Android チュートリアル 1
–Android のはじめ方–*

愛知県立大学 山本研究室
藤浦 祥雅†

0. インストールとプラグインの導入

まずは，Android アプリケーションを開発するための環境設定を行います．必要なツールは，以下の三つです．

1. Android SDK[1]
2. Eclipse[2]
3. ADT[3] (Android Development Tools) プラグイン

この三つをインストールすれば，Eclipse で Android アプリケーションの開発を始めれます．
基本的には，Eclipse を使った開発方法が主流となっております．エディタでの作成方法もありますが，そちらはドキュメントを見てください．

1. プロジェクトの作成

まずは，Android プロジェクトを作成してみます．図 1 は，Android プロジェクト作成画面です．ここでは，4 つの項目を入力します．

プロジェクト名 プロジェクト名を入力します．

パッケージ名 パッケージ名を入力します．パッケージ名は，他のアプリケーションと混合しないよう 2 単語以上でなければなりません．

アクティビティー名 アクティビティー名を入力します．このアクティビティー名がメインで動作するクラスになります．

アプリケーション名 アプリケーション名を入力します．このアプリケーション名は，ホームで表示されるアプリケーションの名前になります．

*URL:[http://www.aichi-pu.ac.jp/ist/lab/yamamoto/android/android-tutorial/tutorial01/tutorial01.pdf]
†mail:[qtutorial@gmail.com]

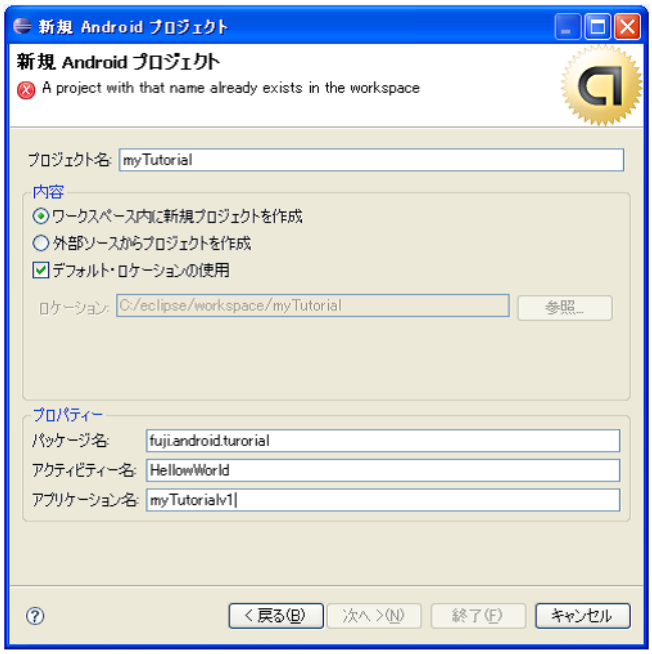

図 1: Android プロジェクト作成画面

以上の 4 項目を入力したら終了をクリックしてください．すると, Android プロジェクトが作成されます．自動生成されるプロジェクトの中身は，図 2 のようになってます．

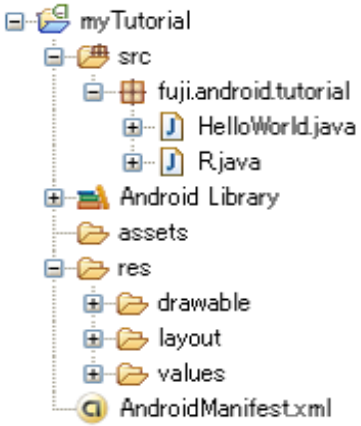

図 2: プロジェクトの構成

各ディレクトリおよびファイルの内容は, 次のようになってます.

myTurorial Android プロジェクト

src ソースディレクトリ

fuji.android.tutorial パッケージ

HellowWorld.java アプリケーションのメインクラス

R.java リソースのインデックスファイル

assets アセットを保持するディレクトリ

res リソースを保持するディレクトリ

drawable 画像を保持するディレクトリ

layout レイアウトファイルを保持するディレクトリ

values 文字列ファイルを保持するディレクトリ

AndroidManifest.xml アプリケーションのマニフェストファイル

2. Hello World を出力してみよう

2.1 とりあえず実行してみよう

まず, プロジェクトを作成したまま実行してみよう. すると図 3 のような画面が表示される.

図 3: とりあえず実行の画面

すでに，“Hellow World” が出力されています．なにもしていないのに．これは，なぜでしょうか．
これには，Android アプリケーションの仕組みが関係しています．
以下のリスト 1 は，HelloWorld.java の中身です．

リスト 1: HelloWorld.java

```
1  package fuji.android.tutorial;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5
6  public class HelloWorld extends Activity {
7      /** Called when the activity is first created. */
8      @Override
9      public void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.main);
12     }
13 }
```

リスト 1 は，Activity クラスを継承したクラスです．この Activity クラスの子クラスは，Activity と呼ばれます．Activity とは，アプリケーションの画面であり，アプリケーションの UI です．ここでは，“HelloWorld” という Activity がこのアプリケーションの UI として動作を行います．

リスト 1 では，たった一つだけ自動生成される ”onCreate()“メソッドがあります．これは，Activity にとってのメイン関数のようなもので，アプリケーションが実行された時に一番初めに呼ばれます．onCreate() のようなメソッドはライフサイクルメソッドと呼ばれ，Activity の状態が変化する時にシステムから呼ばれます．

このメソッドのように，外部から呼ばれるメソッドは，オーバーライドして固有の処理を付加していくのですが，まず親クラスのメソッドを呼び出してから，固有の処理を書いていきます．今回では，11 行目のみがこの “HelloWorld” アプリケーション固有のコードになります．

さて，このなにも手を加えていないプログラムで文字が出力されたのかの答えが，11 行目です．このメソッドは，レイアウトファイルからレイアウトを取り出して設定しています．ようは，このレイアウトファイル自体に，あの文字列が入り込んでいるため，なにもしなくても，文字が出力されたのです．

11 行目で呼び出されるレイアウトファイルと文字列ファイルは，リスト 2，3 に示します．

リスト 2: レイアウトファイル (main.xml)

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >
7  <TextView
8      android:layout_width="fill_parent"
9      android:layout_height="wrap_content"
10     android:text="@string/hello"
11     />
12 </LinearLayout>
```

リスト 3: 文字列ファイル (strings.xml)

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <string name="hello">Hello World, HelloWorld</string>
4     <string name="app_name">Tutorialv1</string>
5 </resources>
```

レイアウトファイルは，画面のレイアウトを決めるファイルである．リスト 2 では，画面全体を LinearLayout というレイアウトマネージャを利用し，TextView という子を持つ．TextView では，テキストを表示する領域を示すもので，リスト 2 では，その縦と横の幅と表示するテキストの内容を保持している．(l. 8 - l. 10)

リスト 2 の 10 行目では，文字列ファイルより文字列を読み込んでいます．ここでは，“@string/hello” が呼び込んでいる部分になります．その内容としては，リスト 3 の 3 行目と対応している．これで，“string” タグの name が “hello” の値 (Hello World, HelloWorld) を出力できます．

また，レイアウトファイルを利用するときは，“R.xxx.yyy” のように記述して利用します．“xxx” の部分には，res ディレクトリ以下のディレクトリ名 (ここでは，layout) を，“yyy” の部分には，xml ファイル名 (ここでは，main.xml であるが，“.xml” を省略して “main” だけ) を記述します．これで，“layout ディレクトリの main.xml を参照” するということを示しています．そして，これら xml ファイルなどのリソースを直接利用するためのファイルとして R ファイルが存在します．前章でも説明した通りこのファイルは，リソースのインデックスファイルなので，このファイルのフィールドを参照して，リソースを扱います．

このように，レイアウトとアプリケーションの振る舞いを分離することで，Android では，UI 部分とアプリケーションの核となる動作部分の実装を分けて開発することが可能になってます．

2.2 Java ファイルから HelloWorld

今度は，Java ファイルを変更して HelloWorld を出力してみる．HelloWorld.java をリスト 4 のように変更する．その出力結果は図 4 のようになります．

リスト 4: HelloWorld.java(extended)

```
1  package fuji.android.tutorial;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5  import android.widget.TextView;
6
7  public class HelloWorld extends Activity {
8      /** Called when the activity is first created. */
9      @Override
10     public void onCreate(Bundle savedInstanceState) {
11         super.onCreate(savedInstanceState);
12         //setContentView(R.layout.main);
13         TextView tv = new TextView(this);
14         tv.setText("Hello World!!");
15         setContentView(tv);
16     }
17 }
```

図 4: Java から出力の結果

前回との違いは，レイアウトファイルを読み込んでません．リスト 4 の 13 行目で，リスト 2 にもあった TextView を作成します．さらに setText() で，TextView にテキストの内容をセットします．そして，setContentView() で TextView を表示します．

このように，Java ファイルから TextView などの UI 部品を作成することもできます．

2.3 レイアウトファイルから UI 部品の読み込み

2.1 では，画面のレイアウトはレイアウトファイルで設定し，Java ファイルでレイアウトファイルを読み込みのみを行いました．2.2 では，レイアウトファイルを利用せずに Java ファイルのみで，UI 部分の実装も行いました．

しかし，せっかくレイアウトファイルを作成して構造と振る舞いを分離したので，レイアウトファイルから読み込んだ UI 部品をオブジェクトとして利用できるようにしたいと思います．

本節では，レイアウトファイルから TextView オブジェクトを読み込むことで，レイアウトファイルのレイアウトを利用します．

そのために，HelloWorld.java と main.xml を次のように変更しました．

リスト 5: HelloWorld.java(more extended)

```
1  package fuji.android.tutorial;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5  import android.widget.TextView;
6
7  public class HelloWorld extends Activity {
8      /** Called when the activity is first created. */
9      @Override
10     public void onCreate(Bundle savedInstanceState) {
11         super.onCreate(savedInstanceState);
12         setContentView(R.layout.main);
13         TextView tv = (TextView) findViewById(R.id.text);
14         tv.setText("Hello World!!");
15     }
16 }
```

リスト 6: main.xml(extended)

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >
7  <TextView  android:id="@+id/text"
8      android:layout_width="fill_parent"
9      android:layout_height="wrap_content"
10     android:text="@string/hello"
11     />
12 </LinearLayout>
```

変更を加えた部分は，リスト 5 では 13 行目，リスト 6 では 7 行目になります．リスト 5 の findViewById() では，TextView を任意の ID からレイアウトファイルより読み込んでいます．またリスト 4 では，TextView をセットしていたが，リスト 5 では 12 行目ですでに全体をセットしているため，tv オブジェクトをセットする必要がありません．

リスト 5 の 13 行目で ID から TextView を読み込むために，レイアウトファイルの TextView に id 属性を追加しました．これは，“@+id/text” という部分で行っています．ここで，+ は新規作成を意味していて，この TextView に “text” をいう値を持つ “id” という属性を作成するというものです．これによって，text という id が作成され，Java ファイルから id を使った参照を行うことができるようになりした．

これで，レイアウトファイルの TextView を設定したので，id の TextView を tv オブジェクトに割り当てました．

これによって可能になるのは，UI 部品をオブジェクトとして管理・設定できることです．リスト 6 の 10 行目では，TextView の内容は，レイアウトファイルで “Hello World, HelloWorld” に設定されてましたが，リスト 5 の 14 行目でその内容を “Hello World” に変更しました．構造と振る舞いを分けたといいましたが，実際は，レイアウトファイルで静的なレイアウトを指定し，Java ファイルで動的にレイアウトを変更することができます．

Android アプリケーションの基本的な構造は，このようになっています．

3. ボタンを追加しよう

前章までで，Android アプリケーションの基本的な作成方法と構造を紹介しました．この章からは，その基本構造を拡張していきます．この章では，この画面にボタンを追加してみます．画面のレイアウトとしては，図 5 のようになります．

図 5: ボタンを追加した画面

前章で説明した通り，レイアウトの静的設定は．レイアウトファイルによって設定します．なので今回の変更点は，レイアウトファイルのみです．リスト 7 には，変更後のレイアウトファイルを示します．

リスト 7: レイアウトファイル

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >
7
8  <Button  android:id="@+id/button"
9      android:layout_width="wrap_content"
10     android:layout_height="wrap_content"
11     android:text="@string/button"
12     />
13
14 <TextView  android:id="@+id/text"
15     android:layout_width="fill_parent"
16     android:layout_height="wrap_content"
17     android:text="@string/hello"
18     />
19
```

```
20
21 </LinearLayout>
```

リスト 6 との変更点は，8 行目からの “Button タグ” が増えたことです．これによって，画面にボタンが追加されました．このボタンには，まだなにも動作を設定していないので，ボタンを押してもなにも起こりません．次の章で，このボタンに動作を設定していきます．

4. ボタンの動作を設定しよう

この章では，ボタンに動作を付加していきます．今回は，ボタンを押すたびに “HelloWorld!!” が数字と共に増えていく機能を付加します．動作を付加した画像とソースを以下に示します．

図 6: ボタンに動作を付加した画面

リスト 8: ボタンに動作を付加したソース

```
1  package fuji.android.tutorial;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5  import android.view.View;
6  import android.widget.Button;
7  import android.widget.TextView;
8
9  public class HelloWorld extends Activity {
10     /** Called when the activity is first created. */
11         TextView tv;
12         Button button;
13         int i = 1;
14
15     @Override
```

```
16      public void onCreate(Bundle savedInstanceState) {
17          super.onCreate(savedInstanceState);
18          setContentView(R.layout.main);
19          tv = (TextView) findViewById(R.id.text);
20          tv.setText("Hello World " + i + "!!");
21
22          button = (Button) findViewById(R.id.button);
23          button.setOnClickListener(new View.OnClickListener() {
24                  public void onClick(View view) {
25                          i++;
26                          tv.append("\nHello World " + i + " !!");
27                  }
28          });
29      }
30  }
```

今回変更したのは，主に 22 行目以降からです．まずボタンをオブジェクト化し，そのボタンが押されたときにイベントを受け取るようにします．そして，そのイベントを受け取ると実行される動作を 24 行目にメソッドに記述します．

このようにして，ボタンが押されたときに動作を指定します．ここで注意するのは，このメソッドを onCreate() 内に記述しなければならないことです．2.1 節でもいったように，Activity の初期設定を行う部分でアプリケーションが呼ばれたとき，一番最初に実行されるメソッドになります．なので，この onCreate() にボタンの動作などを記述します．

5. メニューボタンを作ろう

この章では，前節のボタンをメニューボタンに移行します．その画面とソースを以下に示します．

図 7: メニューボタンを付加した画面

リスト 9: メニューボタンを付加したソース

```
1  package fuji.android.tutorial;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5  import android.view.Menu;
6  import android.view.MenuItem;
7  import android.widget.Button;
8  import android.widget.TextView;
9
10 public class HelloWorld extends Activity {
11     /** Called when the activity is first created. */
12         TextView tv;
13         Button button;
14         int i = 1;
15
16     @Override
17     public void onCreate(Bundle savedInstanceState) {
18         super.onCreate(savedInstanceState);
19         setContentView(R.layout.main);
20         tv = (TextView) findViewById(R.id.text);
21         tv.setText("Hello World " + i + "!!");
22     }
23
24     @Override
25     public boolean onCreateOptionsMenu(Menu menu) {
26         super.onCreateOptionsMenu(menu);
27         menu.add(0, Menu.FIRST, 0, R.string.button);
28         return true;
29     }
30
31     @Override
32     public boolean onMenuItemSelected(int featureId, MenuItem item) {
33         switch(item.getItemId()) {
34         case Menu.FIRST:
35                 i++;
36                 tv.append("\nHello World " + i + "!!");
37                 return true;
38         }
39
40         return super.onMenuItemSelected(featureId, item);
41     }
42
43 }
```

この章での変更は，リスト 9 の 25 行目からです．ここでは，レイアウトファイルの Button タグの代わりに “onCreateOptionsMenu()“を，onClickListener() の代わりに “onMenuItemSelected()“を追加しました．この二つのメソッドは，メニューボタンに関するメソッドで，前者はメニューボタンを作成するために利用され，後者は押されたメニューボタン別の動作を指定しています．

今回は，メニューボタンを押すと “Button” というメニューボタンが現れ，そのボタンを押すと前章のボタンと同じ動作を行います．また，どのボタンが押されたかは，メニューボタンに割り振られた ID によってわかります．その ID は，リスト 9 の 27 行目にあるメニューボタンを追加するメソッドの第二引数によって指定します．そして，メニューボタンが押されると押されたメニューボタンのオブジェクトを onMenuItemSelected() の引数に渡します．そのオブジェクトの ID を利用してボタンにあった動作を指定しています．リスト 9 では，33 行目以降の switch 文がそれにあたります．

6. Activity を増やしてみよう

では最後に Activity を増やしてみたいと思います．Activitiy は画面です．1 つのアプリケーションに画面が一つだけなんてことは，そうそうありません．

今回は，文字を表示する画面 (Display.java) と文字を入力する画面 (Edit.java) の 2 つの Activity を使ったアプリケーションを作成したいと思います．

6.1 文字出力画面 – Display.java –

まず，文字を出力する画面を構成します．Display の画面とレイアウトファイルは，以下の通り．

図 8: 文字出力画面

リスト 10: 文字出力画面のレイアウトファイル

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3          android:orientation="vertical"
4          android:layout_width="fill_parent"
5          android:layout_height="fill_parent"
6          >
7
8          <TextView
9          android:id="@+id/text"
10         android:layout_width="fill_parent"
11         android:layout_height="wrap_content"
12         android:layout_weight="1"
13         android:scrollbars="vertical"
14         />
15
16 </LinearLayout>
```

図 8 には，まだなにも表示されていませんが，ここには，文字入力画面で入力した文字を出力することになります．画面構成としては，TextView が一つあるだけの構成になっています．

このレイアウトファイルを作成するには，いくつかの方法があります．

- 自分で xml を記述
- DroidDraw を利用

この 2 つの違いは，エディタが GUI がです．前者は Eclipse のエディタなどを利用して人手で記述します．後者は，GUI を利用して画面を構成し，その画面のレイアウトと同じレイアウトファイルを自動生成します．今回は DroidDraw を利用して作成しました．

文字入力画面に移動するための方法として，メニューボタンを追加しました．画面下にある “EDIT” を押すと，文字入力画面に移動します．次のリスト 11 は，文字出力画面のソースです．

リスト 11: 文字出力画面の Java ファイル

```
1  package fuji.tutorial.display;
2
3  import android.app.Activity;
4  import android.content.Intent;
5  import android.os.Bundle;
6  import android.view.Menu;
7  import android.view.MenuItem;
8  import android.widget.TextView;
9
10 public class Display extends Activity {
11         private static final int EDIT = Menu.FIRST;
12         private static final int ACTIVITY_EDIT = 0;
13         private TextView tv;
14
15     /** Called when the activity is first created. */
16     @Override
17     public void onCreate(Bundle savedInstanceState) {
18         super.onCreate(savedInstanceState);
19         setContentView(R.layout.display);
20         tv = (TextView) findViewById(R.id.text);
21     }
22
23     @Override
24     public boolean onCreateOptionsMenu(Menu menu) {
25         super.onCreateOptionsMenu(menu);
26         menu.add(0, EDIT, 0, R.string.button);
27         return true;
28     }
29
30     @Override
31     public boolean onMenuItemSelected(int featureId, MenuItem item) {
32         switch (item.getItemId()) {
33         case EDIT:
34                 edit();
35                 return true;
36         }
37         return super.onMenuItemSelected(featureId, item);
38     }
39
40     private void edit() {
41         Bundle bundle = new Bundle();
42         bundle.putString("string", tv.getText().toString());
43
44         Intent i = new Intent(this, Edit.class);
45         if (bundle != null) {
46                 i.putExtras(bundle);
47         }
```

```
48          startActivityForResult(i, ACTIVITY_EDIT);
49      }
50
51      @Override
52      public void onActivityResult(int requestCode, int resultCode, Intent intent) {
53          super.onActivityResult(requestCode, resultCode, intent);
54          Bundle extras = intent.getExtras();
55
56          switch (requestCode) {
57          case ACTIVITY_EDIT:
58                  String text = extras.getString("string");
59                  if (text != null) {
60                          tv.setText(text);
61                  }
62                  break;
63          }
64      }
65  }
```

では，各メソッドについて見ていきましょう．

onCreate()

ここでは，この Activity のメインメソッドの役割を行います．

l.19 レイアウトファイルの読み込み

l.20 TextView オブジェクトの作成

onCreateOptionsMenu()

ここでは，メニューボタンの作成を行います．

l.26 EDIT ボタンを作成

このメソッドでは，4 つの引数を渡していますが，注目すべきは，値が 0 以外の 2 つの引数です．“EDIT” では，メニューボタンの ID を指定しており，“R.string.button” では，ボタンに表示する文字を指定しています．ここで表示される文字は，“EDIT” です．

onMenuItemSelected()

ここでは，メニューボタンが押されたときの動作を指定します．どのボタンが押されたかの判断は，引数に渡される “item” の ID で判断します．この ID は，前の小節で指定したものと同一の ID です．

l.32 - l.36 各ボタンの動作の指定

edit()

ここでは，EDIT ボタンが押された時の動作が実装されています．流れとしては，Bundle に表示している文字列を格納して，その Bundle を Intent に付属させて，別の Activity を起動させます．Intent とは，Activity 間のやりとりを行うクラスで，今回のように情報を Bundle に格納して

送ったり，単に別の Activity を呼び出したりします．Activity を呼び出す時には，この Intent を利用して実行します．

l.41, 42 TextView に表示されている文字列を Bundle に格納

l.44 Intent に呼び出す Activity と呼び出される Activity を登録

l.45 - l.47 Intent に Bundle を追加

l.48 Avtivity の呼び出し

48 行目のメソッドでは，Activity を呼び出すために，Intent と requestCode(ATIVITY_EDIT) を引数に渡しています．Intent は，先ほどいったように，呼び出す側と呼び出される側の指定がです．requestCode は，呼び出したメソッドの ID として扱われるものです．返り値を受け取った時に，この requestCode を利用してどのメソッドが呼び出した Activty からの返り値かを判断します．

onActivityResult()

このメソッドは，呼び出した Avtivty が終了した時に呼ばれるメソッドで．Activity の結果を取得します．

ここでは，文字入力画面で入力された文字を文字出力画面で出力するために，TextView に受け取った文字列をセットしています．そして，switch 文を使い，reqestCode ごとに動作を指定します．

l.54 送られてきた Intent から Bundle を取得

l.56, 57 switch を利用した requestCode で分類

l.58 送られてきた Bundle から id “string” の文字列を取得

l.59 - l.61 文字列が null でなかったら，TextView に文字列をセット

ここまでが，文字出力画面の振る舞いである．次に呼び出される側である，文字入力画面の Activity を作成する．

6.2 文字入力画面 − Edit.java −

次に，文字を入力する画面を構成します．Edit の画面とレイアウトは，次に示した通り．

図 9: 文字入力画面

リスト 12: 文字入力画面のレイアウトファイル

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

        <EditText
                android:id="@+id/edit"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:scrollbars="vertical"
                />

        <Button
                android:id="@+id/confirm"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/confirm"
                />

</LinearLayout>
```

この画面では，文字の入力・編集のできる “EditText” と Button で構成されている．EditText 部分に文字列を入力し，Button を押すと，文字出力画面で出力するため，Display Activity へ文字列を返り値として返す．リスト 13 に Edit.java を示す．

リスト 13: 文字入力画面の Java ファイル

```
1  package fuji.tutorial.display;
2
3  import android.app.Activity;
4  import android.content.Intent;
5  import android.os.Bundle;
6  import android.view.View;
7  import android.widget.Button;
8  import android.widget.EditText;
9
10 public class Edit extends Activity {
11         EditText et;
12         Button confirm;
13
14         @Override
15         protected void onCreate(Bundle savedInstanceState) {
16                 super.onCreate(savedInstanceState);
17                 setContentView(R.layout.edit);
18                 et = (EditText) findViewById(R.id.edit);
19                 confirm = (Button) findViewById(R.id.confirm);
20
21                 Bundle extras = getIntent().getExtras();
22                 if (extras != null) {
23                         String text = extras.getString("string");
24
25                         if (text != null) {
26                                 et.setText(text);
27                         }
28                 }
29
30                 confirm.setOnClickListener(new View.OnClickListener() {
31                         public void onClick(View view) {
32                                 sendText();
33                         }
34                 });
35         }
36
37         private void sendText() {
38                 Bundle bundle = new Bundle();
39                 bundle.putString("string", et.getText().toString());
40
41                 Intent mIntent = new Intent();
42                 mIntent.putExtras(bundle);
43                 setResult(RESULT_OK, mIntent);
44                 finish();
45         }
46 }
```

ここではの流れは，onCreate() で，送られてきた Intent から Bundle を取得し，その中の文字列を EditText にセットする．また，ボタンを押されたときの動作は，EditText の文字列を Bundle にセットし，Intent に格納．そして，Intent を結果としてセットし，終了する．

各メソッドを詳しく見ていく．

onCreate()

ここでは，この Activity の初期化を行っている．ここで行うことは三つ．

1. レイアウトファイルから情報を読み込む (l.17 - l.19)
2. Bundle から文字列を取得 (l.21 - l.28)

3. ボタンの動作を指定 (l.30 - l.34)

Activity 間でのデータの受け渡しは，Intent と Bundle を利用することは，前節と本説でよくわかる．このほかにもデータの受け渡しの方法としては，データベースやローカルファイルを利用した方法もあるが，それはまた次回に解説する．

sendText()

ここでは，ボタンが押されたときの動作が記述してある．このメソッドの流れとしては，以下のようになる．

1. Bundle に EditText の文字列をセット (l.38 - l.39)
2. Intent に Bundle をセット (l.41 - l.42)
3. 結果として送る Intent と resultCode(RESULT_OK) をセット (l.43)
4. Activity の終了 (l.44)

ここで作成される Intent には，引数を渡さない．これは呼び出すための Intent ではなく，返り値として Intent を呼び出しもとへ渡すためである．また，(3) で渡した resultCode は, この返り値が正常終了かどうかを示している．この結果を利用して呼び出し元の Activity は，処理を変更することができる．

6.3 マニフェストの変更

前節までで，2 つの Activity を作成し，Activity 間でデータの受け渡しや振る舞いなどを決定してきた．しかしこのままでは，このアプリケーションを利用することはできない．最後に変更するのは，“Android Manifest” である．このマニフェストの変更点は，利用する Activity の登録である．この操作を行わないと，Edit Activity を利用するとシステムに認識してもらえず実行することができない．

なので，以下のリスト 14 のように，マニフェストを変更する．

リスト 14: AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="fuji.tutorial.display">
4      <application android:icon="@drawable/icon" android:label="@string/app_name">
5          <activity android:name=".Display" android:label="@string/app_name">
6              <intent-filter>
7                  <action android:name="android.intent.action.MAIN" />
8                  <category android:name="android.intent.category.LAUNCHER" />
9              </intent-filter>
10         </activity>
11     <activity android:name=".Edit"></activity>
12 </application>
13 </manifest>
```

マニフェストには，さまざまな情報を記述する．主な内容は，以下のようになっている．

- アプリケーションのソースが格納されているパッケージ名 (l.3)
- HOME で表示されるアイコン画像とアプリケーション名 (l.4)
- アプリケーションが利用する Activity(l.5 - l.11)

リスト 14 の中で，変更した部分は 11 行目だけである．このように，利用する Activity 名を登録しておかないと，利用しないものだと認識されてしまうため，Activity を複数利用する場合には，この変更を必ず行ってください．

7 まとめ

今回のチュートリアルでは，以下のことを説明しました．

- 開発環境の整備
- プロジェクトの作成方法
- Activity の利用法
- レイアウトファイルからの読み込みとオブジェクト生成
- メニューボタンの設定
- 複数 Activity 利用法

今回は第 1 回目なので，Activity を使った文字の入出力を中心に行いました．しかし，Android で開発で開発できるのは，文字を扱うものだけでなく，画像，音声，ゲームなどざまざまな分野があります．次回は，これらに注目したものかもしれませんし，他のものかもしれません．まだ，どんなチュートリアルになるかわかりませんが，よかったらお付き合いを．

参考文献

[1] Android SDK: `http://code.google.com/android/`

[2] Eclipse: `http://www.eclipse.org/`

[3] ADT: `http://code.google.com/android/intro/installing.html`

変更履歴

2008/10/21：Ver 1.0：公開開始
2008/10/30：Ver 1.1：誤字修正，URL 追加, メールアドレス追加