view.View;

public class MyView extends View {
    public MyView(Context context) {
        super(context);
        // TODO Auto-generated constructor stub
    }
}
```

このまでは何も表示されませんので、Viewが表示される時に実行される「onDraw()」メソッドをオーバーライドします。そのために、「Source」メニューから「Override/Implements Methods…」を選択します(図26)。

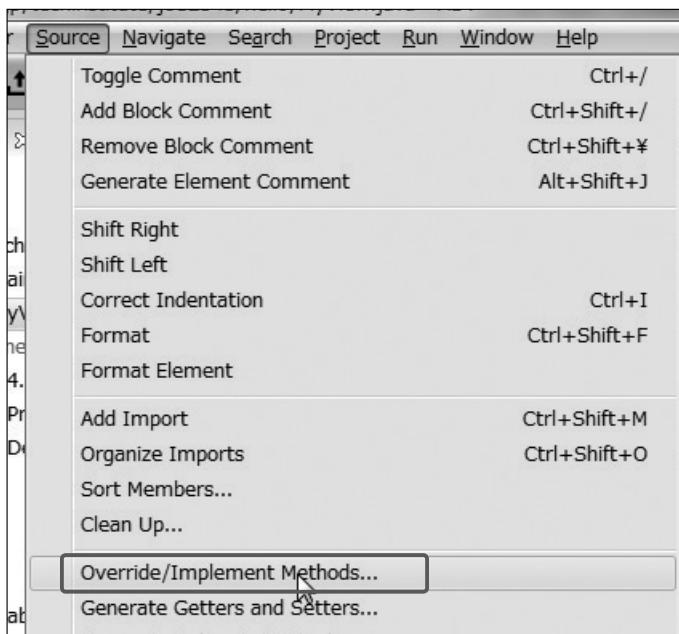


図26:メソッドのオーバーライド

ダイアログボックスが表示されるので、`onDraw()`メソッドを選択し、「Ok」をクリックしましょう(図27)。

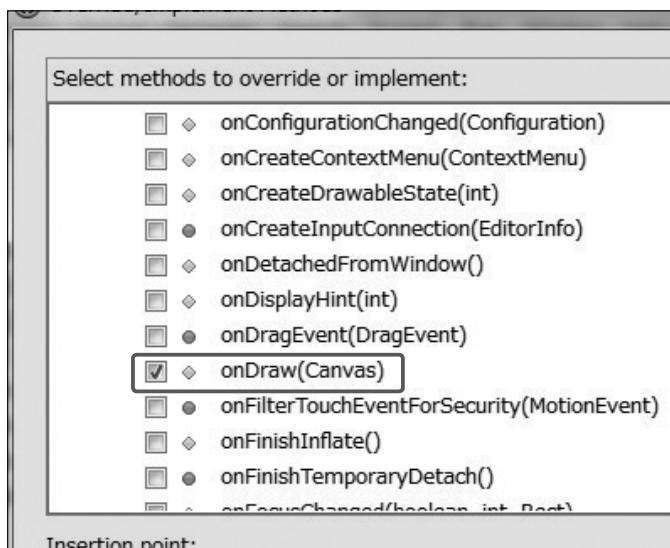


図27:`onDraw()`メソッドの選択

そうすると、ソースコードに次のように`onDraw()`メソッドが追加されます。

```
MyView.java
@Override
protected void onDraw(Canvas canvas) {
    // TODO Auto-generated method stub
    super.onDraw(canvas);
}
```

`onDraw()`メソッドでは「Canvas」のインスタンスが引数として渡ってきます。Canvas(キャンバス)は、油絵などを描く時に利用するキャンバスと同じ意味で、さまざまな描画を行うことができます。Canvasでできることの詳細は、APIドキュメント等で確認しましょう。ここでは、簡単に背景色を緑にしてみましょう。次のようにソースコードを修正します。

ButtonやCanvasなどのAndroidで利用するクラスは「API(Application Programming Interface)ドキュメント」として、開発者向けサイトの次のURLで公開されています。

<http://developer.android.com/reference/packages.html>

Canvasについて調べたい場合は、右上の検索窓で検索すると、CanvasのAPIドキュメントで性質(フィールド)や動作(メソッド)の説明を読むことができます

MyView.java

```
@Override  
protected void onDraw(Canvas canvas) {  
    super.onDraw(canvas);  
    canvas.drawColor(Color.GREEN);  
}
```

Activityを修正する前にViewの変更は保存しておきましょう

ここでは「Color」クラスに定義されているGREENを利用して、Canvasクラスの「drawColor()」メソッドを実行しています。Colorもインポートが必要ですので、赤波線が出た場合はインポートを実行してください。

このViewを利用するように、MainActivityを修正します。後でまた元に戻すので、次のように修正しましょう。

MyView.java

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    Log.d("TAG", "onCreateが実行されました");  
    MyView v = new MyView(this);  
    setContentView(v);  
/*  
    setContentView(R.layout.activity_main);  
    Button btn = (Button) findViewById(R.id.button1);  
    btn.setText("Hello");  
    mPlayer = MediaPlayer.create(this, R.raw.music);  
    btn.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View arg0) {  
            Button b = (Button) arg0;  
            b.setText("こんにちは");  
            mPlayer.start();  
        }  
    });  
*/  
}
```

A

B

Aの部分が、作成したMyViewのインスタンスを生成して、画面に貼り付けているところです。Bはコメントアウトして、動作しないようにします。実行すると、次のように表示されます(図28)。

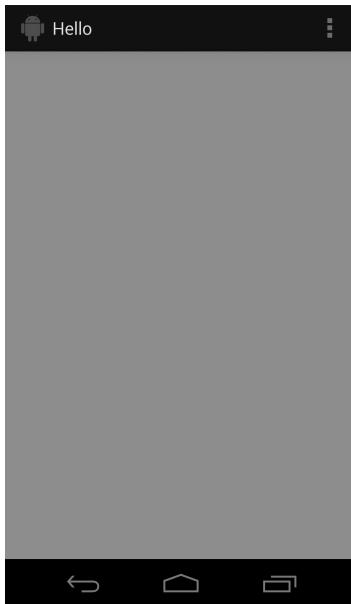


図28:背景色を変更

原色の緑だと少々目に悪そうです。そこで、自分で色を指定してみたいと思います。色の指定はRGB(Red, Green, Blue)の3原色の強さで指定することができ、それぞれ0-255の整数で指定します。RGBにどんな値を入れれば何色になるのかについては、「RGB 色」などのキーワードで検索して、調べてみて下さい。

ここでは、Colorクラスの「argb()」メソッドを利用して、色を指定してみます。なお、argbのaは色の透過を示していて、塗りつぶす場合は255を指定します。次のようにソースを修正してみましょう。

MyView.java

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    canvas.drawColor(Color.argb(255, 80, 116, 62));
}
```

実行すると、次のように表示されます(図29)。

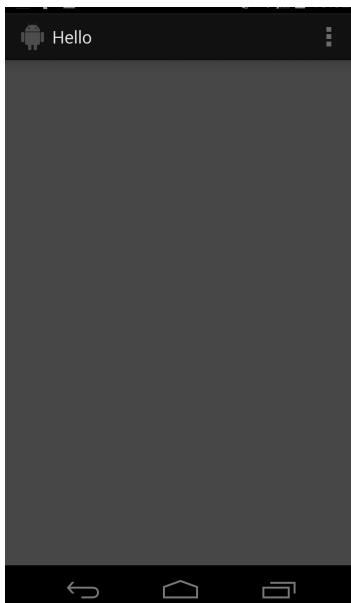
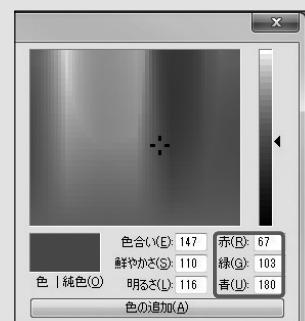


図29:色の指定

色を指定するためのRGB値は、Windowに付属するペイントなどを利用して確認できます。ペイントを起動し、右上にある「色の編集」ボタンをクリックすると、色の編集ダイアログボックスが表示されます。このダイアログボックスの右側にRGB値を設定すれば、どんな値がどんな色になるのかを確認できます。



Canvasを利用すれば、背景色の変更以外に四角形や円など、さまざまな図形を描くこともできます。試しに円を描いてみましょう。円を描くには、Canvasの「drawCircle()」メソッドを利用します。引数に(円の中心のX座標、円の中心のY座標、半径、Paintオブジェクト)を指定します。

このとき、座標は左上が原点で次のようになっています(図30)。

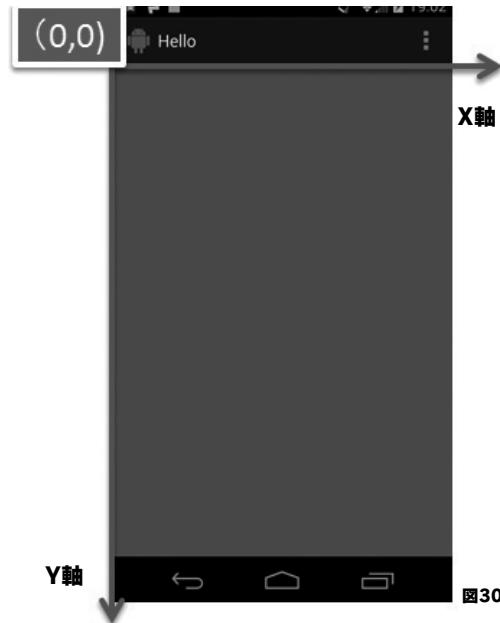


図30:Viewと座標の関係

また第4引数の「Paint」は、筆に相当するオブジェクトで、円などの対象の色を指定することができます。次のようにソースコードを修正して、円を描いて見ましょう。

```
MyView.java
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    canvas.drawColor(Color.argb(255, 80, 116, 62));
    Paint paint = new Paint();
    paint.setColor(Color.WHITE);
    canvas.drawCircle(100, 100, 100, paint);
}
```

実行すると、次のように左上に円が表示されます(図31)。

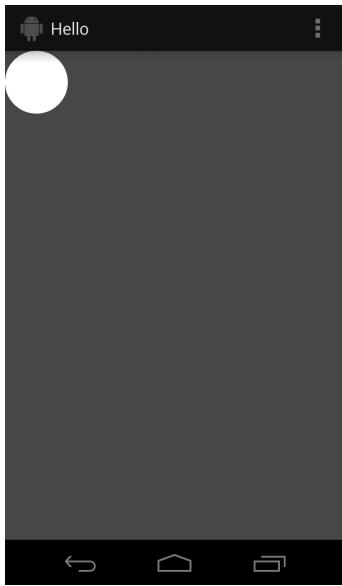


図31:円を表示

では、描画した円を動かすにはどのようにすればいいのでしょうか。描画位置を少しづつずらして、再描画をくり返せば動いているように見えるはずです。円の位置を示すmX、mYと移動量を示すmVX、mVYを、次のようにフィールドとして定義しましょう。

MyView.java

```
public class MyView extends View {
    int mX = 100;
    int mY = 100;
    int mVX = 1;
    int mVY = 1;
~略~
```

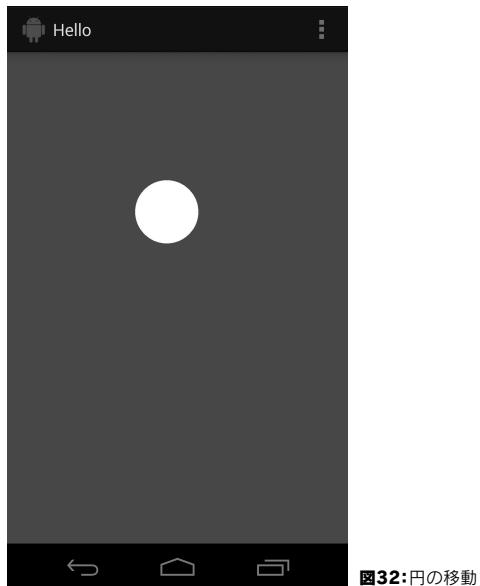
定義したフィールドを利用するようにonDraw()メソッドを変更します。また、onDraw()メソッドの最後に、「invalidate()」メソッドを記述して、強制的に再描画させてみましょう。

MyView.java

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    canvas.drawColor(Color.argb(255, 80, 116, 62));
    Paint paint = new Paint();
    paint.setColor(Color.WHITE);
    canvas.drawCircle(mX, mY, 100, paint);
    mX += mVX;
    mY += mVY;
    invalidate();
}
```

invalidateは無効という意味です。invalidate()メソッドを実行すると一度Viewを無効化した後に、onDraw()メソッドが呼び出されます。invalidate()メソッドをonDraw()メソッドの中に記述することで、onDraw()→invalidate()→onDraw()といった形でonDraw()メソッドが繰り返し呼び出され、アニメーションを実現することができるのです。

実行すると、次のように円が右下に少しずつ動いていくことが確認できます(図32)。



特に画面のふちで跳ね返るような処理は入れていないため、しばらく経つと円は画面から外へはみ出てしまいます。画面から跳ね返るような処理を入れると、簡単なゲームのようなものを作成することもできます。ここでは、Viewの境界で跳ね返るような処理だけ入れてみましょう。右や下の境に来た時には移動量をマイナスに、左や上に来た場合はマイナスになっている移動量を更にマイナスをかけてプラスに変更します。

MyView.java

```
@Override  
protected void onDraw(Canvas canvas) {  
    super.onDraw(canvas);  
    canvas.drawColor(Color.argb(255, 80, 116, 62));  
    Paint paint = new Paint();  
    paint.setColor(Color.WHITE);  
    canvas.drawCircle(mX, mY, 100, paint);  
    if (mX > this.getWidth() ) {  
        mVX = -mVX;  
    } else if (mX < 0) {  
        mVX = -mVX;  
    }  
    if (mY > this.getHeight()) {  
        mVY = -mVY;  
    } else if ( mY < 0){  
        mVY = -mVY;  
    }  
    mX += mVX;  
    mY += mVY;  
    invalidate();  
}
```

まとめ

- 1.APIを利用すると簡単にアプリケーションを作成することができる
- 2.アプリケーション(Activity)にはライフサイクルがあり、ライフサイクルに応じたメソッドが用意されている
- 3.アプリケーションの動作はLogを出力することで確認できる
- 4.実行時に例外が発生した場合は、LogCatに内容が出力されるので、原因を確認することができる

6-3

ためしてわかる Android のしくみ (3)

著：飯塚康至

6-2で“作成したアプリケーションに機能を追加していくことで”、
アプリケーション間での連携や、他のアプリケーションを起動
する方法について学習していきます。



6-3-1 他のアプリケーション (Activity) との連携

ここでは、次のことを学習します。

- ①アプリケーション間で連携する方法
- ②独自のViewを定義して利用する方法

Intentによる複数アプリケーションの連携

Androidでは、端末にインストールされている複数のアプリケーションを連携させることができます。たとえば、あるアプリケーションからメールアプリを呼び出してメールを送信したり、地図アプリを呼び出して地図を表示したりすることができます。複数のアプリケーションを連携させることにより、まるで1つの大きなアプリケーションのように動作させることができます。

また、他のアプリに処理を移譲することにより、自分のアプリでは地図やメールの機能を実装しなくとも良くなります。

このようなAndroidの仕組みを、「Intent」(インテント)といいます。Intentとは、意図や目的といった意味で、ある目的のために処理を移譲するAndroidの機能になります。

Intentには、アプリケーション(Activity)を指定して処理を移譲する「明示的Intent」と、処理内容だけを決めて処理を行うアプリケーションは自動的に選択(複数ある場合はユーザーが選択)する場合の「暗黙的Intent」の2種類があります。

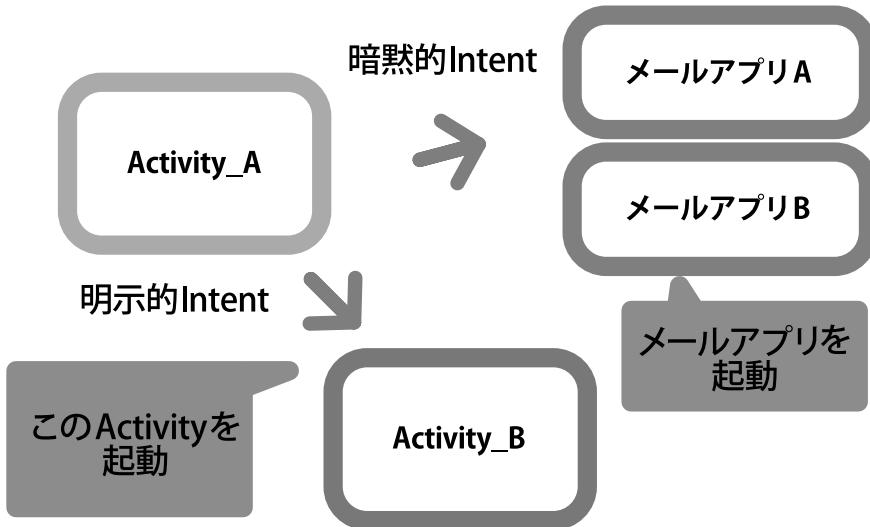


図1:インテントによる連携

これからの作業のために、前節で変更したMainActivityの記述を次のように元に戻しておきます。

```
MainActivity.java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Log.d("TAG", "onCreate が実行されました ");
    setContentView(R.layout.activity_main);
    Button btn = (Button) findViewById(R.id.button1);
    btn.setText("Hello");
    mPlayer = MediaPlayer.create(this, R.raw.music);
    btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Button b = (Button) v;
            b.setText("こんにちは");
            mPlayer.start();
        }
    });
}
```



6-3-2 暗黙的 Intent

暗黙的Intentを確認するために、「activity_main.xml」を開き、レイアウトにボタンを追加してみましょう。

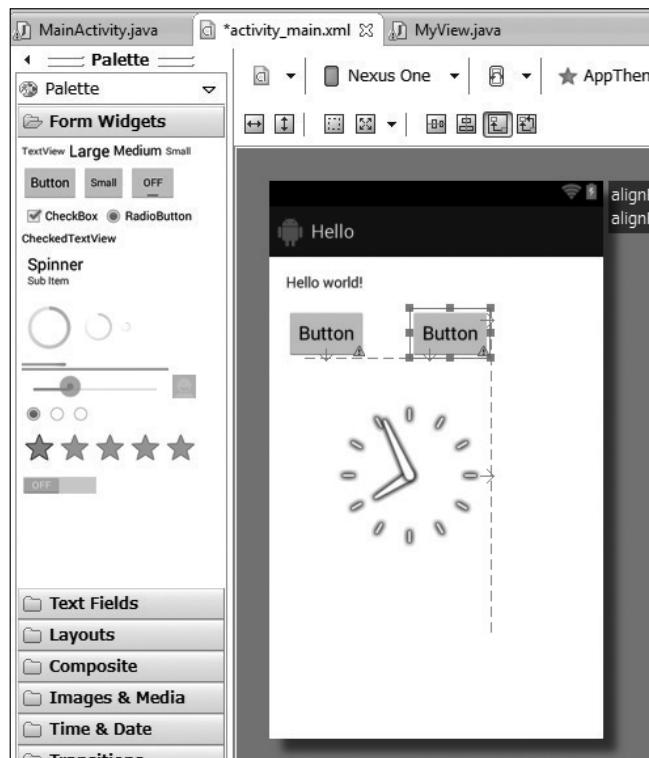


図2:ボタンの追加

追加したボタンのIDを確認してみます。ボタンのテキストを、「インテント」に変更しましょう。変更や確認は、レイアウト画面の右下に表示される「Properties」から行なうことができます。



図3:IDの確認とテキストの変更

IDは「button2」となっています。IDを利用してこのボタンをActivityから取得するため、onCreate()内に次のような処理を追加しましょう。

MainActivity.java

～略～

```
mPlayer = MediaPlayer.create(this, R.raw.music);
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View arg0) {
        Button b = (Button) arg0;
        b.setText("こんにちは");
        mPlayer.start();
    }
});

Button btn2 = (Button) findViewById(R.id.button2);
btn2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
    }
});
```

～略～

ボタンを押した時に、Intentを利用するようにしてみたいと思います。暗黙的Intentを利用するには以下の手順を行います。

- 1. 移譲したい処理内容を記載するURIを定義する**
- 2. Intentのオブジェクトを定義し、移譲したい処理を記述する**
- 3. 処理を移譲する**

たとえば、地図を開くにはbtn2の「onClick(View v)」の「{}」の中に、次のように記述します。

MainActivity.java

～略～

```
Uri uri = Uri.parse("geo:0,0?q=Shibuya");
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(intent);
```

～略～

URIとは「Uniform Resource Identifier」の略で、「http://」のような一定の書式によってリソース(資源)を指し示すものです。Webサイトの場所を表すURLもURIに含まれますが、URIは場所だけでなく名前も定義できます。

実行して、ボタンをタップすると、地図を表示するためのインテントが発行され、対象となるアプリケーションが複数ある場合は、次のようなリストが表示されます(図4)。



図4:起動対象リスト

マップを選択すると、次のように地図が開きます(図5)。



図5:「マップ」アプリによって、地図が表示される

今度は連絡帳を表示させてみましょう。次のようにソースコードを修正します。

MainActivity.java

~略~

```
Uri uri = Uri.parse("tel:090-222-333");
Intent intent=new Intent(Intent.ACTION_DIAL,uri);
startActivity(intent);
```

~略~

実行してボタンをタップすると、次のように連絡先が表示されることがわかります（図6）。

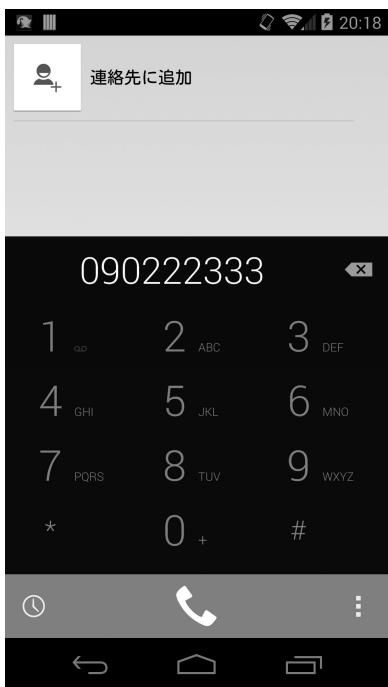


図6:連絡先の表示

このように、暗黙的Intentを利用することで、複数のアプリケーションと間で連携を行い、あたかもひとつの大きなアプリケーションのように振る舞わせることが可能となります。

暗黙的Intent利用時の、URIの代表的な指定方法は次のとおりです。

機能	サンプル
ブラウザを起動する	<code>Uri uri = Uri.parse("http://www.exsample.com"); Intent intent = new Intent(Intent.ACTION_VIEW, uri);</code>
通話開始	<code>Uri uri = Uri.parse("tel:0120-0123-123"); Intent intent = new Intent(Intent.ACTION_CALL, uri);</code>
地図の表示	<code>Uri uri = Uri.parse("geo:0,0?q=Shibuya"); Intent intent = new Intent(Intent.ACTION_VIEW, uri);</code>

表3:代表的なURIの指定



6-3-3 パーミッション

暗黙的Intentを利用して、アプリケーションから家族などの親しい人に電話をかけられるようにすると、便利かもしれません。Actionを「ACTION_CALL」に変更することで、暗黙的Intentを利用して直接電話をかけることができるようになります。次のように変更してみましょう。

～略～

```
Uri uri = Uri.parse("tel:090-222-333");
Intent intent=new Intent(Intent.ACTION_CALL,uri);
startActivity(intent);
```

～略～

しかし、アプリケーションを実行し、ボタンをタップすると、異常終了してしまいました。原因を確かめるためにLogCatを確認してみましょう。

```
E 06-09 20:31:22.558 27554 27554 jp.techinstitute.AndroidRun... java.lang.SecurityException: Permission Denial: starting Intent { act=android.intent.action.CALL dat=tel:xxx-xxx-xxx cmp=com.android.phone/.OutgoingCallBroadcaster } from ProcessRecord{426e85f8 27554:jp.techinstitute.j012345. hello/u0a122} (pid=27554, uid=10122) requires android.permission.CALL_PHONE
```

図7:LogCatの確認

「java.lang.SecurityException: Permission Denial」と表示されていることから、どうやらセキュリティ上の問題で異常終了してしまったようです。確かに、アプリケーションから直接電話をかけることができてしまうと、利用者が意図しないところに勝手に電話をかけてしまうかもしれません、これは問題がありそうです。

利用者の不利益が考えられる動作に関しては、Androidアプリケーションは権限を取得しなければ実行できない仕様になっています。実際にアプリケーション配布場所のひとつであるGooglePlayからアプリケーションをインストールしようとすると、次のような権限の一覧が表示されるので、利用者は権限を確認してインストールするかどうかを決めるというセキュリティモデルになっています(図8)。

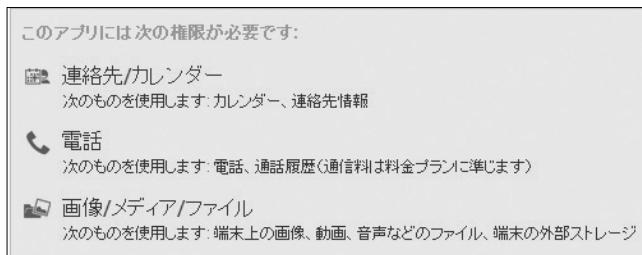


図8:GooglePlayでの権限の表示

このセキュリティモデルでは、たとえば電卓のアプリケーションが、インターネットにアクセスしたり連絡帳にアクセスしたりする権限を要求するのはおかしいなどと、利用者が自主的に利用の是非を判断することを考えています。

では、アプリケーションに権限を付与するにはどうすればいいでしょうか。Android

アプリケーションには、バージョン情報などのさまざまな情報を記した、「マニフェストファイル」というものがあります。権限もこのマニフェストファイルに記載するようになっています。権限を、マニフェストファイルに記載することで、外部からこのアプリケーションの動作にどのような許可が必要なのかが分かるようになります。

マニフェストファイルを開くには、Package Explorerから「AndroidManifest.xml」ファイルを開きます(図9)。

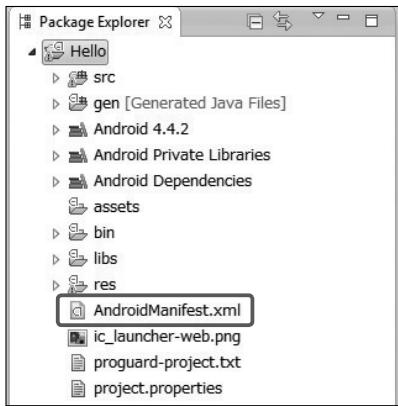


図9:マニフェストファイル

マニフェストファイルを開くと、次のような画面が表示されます(図10)。

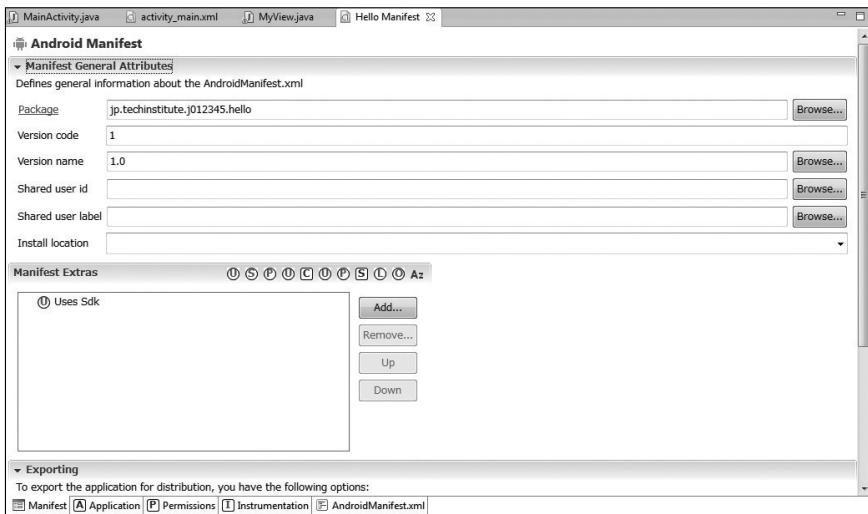


図10:AndroidManifest.xml

マニフェストファイルの下部には、カテゴリごとに分けられたタブが表示されています(図11)。設定したい内容に応じてタブを切り替えていきます。

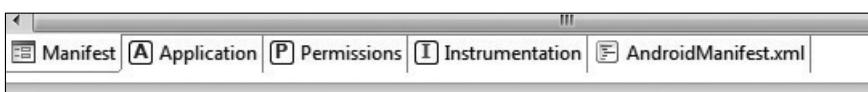


図11:下部タブの一覧

許可権限を付与するには、「Permissions」タブを開きます(図12)。



図12:Permissionsタグ

ここで「Add…」ボタンをクリックすると、次のようなダイアログボックスが表示されます(図13)。

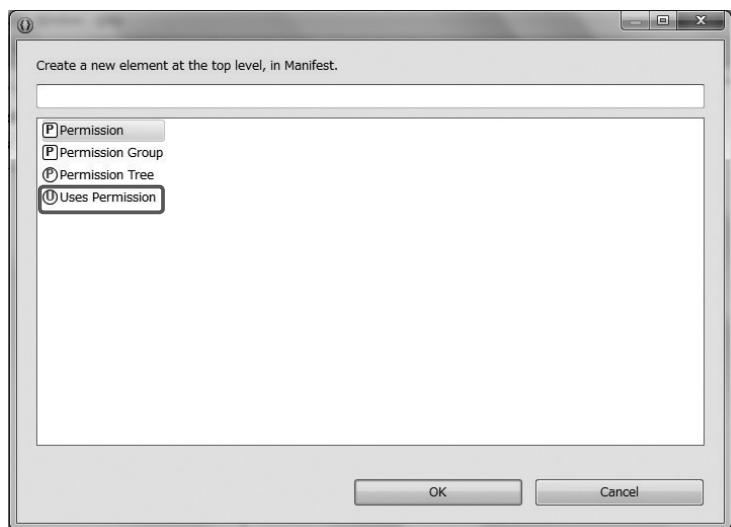


図13:パーミッションダイアログボックス

「Use Permission」を選択してOKをクリックし。すると、マニュフェスト画面の右部分に次のような設定項目が表示されます(図14)。

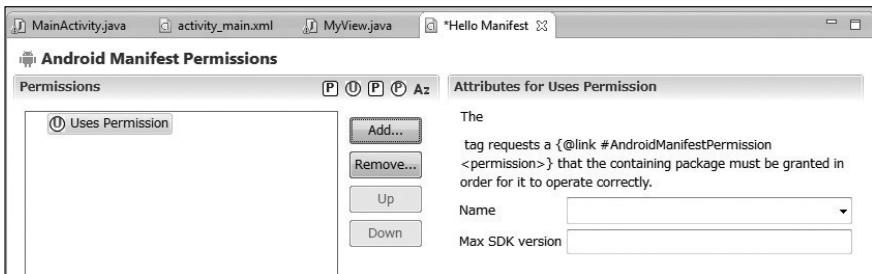


図14:パーミッション設定

「Attributes for Uses Permission」の「Name」の右側にあるドロップダウンメニューをクリックすることで、パーミッションを選択できます。電話をかけるためのパーミッションは、「Android.permission.CALL_PHONE」です(図15)。

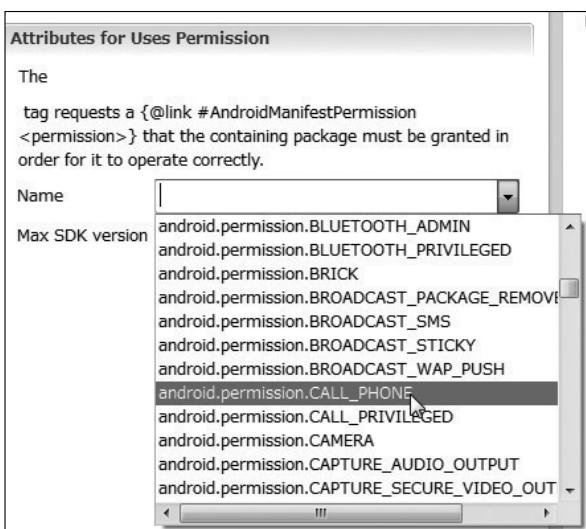


図15:パーミッションの選択

パーミッションを選択したら、マニフェストファイルを保存して、アプリケーションを実行してみましょう。パーミッションを追加したことでのIntentを利用してアプリケーションから電話機能を呼び出し、直接電話をかけるようになりました。

パーミッションにはその他に、インターネットアクセスや連絡帳データの読み込み、外部ストレージへの書き込み、カメラの利用など、さまざまな種類のものが存在します。その他のパーミッションの詳細は、次のURLを参照するようにしてください。

【パーミッションの一覧】

<http://developer.android.com/reference/android/Manifest.permission.html>



6-3-4 明示的 Intent

あるActivityから別のActivityを指定して起動する方法を、「明示的Intent」といいます。明示的Intentを利用することで、画面遷移を行うことができます。

明示的Intentを確認するために、新しいActivityを作成してみましょう。Package Explorerから「src」を展開し、パッケージ名を右クリックして「New」→「Class」を選択します(図16)。

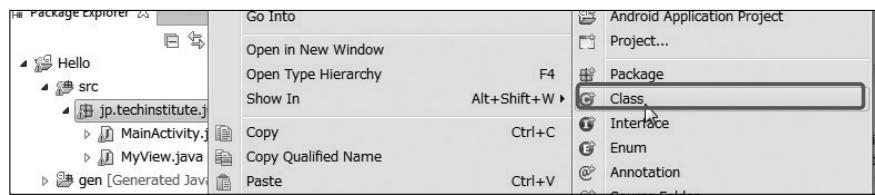


図16:新規クラスの作成

新規クラスのダイアログボックス表示がされるので、Name欄に「MyActivity」と入力して「Finish」をクリックします

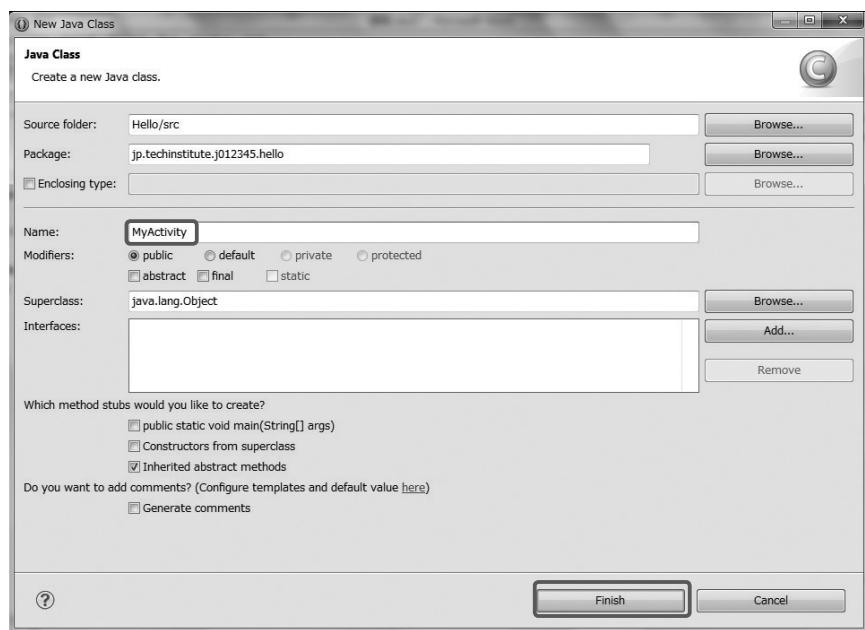


図17:新規クラスのダイアログボックス

すると、クラスが生成されて編集画面が開くので、次のようにActivityを継承します。Activityはインポートが必要なので、赤波線が表示された場合はマウスカーソルをポイントしてインポート文を追加します。

```
MyActivity.java
package jp.techinstitute.j012345.hello;

import android.app.Activity;

public class MyActivity extends Activity {
}
```

次に、Activity起動時に実行される「onCreate()」メソッドをオーバーライドするために、Eclipseの「Source」メニューから「Override/Implements Methods …」を選択し、表示されたダイアログボックスで「onCreate()」にチェックを入れ、「OK」をクリックします(図18)。

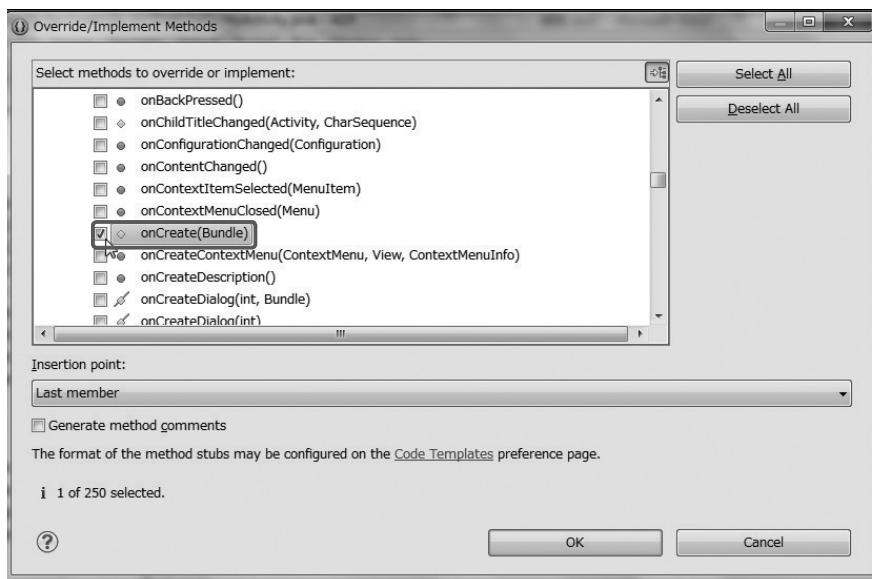


図18:onCreate()のオーバーライド

挿入されたonCreate()メソッドに、Viewを設定します。Viewは前節で利用した独自のViewを設定しましょう。

```
MyActivity.java
~略~
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    MyView v = new MyView(this);
    setContentView(v);
}
~略~
```

これで、独自のViewを持つ独自のActivityが完成したので、明示的Intentを利用してこのActivityを起動するために、MainActivity.javaを開き、次のようにbutton2をクリックした時の動作を変更します。

MainActivity.java

```
~略~  
btn2.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        Intent intent = new Intent(MainActivity.this, MyActivity.class);  
        startActivity(intent);  
    }  
});  
~略~
```

明示的Intentを行うにはIntentのコンストラクタで、自分自身を示すMainActivity.thisと、起動したいActivityであるMyActivityを、MyActivity.classのようにクラスファイル名で指定します。ファイルを保存して実行すると、次のような実行時例外が出力され、異常終了します。

```
AndroidRun... android.content.ActivityNotFoundException: Unable to find explicit activity class {jp.techinstitute.j012345.hello/jp.techinstitute.j012345.hello.MyActivity}; have you declared this activity in your AndroidManifest.xml?
```

図19:明示的Intent実行時の例外

例外の内容を見ると、「ActivityNotFoundException」と表示されています。どうやら作成したActivityが見つからないようです。また、マニフェストファイルに宣言しましたか? といった記述があります。

Androidアプリケーションは、アプリケーション内で利用するActivityをマニフェストファイルに登録して、他のアプリケーションなどの外部から見える状態にしないと利用することができません。Intentは、マニフェストに定義された情報をもとに動作しているのです。

そこで、作成したMyActivity.javaを追加するためにマニフェストファイルを開き、「Application」タブから、「Application Nodes」の「Add…」ボタンをクリックします。

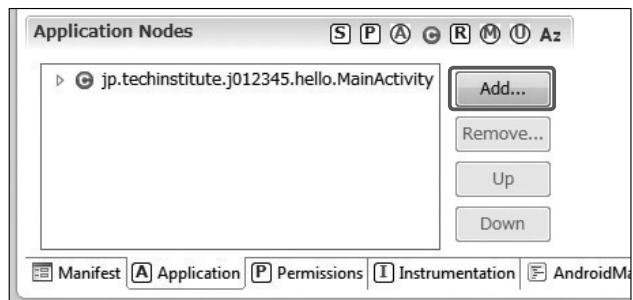


図20:Activityの登録

ダイアログボックスが開くので、Activityを選択して「OK」をクリックします(図21)。

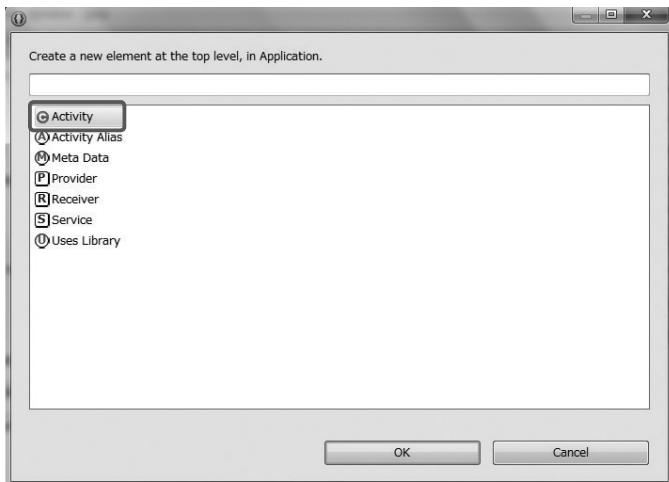


図21:Nodes追加ダイアログボックス

マニュフェストの画面が次のように切り替わるので、Nameの「Browse…」ボタンをクリックします(図22)。



図22:Activityの設定

すると、次のダイアログボックスが表示されます。利用できるActivityが検索するので少し待つと、次のように表示されます(図23)。

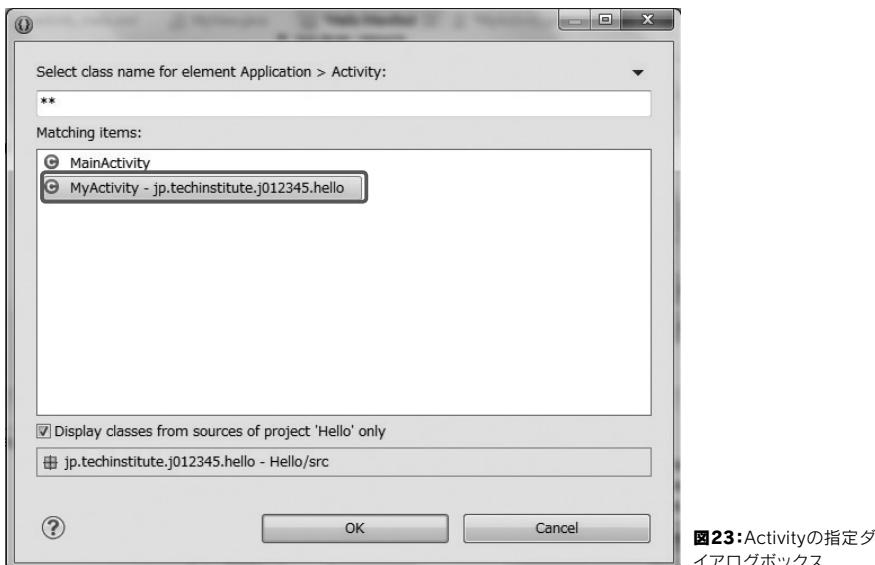


図23:Activityの指定ダイアログボックス

MyActivityを選択し、「OK」をクリックします。その後、ファイルを保存すると、マニュフェストの画面が次のようになります(図24)。

このようにActivityを登録することで、初めて新規に作成したActivityを利用することができます。

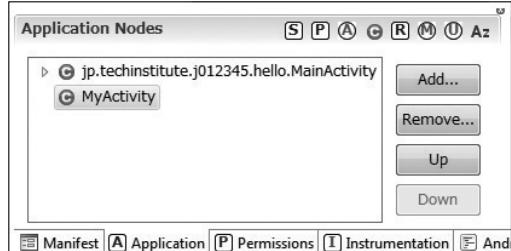


図24:Activity登録後



6-3-5 データの受け渡し

Intentを利用して他のActivityを起動する時にはデータを受け渡すこともできます。データを受け渡すには、Intentクラスの「putExtra()」メソッドを利用します。putExtra()メソッドの第1引数に受け渡すデータの名前、第2引数に受け渡すデータを記述します。例えば、vxという名前で整数の10をセットしたい場合は、

```
intent.putExtra("vx", 10);
```

と記述します。MainActivityのbutton2をクリックした時の動作を、次のように修正してみましょう。

MainActivity.java

```
~略~
Button btn2 = (Button) findViewById(R.id.button2);
btn2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(MainActivity.this, MyActivity.class);
        intent.putExtra("vx", 10);
        intent.putExtra("vy", 10);
        startActivity(intent);
    }
});
~略~
```

このようにセットしたデータは、起動先のMyActivityで取得することができます。まずActivityのメソッドである「getIntent()」を利用してIntentのオブジェクトを取得します。次に受け渡したデータの型にあわせて「get-型名-Extra」(名前、デフォルト値)といった形式でデータを取得します。

MyActivityのonCreate()メソッドを、次のように修正してみましょう。

MyActivity.java

```
~略~
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    MyView v = new MyView(this);
    Intent intent = getIntent();
    int vx = intent.getIntExtra("vx", 1);
    int vy = intent.getIntExtra("vy", 1);
    v.mVX = vx;
    v.mVY = vy;
    setContentView(v);
}
~略~
```

Intentを利用してデータを受け渡すことで、この場合はMyViewで描画している円の移動速度を変更しています。保存して実行すると、MyViewの円が早く動くようになったことがわかります(図25)。

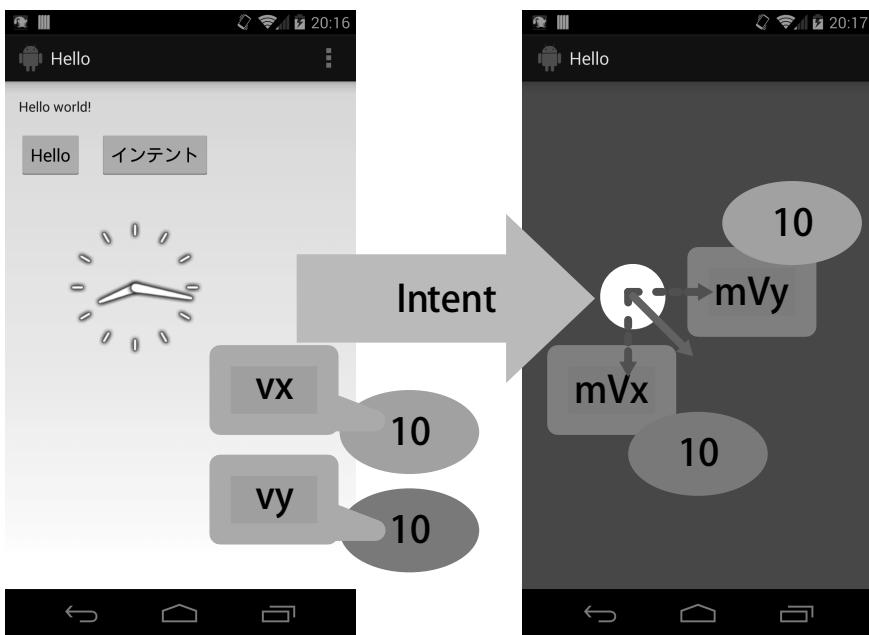


図25:実行例

まとめ

- 1.Intentを利用することで、他のアプリケーション(Activity)と連携できる
- 2.暗黙的Intentは処理内容を指定して他のアプリケーションと連携できる
- 3.処理内容によっては、権限の追加が必要となる
- 4.明示的Intentを利用することで、画面遷移を行うことができる
- 5.Intentを利用する際、データを受け渡すことができる

