

Android 入門カリキュラム I

講師: 原 弘
Twitter: @NowDeveloping

サポート: 大貫 真由

講義カリキュラム

- 第一回
 - 環境構築について
 - Activityとライフサイクルについて
 - Intentについて
- 第二回 Sample Projectの構成について説明
 - Layoutについて
 - Viewコンポーネント その1
 - Viewコンポーネント その2
- 第三回
 - SQLiteについて
 - アプリ制作のアイデア出しとグループ分け
- 第四回・第五回
 - アプリ制作
 - アプリ発表(時間があれば)

環境構築

1. JDK (Java Development Kit) のセットアップ
2. Android SDK のセットアップ
3. Eclipse のセットアップ

1. JDK (Java Development Kit) のセットアップ

1. Sun Developer Network のサイト (<http://java.sun.com/javase/ja/6/download.html>) にアクセスします。
2. ダウンロードしたい Platform (Windowsなど) を選択し、「Download」をクリックします。
3. 「Log In for Download (Optional)」が表示された場合、「Skip this Step」をクリックします。
4. 「jdk-6u24-windows-i586.exe」をクリックし、ダウンロードします。
5. ダウンロードしたインストーラー (jdk-6u24-windows-i586.exe) を実行し、JDK をインストールします。
6. Java の実行ファイルへのパスはインストーラーによって自動的に設定されますが、そうでない場合には適宜設定してください

2. Android SDKのセットアップ

1. Android Developersのサイト (<http://developer.android.com/sdk/index.html>) にアクセスします。
2. ダウンロードしたい Platform (Windowsなど) を選択し、「android-sdk_r10-windows.zip」をクリックします。
3. ダウンロードした圧縮ファイルを、任意の場所に展開します。
4. ここでは「c:¥android」ディレクトリを作成し、その下に展開しています (C:¥android¥android-sdk-windows)
5. 展開したディレクトリ下の「tools」にある「SDK Setup.exe」を実行します。(「SDK Setup.exe」がない環境 (Mac/Linux) では、「tools」下にある「android」コマンドを実行します。)

3. Eclipseのセットアップ

1. Eclipseのサイト(<http://www.eclipse.org/downloads/>)にアクセスし、ダウンロードします。
2. ダウンロードした圧縮ファイルを、任意の場所に展開します。
3. ここでは c ドライブ直下に展開しています (C:\eclipse)
4. Eclipse を起動し、Workspace 設定画面が表示されたら、「OK」をクリックします。
5. Workspace とは、プロジェクトを保存するディレクトリであり、任意の場所に変更してもいいですし、そのままでも問題ないです

3. Eclipseのセットアップ

6. 以下の手順で ADT Plugin (Android Development Tools Plugin) をインストールします。
 1. メニュー、「Install New Software...」をクリック
 2. 「Install」画面の「Add...」ボタンをクリック
 3. 「Add Site」画面の「Name」に「Android Plugin」、「Location」に「<https://dl-ssl.google.com/android/eclipse/>」を入力し、「OK」をクリック
 4. 「Developer Tools」をチェックし、「Next」をクリック
 5. ライセンス規約を確認し、同意できれば、「I accept the terms of the license agreements」をクリック
 6. 「Finish」をクリックし、ADT Plugin をインストールします

※途中、署名や再起動の確認がありますが、内容を確認し、処理を進めてください

※「<https://dl-ssl.google.com/android/eclipse/>」の接続に失敗する場合は、
「<http://dl-ssl.google.com/android/eclipse/>」にしてください

※Eclipse の HTTP Proxy の設定は、

「Window」メニュー、「Preferences」-「General」-「Network Connection」から設定できます

3. Eclipseのセットアップ

7. 以下の手順でAndroid SDK の設定をします。

1. 「Window」メニュー、「Preferences」をクリック
2. Preferences」画面の左側ペインで「Android」を選択
3. 「SDK Location」にインストールした Android SDK のディレクトリを入力
（「Browse」からも選択できます）
4. 「Apply」「OK」をクリックします。

Activityとライフサイクルについて

1. Activityとは
2. ライフサイクルとは
3. ライフサイクルの確認

1. Activityとは

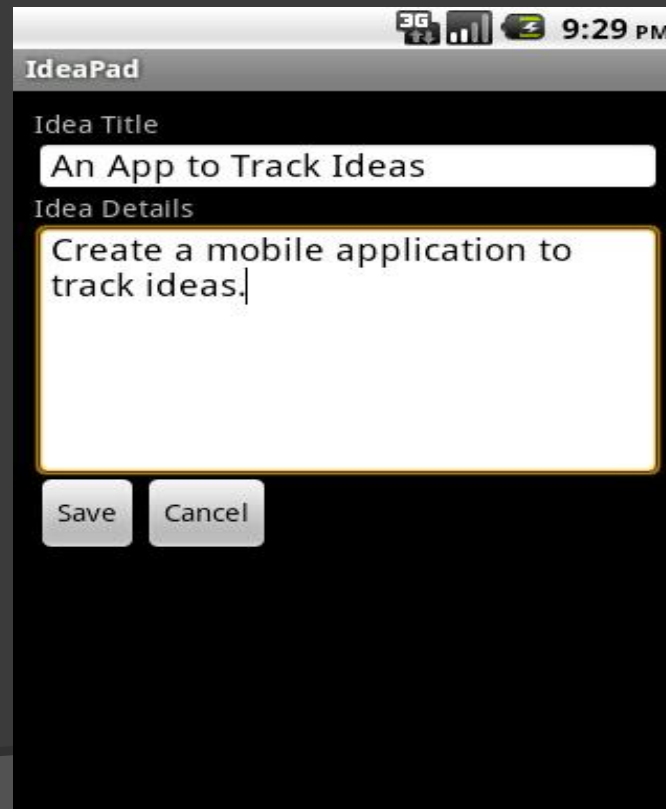
“Androidアプリの画面”に相当します。ボタンやリストが配置された画面、Webページが表示されている画面、3Dグラフィックスが表示されている画面などはそれぞれがActivityです。

Androidは画面を持たないアプリも作成可能で、その場合はActivityを使用しないのですが、最初は画面のあるアプリから徐々に慣れていきましょう。

Activityはさまざまな表示を行ったり、ユーザーと対話したり、イベントを受け取ったりといろいろなことが可能ですが今回はまず初めに知っておいてもらいたいライフサイクルを見ていきます。

図 AndroidのActivity

実習：Activityの確認



The screenshot shows an Android application interface. At the top, there's a status bar with '3G', signal strength, battery, and the time '9:29 PM'. Below that is a title bar labeled 'IdeaPad'. The main content area has a dark background. It contains a section titled 'Idea Title' with a text input field containing 'An App to Track Ideas'. Below this is a section titled 'Idea Details' with a larger text input field containing 'Create a mobile application to track ideas.'. At the bottom of the form are two buttons: 'Save' and 'Cancel'.

2. ライフサイクルとは

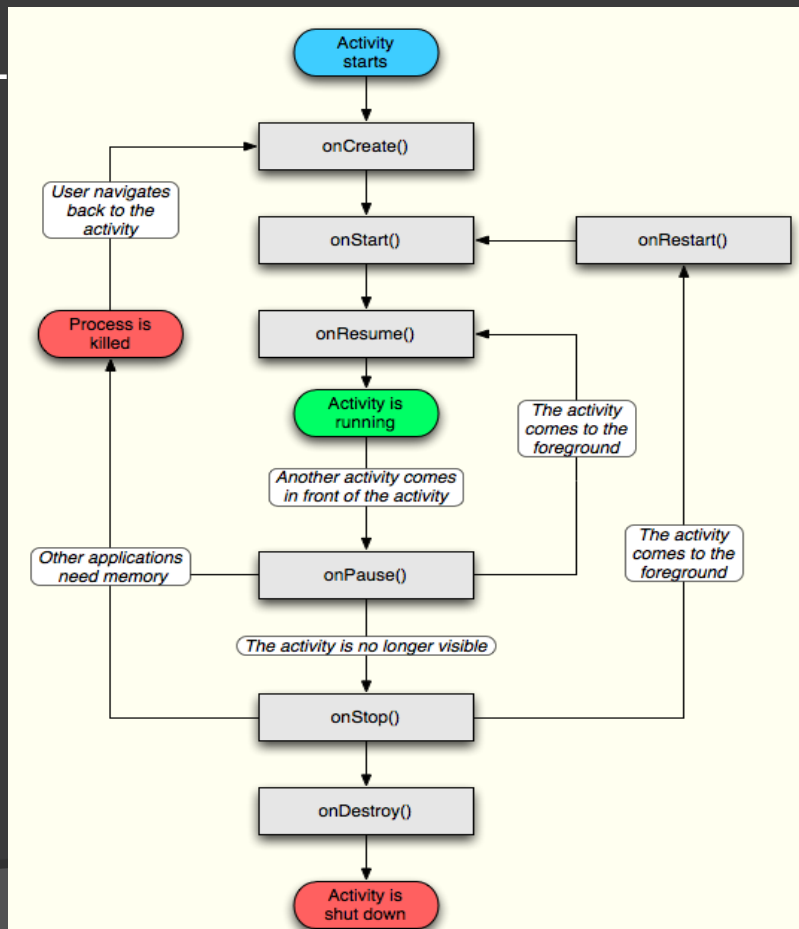
アプリケーションには“始まり”があって、“終わり”があります。始まってから終わるまでを「ライフサイクル」(状態遷移)といい、Activityにも同様にライフサイクルがあります。

ライフサイクルを分かりやすく図にすると

「状態遷移図」(フローチャート図)というものになります。

今回はこの状態遷移図を使用してActivityのライフサイクルを説明していきます。

図 ライフサイクルのフロー



3. ライフサイクルの確認

実際にエミュレーターでライフサイクルのタイミングの確認を行ってみましょう。

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

/**
 * 起動クラス
 */
public class Sample extends Activity {

    /**
     * Activityが起動すると一番最初に呼び出される
     */
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Toast.makeText(this, "onCreate()", Toast.LENGTH_LONG).show();
    }
}
```

Activityつまり画面を保持したアプリケーションを開発するには、Activityを継承(Extends)したクラスを作成します。

Sample.javaでは 10行目で、public class Sample extends Activityとして、Activityを継承(Extends)したSampleという名前のクラスを生成しています。

Activityを継承したクラスでは、起動直後にonCreate()が呼び出されます。Sample.javaでは、Toast.makeText(this, "onCreate()", Toast.LENGTH_LONG).show();として、Toastを呼び出し、onCreate()が呼び出されたことを確認します。

こちらを確認したらフローに習ってライフサイクルの流れを見てみましょう

```
/**
 * 起動クラス
 */
public class Sample extends Activity {

    /**
     * Activityが起動すると一番最初に呼び出される
     */
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        Toast.makeText(this, "onCreate()", Toast.LENGTH_LONG).show();
    }

    /**
     * onStart
     */
    public void onStart(){
        super.onStart();
        Toast.makeText(this, "onStart()", Toast.LENGTH_LONG).show();
    }

    /**
     * onResume
     */
    public void onResume(){
        super.onResume();
        Toast.makeText(this, "onResume()", Toast.LENGTH_LONG).show();
    }

    /**
     * onPause()
     */
    public void onPause(){
        super.onPause();
        Toast.makeText(this, "onPause()", Toast.LENGTH_LONG).show();
    }

    /**
     * onStop()
     */
    public void onStop(){
        super.onPause();
        Toast.makeText(this, "onStop()", Toast.LENGTH_LONG).show();
    }

    /**
     * onDestroy()
     */
    public void onDestroy(){
        super.onPause();
        Toast.makeText(this, "onDestroy()", Toast.LENGTH_LONG).show();
    }
}
```

Intentについて

1. Intentの目的
2. 明示的Intent、暗黙的Intent
3. Intent-filter
4. startActivity()とstartActivityForResult()

1. Intentの目的

他のアプリやデバイス(カメラや加速度センサ等)から情報を得ようとする、プロセスやスレッドを跨いだ処理を考慮しなくてはならず、これが非常にめんどくさいです。

このやりとりを簡単に実現するために登場したのがIntentで、ブラウザ立ち上げたとか、コール中といった情報をintentで通知する、いわゆるeventのような位置付けになります。

2.明示的Intent、暗黙的Intent

Intentを作成する時には大きく分けて二通りの方法があります。

一つは直接クラスを指定して呼び出す方法で、これを明示的Intentといいます。

もう一つはこうして欲しいという「意図」をIntentに情報として付与し、システムに誰に処理してもらうかを決めてもらうやり方で、これを暗黙的Intentといいます。

また、電話機能から「今電話かかってきたけど何かする？」という暗黙的Intentやシステムから「今起動したけど何かする？」というIntentが送付されてきます。

これらのIntentを処理する際に必要になるのが次のIntent-filterです。

3. Intent-filter

外部からのIntentを処理するかどうかはAndroidManifest.xmlのIntent-filterにて宣言します
例えばhelloworldのAndroidManifest.xmlを見てみます。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="jp.demo.helloworld"
android:versionCode="1"
android:versionName="1.0.0">
<application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".helloworld"
        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>
```

まず、「android.intent.action.MAIN」ですが、これはactionタグの親のactivityが、このアプリケーションのエントリーポイント、開始画面として設定されていることを表します。ランチャーからアプリケーションを起動して、一番最初に表示されるわけです。ランチャーを叩くと、ACTIONが「ACTION_MAIN」であるIntentがアプリケーションに対して投げられ、それを処理できる「android.intent.action.MAIN」のintent-filterを持つactivityがintentを受理し、画面が起動するという流れになります。

次に赤字の部分の「android.intent.category.LAUNCHER」ですが、categoryというのはActionを受理する際に付加的な情報を付け加えます。ここでのcategoryは「LAUNCHER」ですが、これはシステムのLAUNCHERに登録できることを示しています。

4. startActivity()とstartActivityForResult()

同一アプリにおいて、別の画面を呼び出すときは明示的Intentを作成し、startActivity()の引数として引き渡します。

例えば以下のように書けます。

```
Intent intent = new Intent(helloworld.this,XXXXX.class);  
intent.setAction(Intent.ACTION_VIEW);  
startActivity(intent);
```

コンストラクタで自分自身、そして呼び出し先 (XXXXX.class)のクラスを指定します。ACTIONにはActivityを表示させるのでACTION_VIEWを指定しています。

これで明示的なIntentの送付完了です。

ただし、このstartActivity()では、遷移先のActivityから情報を引き取れません。

情報を返して欲しいときはstartActivityForResult()を使うと良いでしょう。

実習：画面遷移

Layoutについて

1. LayOutとは
2. Eclipseの上でのレイアウト変更ツール
3. xmlでのレイアウト変更

1. Layoutとは

Androidアプリケーションでは、XMLを用いてUIを作成することが可能です。
UIの作成は、エディタを使用しての作成が可能です。
次の項ではEclipseを用いてLayoutを作成します。

■ HelloWorldの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"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```

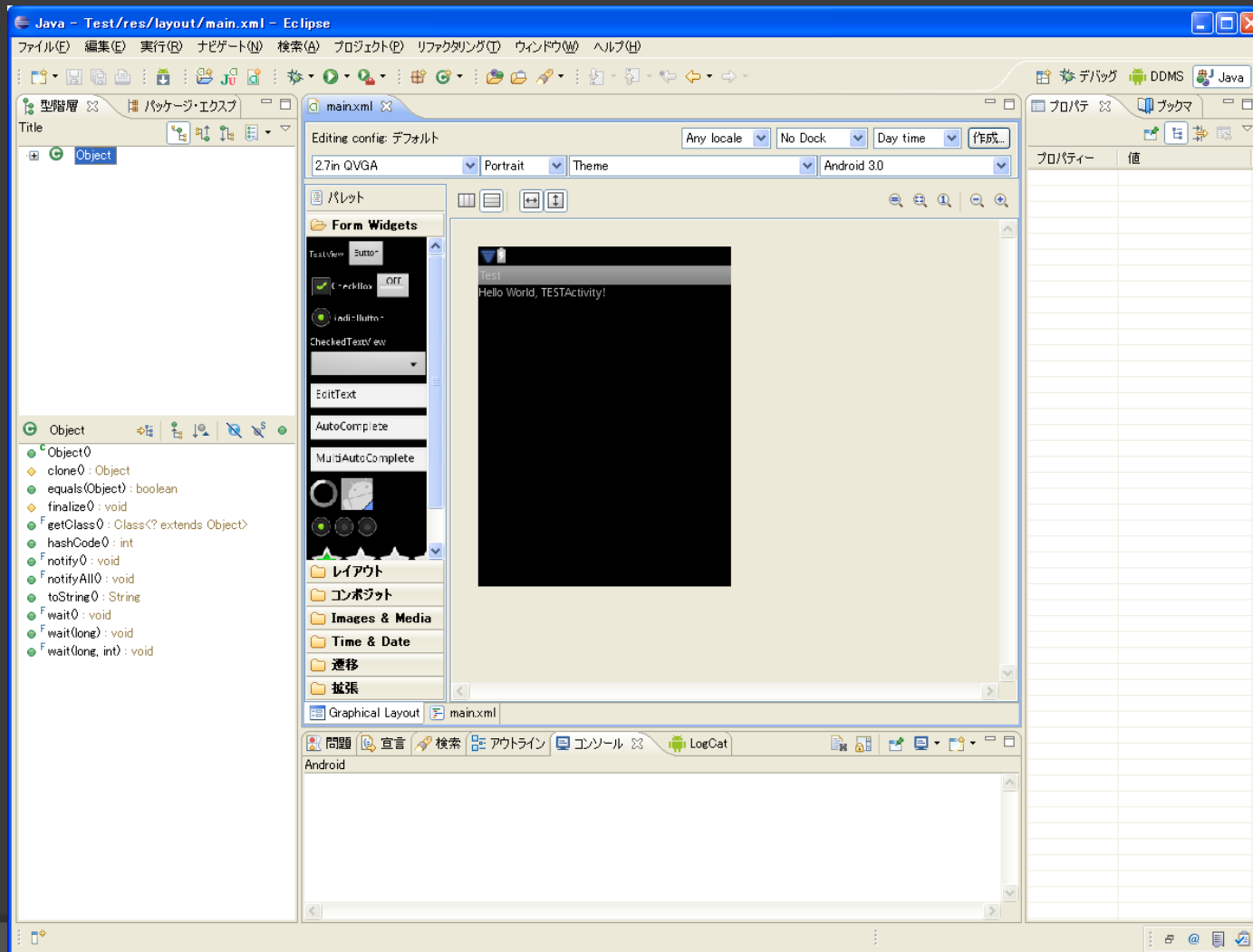
2. Eclipseの上でのレイアウト変更ツール

レイアウトで使用するXMLはEclipseで変更・操作を行うことができます。

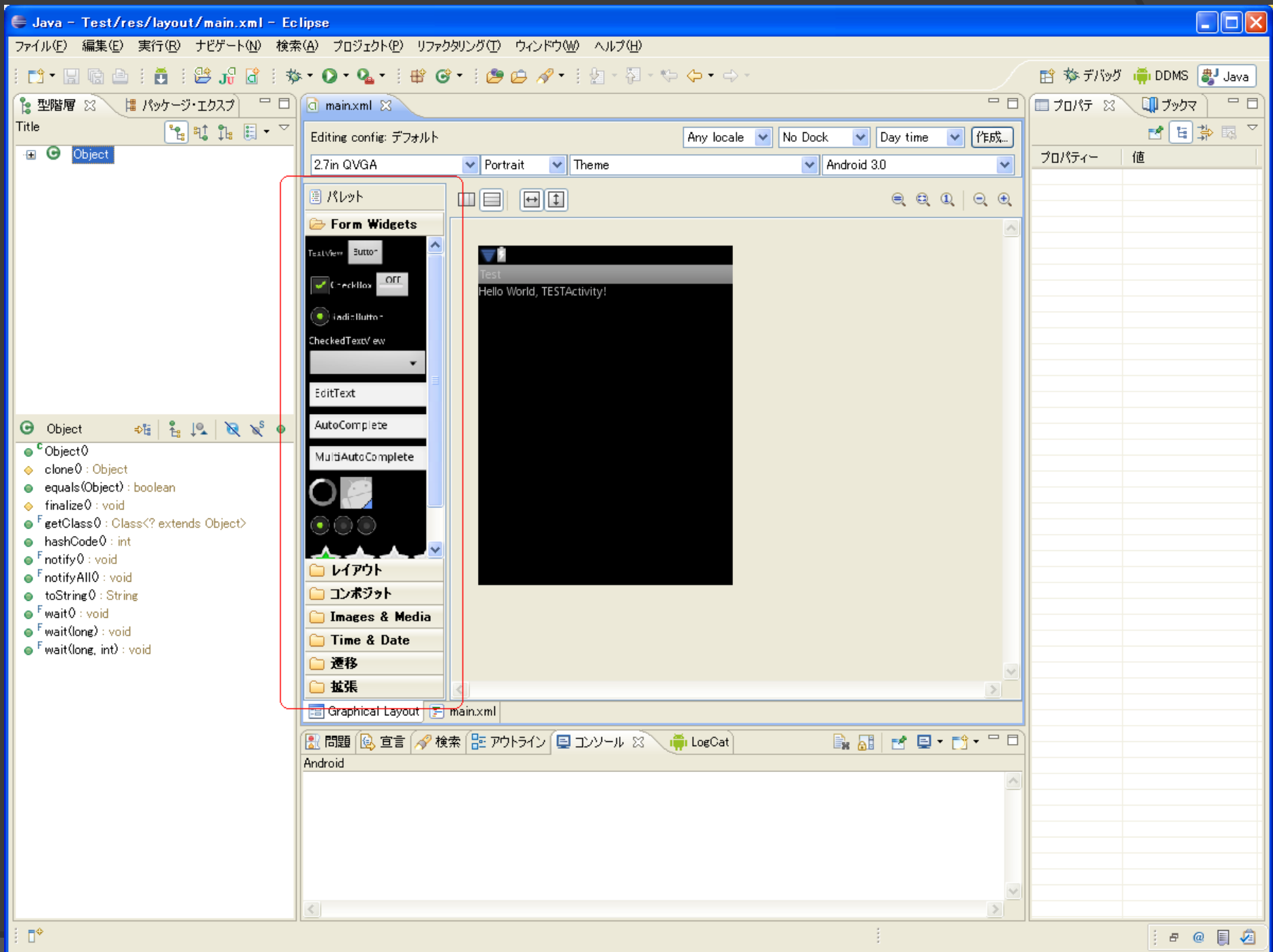
それではレイアウトの追加を行ってみます。

(1).res/main.xmlを、Android Layout Editorで開き、layoutタブを開きます。

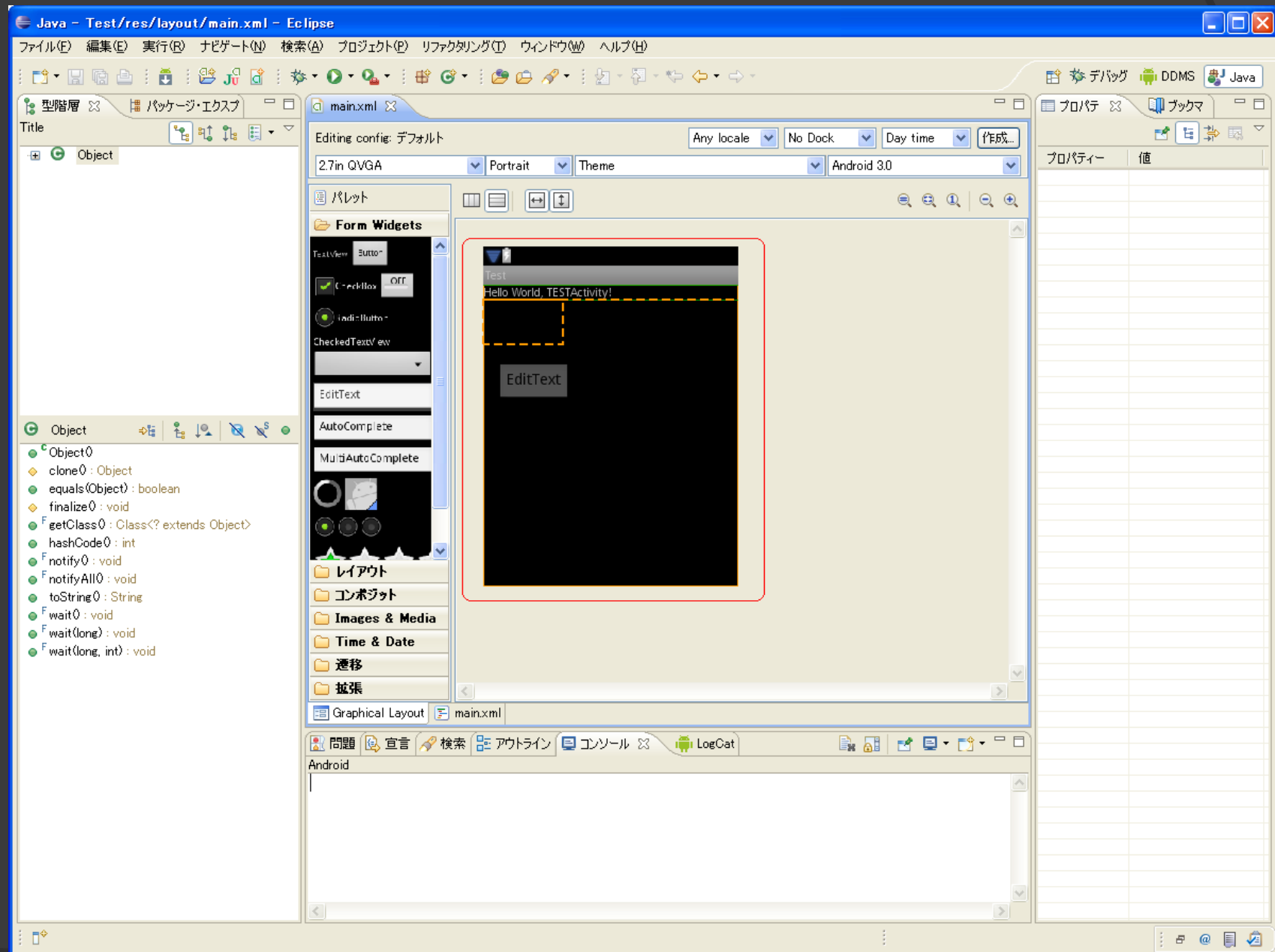
すると、以下のような画面が開きます。



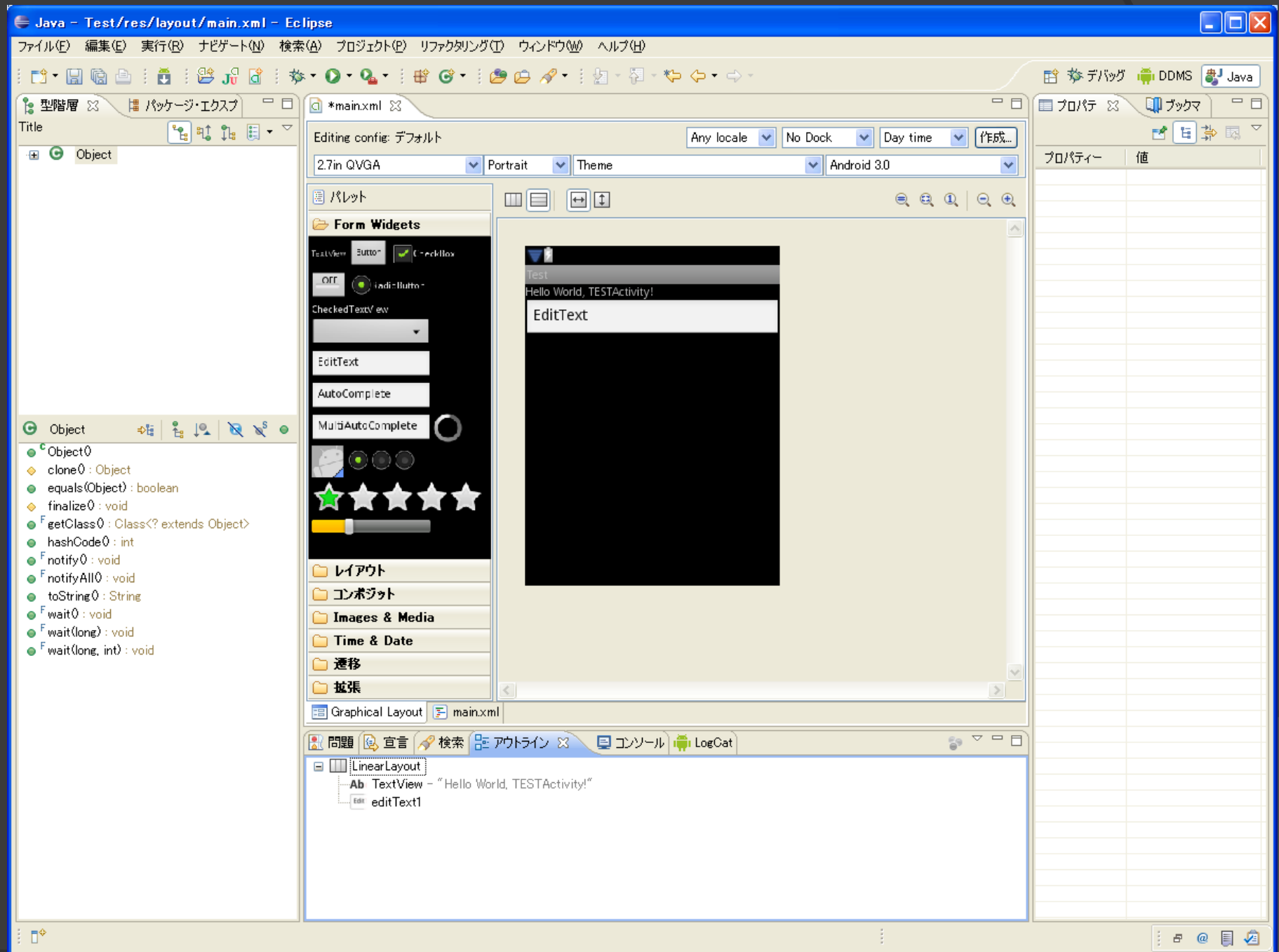
(2).画面赤く囲われた部分「パレット」から追加したい項目を選択(ドラッグ)します。



(3).すると、以下の画面のように、ドラッグ状態になりますので赤く囲われた部分にドロップしてください。



(4).これで、以下の画面のように、TextViewの下に、EditTextが出来ます。



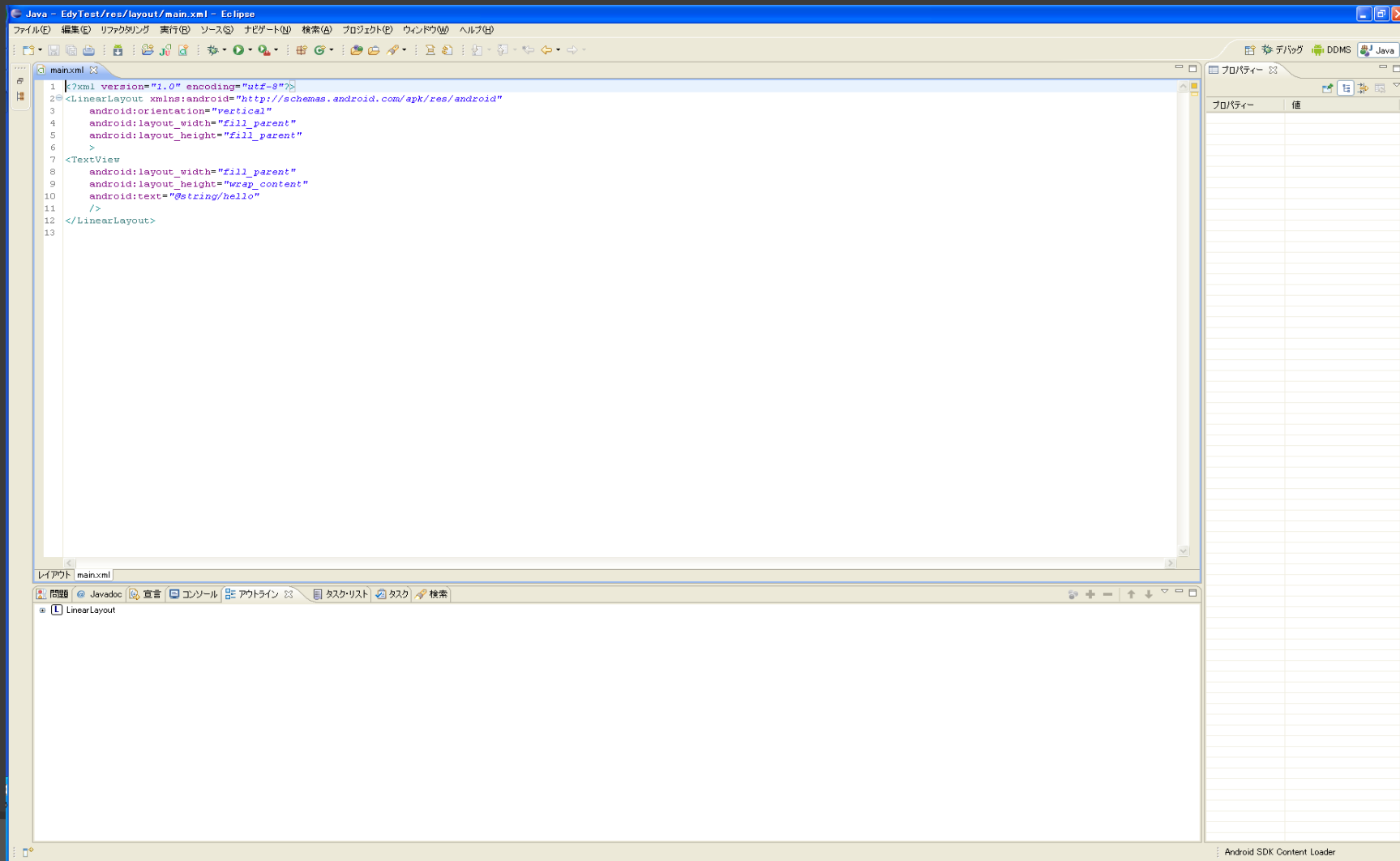
3. xmlでのレイアウト変更

レイアウトで使用するXMLはEclipseで変更・操作を行うことができます。

それではレイアウトの追加を行ってみます。

res/main.xmlを、Android Layout Editorで開き、main.xmlタブを開きます。

すると、以下のような画面が開きますので直接書式を入力することができます。



Viewコンポーネント その1

1. TextViewとは
2. EditTextとは
3. 実践

1. TextViewとは

TextViewクラスは文字列などの表示を行うためのコンポーネントタイプのビューです。

ButtonクラスやEditTextクラスの基本クラスでもあります。

TextViewクラスの定義を確認します。クラス図は次のようになっています。

```
java.lang.Object
└─ android.view.View
    └─ android.widget.TextView
```

TextViewクラスはViewクラスのサブクラスとなっています。

2. EditTextとは

EditTextクラスはユーザーからの入力を行ってもらうためのテキストボックスを提供するコンポーネントタイプのビューです。

EditTextクラスの定義を確認します。クラス図は次のようになっています。

```
java.lang.Object
└─ android.view.View
    └─ android.widget.TextView
        └─ android.widget.EditText
```

EditTextクラスはTextViewクラスのサブクラスとなっています。

3. 実践

それでは実際に前項のLayoutの追加方法に基づき
「TextView」と「EditText」を所定の位置に配置してください。



Viewコンポーネント その2

1. Buttonとは
2. ハンドラとは
3. 実践

1. Buttonとは

Buttonクラスはユーザーがクリックすることによって何らかのアクションを行うトリガーとなるコンポーネントタイプのビューです。

Buttonクラスの定義を確認します。クラス図は次のようになっています。

```
java.lang.Object
├── android.view.View
│   ├── android.widget.TextView
│   │   └── android.widget.Button
```

ButtonクラスはTextViewクラスのサブクラスとなっています。

2. ハンドラとは

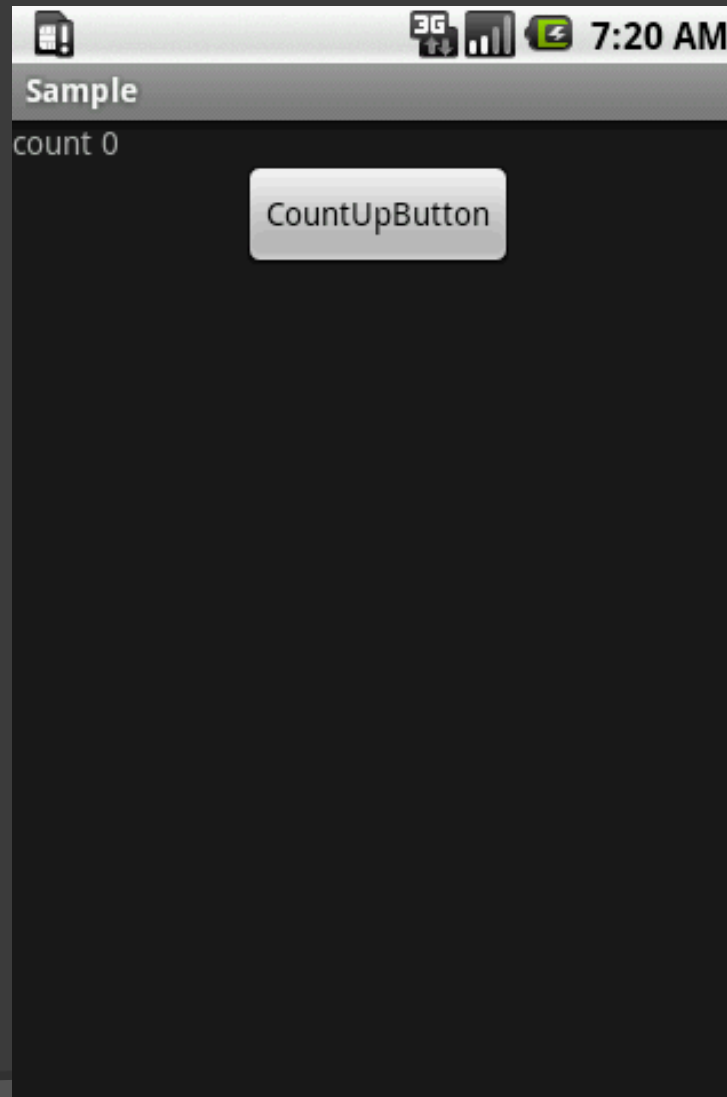
Androidでは任意のタイミングで処理を実行したい際にHandlerクラスを使用します。
Handlerクラス使用の際にはHandlerクラスを継承して利用します。

宣言の例)

```
Handler mHandler = new Handler();
```


3. 実践

それではButtonとハンドラを使用し、ボタンを押下した際に画面が変化するアプリケーションを作成します。



3. 実践

まずはLayoutツールを用いてButtonを画面に配置します。
これでボタンは配置されましたが、ボタンとして使うにはクリック時にイベントを取得できるようにしなければいけません。

```
public class Test extends Activity ①implements OnClickListener
    @Override protected void onCreate(Bundle icle) {
        super.onCreate(icle);

        ③ Button button = (Button)findViewById(R.id.Button01);
        ④ button.setOnClickListener(this);
    }

    ②public void onClick(View v) {
        /* .... */
    }
}
```

ボタンのクリックイベントを追加するために赤字の部分を追加します。

- ①Activityに「OnClickListener」インタフェースを実装します。
- ②「OnClickListener」インタフェースに必要な「onClick」メソッドを定義します。
- ③ボタンを定義します、ここでLayoutツールにて作成したButtonと関連付けをします。
- ④定義したボタンに「OnClickListener」インタフェースをセットします。

3. 実践

ボタンの準備は整ったので次はハンドラを使用します。

```
public class Test extends Activity implements OnClickListener{
    ①Handler handler=new Handler();
    @Override protected void onCreate(Bundle icle) {
        super.onCreate(icle);

        Button button = (Button)findViewById(R.id.Button01);
        button.setOnClickListener(this);
    }

    public void onClick(View v) {
        ②mHandler.post(new Runnable() {
            public void run() {
                //ここに処理を入れる
            }
        });
    }
}
```

ハンドラを追加するために赤字の部分を追加します。

①ハンドラを使用するためにHandlerを定義します。

②定義したハンドラを使用した実行処理をクリックイベント内に用意します。

SQLiteについて

1. SQLiteとは
2. データベースとは
3. データベースの作成
4. データベースの検索
5. 検索結果の取り出し

1. SQLiteとは

PostgreSQLやMySQLなどと同様に、RDBMSの一種ではあるが、DB本体がローカルファイルとして存在します。

アプリケーションに組み込まれる事を目的とした軽量なDBであり、サーバー機能などはありません。

他の端末とのデータ共有のためなどには利用されません。

Androidアプリケーションでは、標準でSQLiteを使用できます。

2. データベースとは

データベースは全てのデータをテーブルという形式で格納します。
テーブルとは同一形式の繰り返しデータの集まりで、カラム(項目)と型をもっています。
各データの最小単位はレコード(行)と呼び、表の中では行として表されます。

3. データベースの作成

データベースの作成に当たって「SQLiteOpenHelper」というクラスを利用します。このクラスは、abstract(抽象)クラスなので、以下のコールバックを実装する必要があります。

- ・`public void onCreate(SQLiteDatabase db)`

データベースを作成したタイミングで呼び出される。通常はここでテーブルを作成する。今回はデータもこのタイミングで追加する。

- ・`public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)`

データベースをバージョンアップしたタイミングで呼び出される。通常はここでデータのコンバートなどを行います。

3. データベースの作成

テーブルにレコードを追加する場合、通常は1レコードずつ追加しなければなりません。これを素直に実装すると、以下のようになります。

```
for (String s : array) {  
    db.execSQL("insert into table values (" + s + ");");  
}
```

1レコードを追加する場合であれば、この方法で十分ですが、複数レコードを追加するのであれば、以下のようにトランザクションとプリコンパイルステートメントを使うことを推奨します。

```
db.beginTransaction();  
try {  
    SQLiteStatement stmt = db.compileStatement("insert into table values (?);");  
    for (String s : array) {  
        stmt.bindString(1, s);  
        stmt.executeUpdate();  
    }  
    db.setTransactionSuccessful();  
} finally {  
    db.endTransaction();  
}
```

このコードは前述のexecSQLを単体で実行するのに比べ、以下のメリットがあります。

- ・途中で失敗した場合ロールバックができる
- ・高速

4. データベースの検索

作成したデータベースを取り扱う際に、まずはデータベースの検索を行います。

```
//.rawQueryでSELECTを実行
String sql = "select foo.a, bar.b from foo, bar where foo.a = bar.a;";
Cursor c = db.rawQuery(sql, null);

// queryでSELECTを実行
String table = "foo, bar";
String[] columns = {"foo.a", "bar.b"};
String selection = "foo.a = bar.a";
Cursor c = db.query(table, columns, selection, null, null, null, null);
```

query:

簡単な検索ならこれが使いやすい。使いにくい複雑な検索も可能

rawQuery:

SQL文を渡して、データもバインドできるので、複雑な検索や一部条件だけ異なる繰り返しの検索などに便利

5. 検索結果の取り出し

検索した結果は、Cursorという形で取得できます。

結果のリストとそのリストを指すカーソルがあって、カーソルを動かしながら結果を取り出すというような処理をします。

```
Cursor c = db.query("capitals", new String[] {"prefecture"}, null, null, null, null, null);
c.moveToFirst();
CharSequence[] list = new CharSequence[c.getCount()];
for (int i = 0; i < list.length; i++) {
    list[i] = c.getString(0);
    c.moveToNext();
}
c.close();
```

まずはqueryメソッドでカーソルを取得して、moveToFirstメソッドでカーソルを先頭に移動しています。次に、getCountメソッドで結果の数を取得し、その数分だけカーソルをmoveToNextメソッドで移動しながら、GetStringメソッドで結果を取り出しています。最後に、closeメソッドでカーソルを閉じておしまいです。