リソース関連

■リソースからリソース参照

```
@string/app_name
```

とか

■ ID を使ったリソース参照

```
android:id="@+id/imageview1"
```

のように XML に書いて、プログラムから

```
R.id.imageview1
```

で参照

エミュレータの電源などの設定を変える

```
telnet localhost 5554
```

等で telnet でエミュレータに接続し、コマンドを流す。
例えば

```
power ac on
```

で電源の接続になる

versionCode と versionName

http://labs.karappo.net/mobiledev/index.php?itemid=209

```
versionCode は、何回目のアップロードかを表す整数
versionName は、自分で好きに決めたバージョン名を表す数字
```

です。なので、アップデートする際は少なくとも前回時よりも versionCode が大きくなっていなければなりません。

スレッドに関する注意点

環境によっては、

- Thread#sleep
- Object#wait

等で、スレッドを止める処理を行うと復帰しないことがあるようだ。

特に 3 秒（3000 ミリ秒）以上停止させると復帰しない現象が多発した。

CountDownTimer でも、countDownInterval を３秒以上にすると、
onFinish が呼ばれないことがあった。

長時間スレッドを止めるときは、１秒毎に復帰、スリープを繰り返した方が良いかもしれない。
CountDownTimer の countDownInterval は 1000 ミリ秒 以上の値は渡さない方が無難かも。

hndroid の Handler とは何か？

http://www.adamrocker.com/blog/261/what-is-the-handler-in-android.html

Android で Web API を使う場合、マルチスレッドによるユーザビリティ向上を以前のエントリで説明しました。
Android アプリの UI はシングル・スレッド モデルです。
単純にマルチスレッドにして UI の操作をしてしまうと、CalledFromWrongThreadException でアプリがダウンしてしまいます。
これを回避する仕組みが Handler です。

Handler の仕組みを簡単に説明しようと思ったのですが、
またもや長くなってしまったので、先にまとめます。

- Android の UI 操作はシングル・スレッド モデル
- ユーザビリティ向上の為にはマルチスレッドが必要
- Handler で実現
- Handler を使わない場合に起きる例外は実行スレッドのチェックで発生
- Handler を使うと、UI Thread の持つキューにジョブを登録できる
- キューは UI Thread により実行される
- 別スレッドから UI Thread に処理を登録するのでスレッドチェックで例外が発生しない

指定時間後に処理をする

```
new CountDownTimer(30000, 1000) {
     public void onTick(long millisUntilFinished) {
       System.out.println("seconds remaining: " + millisUntilFinished / 1000);
     }

     public void onFinish() {
       System.out.println("done!");
     }
   }.start();
```

WIFI 制御

```
WifiManager m = (WifiManager)getSystemService(Context.WIFI_SERVICE);
  try {
    System.out.println(m.setWifiEnabled(true));
  } catch (Exception e){
    e.printStackTrace();
  }
```

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

画面の自動回転制御

```
  try {
    Settings.System.putInt(getContentResolver(), Settings.System.ACCELEROMETER_ROTATION, 0);
```

```
} catch (Exception e){
  e.printStackTrace();
}
```

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

Blluetooth 制御

```
BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
if (mBluetoothAdapter == null) {
  System.out.println("##### not support ######");
}
```

バックライト消灯時間の制御

```
defTimeOut = Settings.System.getInt(getContentResolver(),
Settings.System.SCREEN_OFF_TIMEOUT, DELAY);
Settings.System.putInt(getContentResolver(),
Settings.System.SCREEN_OFF_TIMEOUT, DELAY);
```

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

バックライトの輝度モードの制御

```
try {
  // SCREEN_BRIGHTNESS_MODE
  // SCREEN_BRIGHTNESS_MODE_AUTOMATIC
  // SCREEN_BRIGHTNESS_MODE_MANUAL
  System.out.println(Settings.System.getInt(getContentResolver(),
    "screen_brightness_mode" ));
} catch (Exception e){
  e.printStackTrace();
}
```

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

バックライトの輝度の値の制御

```
try {
  System.out.println(Settings.System.getInt(getContentResolver(),
    Settings.System.SCREEN_BRIGHTNESS ));
} catch (Exception e){
  e.printStackTrace();
}
```

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

エミュレータで SD カード

http://www5d.biglobe.ne.jp/~hra/note/android/004_sd_card_access.htm

■ SD カードのイメージを作る

```
mksdcard 16M test.img
```

■エミュレータに SD カードを指定する

```
emulator -sdcard test.img -avd level4
```

■ SD カードにデータ転送

エミュレータが起動している状態で

```
adb push test.txt /sdcard
```

アラームアプリを作る

http://android.roof-balcony.com/service/alarm/

```
AlarmManager
```

を使うらしい。詳しくは未確認。

アプリを SD カードへ保存可能にする方法

http://android49.blog.fc2.com/blog-entry-24.html

AndroidManifest.xml のタグに次の行を追加するだけです。

```
android:installLocation="auto"
```

これを加えるだけでアプリが SD カードへ保存可能になります。ただし SD カードへ保存する機能は Android 2.2 以降の機能ですので、アプリの API レベルには注意が必要です。

「auto」の部分を

```
internalOnly
```

に設定すると SD カードなど外部ストレージへ保存できなくすることができます。

```
preferExternal
```

に設定するとデフォルトで外部ストレージへ保存され、本体への移動も可能になります。