

## 画面の向きによってレイアウトを変更する

端末の液晶画面は基本的に長方形です。アプリによっては端末の向きを変えることで、縦長のポートレートモード(portrait)と横長のランドスケープモード(landscape)でレイアウトが変わるものや、向きを変えてもレイアウトが固定されているものがあります。

例えば、エミュレータの電話アプリでは下記の画像のように、ポートレートとランドスケープでレイアウトが変わります。



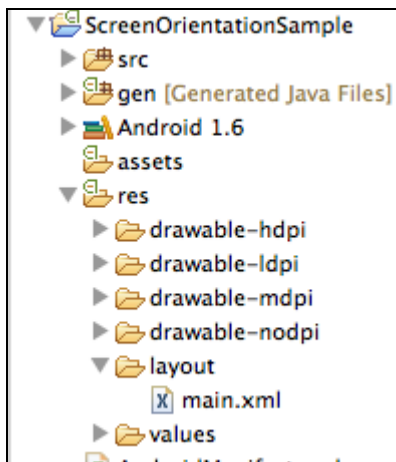
エミュレータで端末（画面）の向きを変える場合は「**CTRL + F11**」を押してください。

それでは、実際に横向き用、縦向き用のレイアウトを用意したい場合にどのようにしたらよいかなどを続きで説明します。

### layout-land フォルダ

res 以下に **layout-land フォルダ**を準備します。

ですがその前に、まず準備しないとどうなるか試してみましょう。以下のようなレイアウトファイルを準備します。



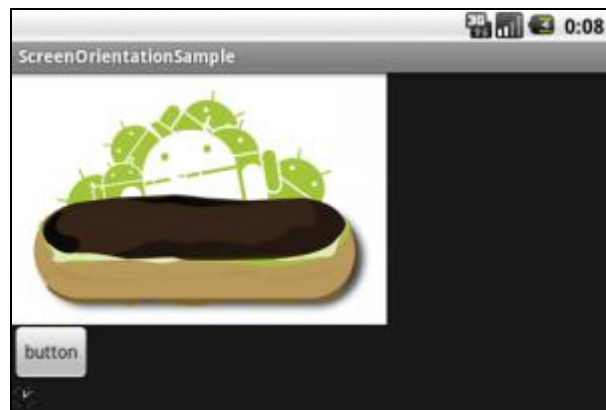
#### main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <ImageView android:id="@+id/ImageView01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/eclair">
    </ImageView>
    <Button android:text="@string/button"
        android:id="@+id/Button01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </Button>
    <AnalogClock android:id="@+id/AnalogClock01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </AnalogClock>
</LinearLayout>
```

---

端末が縦の時と横の時の画像です。

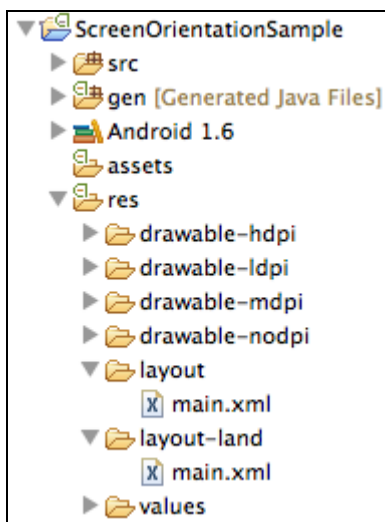
LinearLayout を利用して orientation を vertical に設定しているので横向きの時に縦方向が窮屈ですね。



それではランドスケープ用のレイアウトを用意しましょう。

繰り返しになりますが、まず **res** 以下に **layout-land** フォルダを準備します。

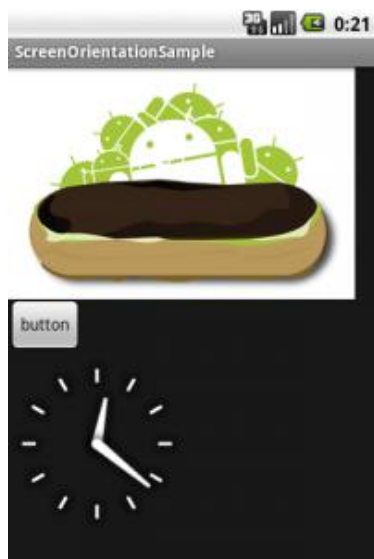
そして同じファイル名でレイアウトファイルを準備します。今回は同じく LinearLayout を利用して orientation を **horizontal** にしてみます。



## main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <ImageView android:id="@+id/ImageView01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/eclair">
    </ImageView>
    <Button android:text="@string/button"
        android:id="@+id/Button01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </Button>
    <AnalogClock android:id="@+id/AnalogClock01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </AnalogClock>
</LinearLayout>
```

横向き (LandScape) の時には layout-land フォルダのレイアウトファイルを読み込みます。



## 画面の向きを固定したいとき

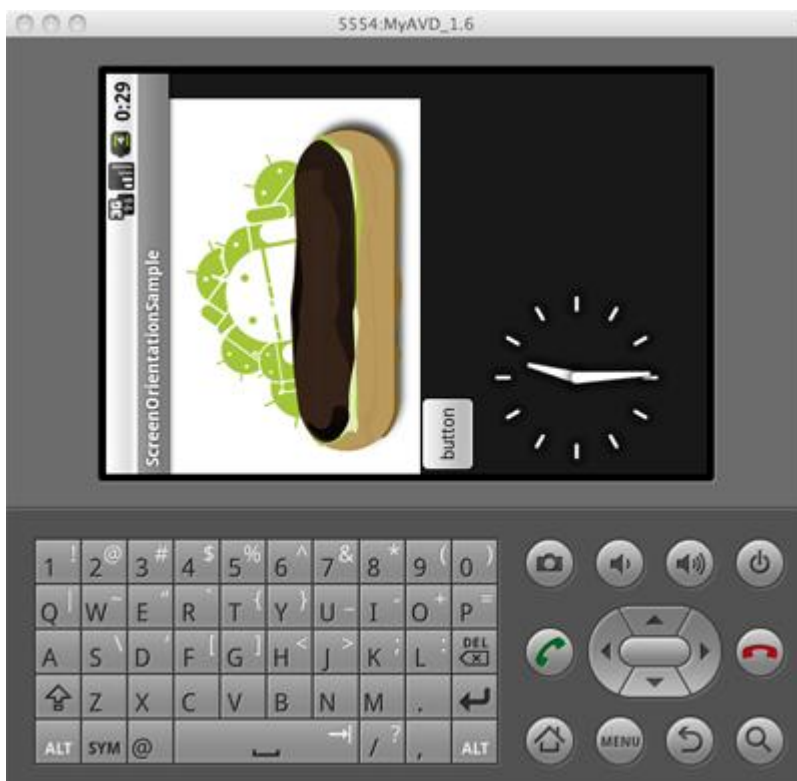
ゲームアプリなどでは画面の向きを固定したい時があると思います。その場合は、AndroidManifest.xml にて Activity に **android:screenOrientation** を設定します。

- portrait : 縦向きに固定
- landscape : 横向きに固定

試しに縦向きに固定してみます。

## AndroidManifest.xml の抜粋

```
<activity android:name=".Activity 系クラス名"  
    android:label="@string/app_name"  
    android:screenOrientation="portrait">
```



このように端末を横にしてもポートレートのままでランドスケープになりません。

## Android アプリ内で使用する文字列の書式を合わせる

Android アプリには、**スタイル**というものを定義する事ができます。

スタイルとは、文字列などに設定する色んな属性を一つの ID で使用できるものです。

---

例えば、Android アプリに複数の画面があるとして、それぞれに配置している TextView の書式を合わせたいと思ったとします。

もちろん、それぞれの TextView の属性を設定してやれば実現する事ができますが、もし、仕様変更などの理由で、文字列の色を変えたいとか、文字列の太さを変えたいという場合に、その都度、全ての TextView の属性を書き換えるのは大変です。

そういう仕様変更にも即座に対応できるように、上記のようなケースでは、「スタイル」を使用すべきです。

それではスタイルの使用方法を説明します。

### ●Android のスタイルの使用方法

まず、Android アプリのプロジェクトの **res/values** フォルダ内に、xml ファイルを新規に用意します。

このファイルは、スタイルを定義するための xml ですので、ファイル名は **style.xml** とします。

ファイル名は任意ですが、一般的には **style.xml** とするそうです。

---

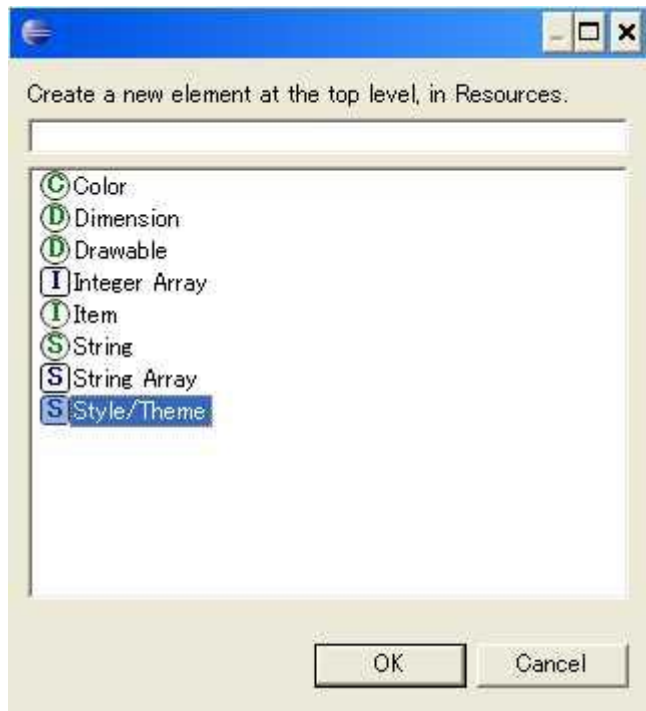
次に、作成した **style.xml** を Android Resource Editor で開きます。

通常は関連付けられているので、ダブルクリックで OK です。

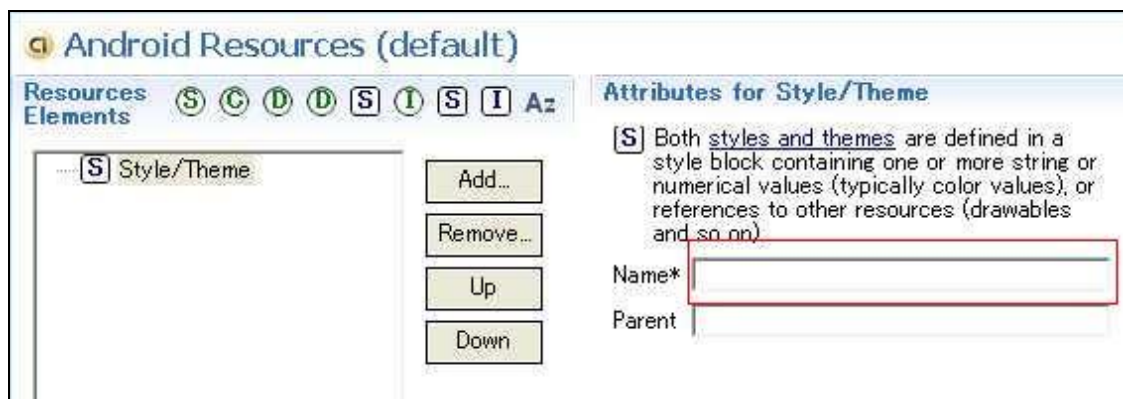
以下の画面が表示されます。



赤枠部分の Add ボタンを押し、「Style/Theme」を選択します。

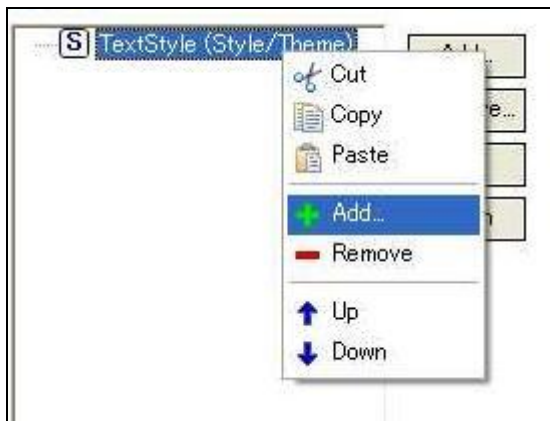


すると、以下の画面のように Style/Theme の要素が追加されます。



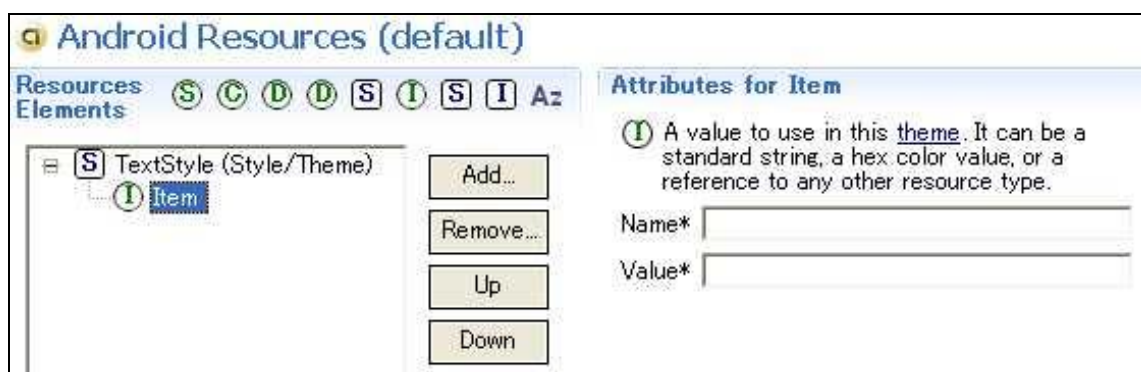
上の画像赤枠部分の Name 属性へ、わかりやすく適当な名前をつけます。  
例えば、今回は TextStyle とでもしておきます。

そして、以下の画面のように、追加した Style/Theme を右クリック⇒Add と選択します。



選択肢には「Item」しかありませんので、それをダブルクリックするなどで、Style/Theme 属性の子要素として、Item 属性を追加します。

以下の画面のようになりましたね。



Item 属性には、統一させたいスタイルを設定します。

Name には統一したい View の属性名を、Value には何で統一するかを設定します。

今回は、TextView の文字色と文字サイズを統一させたい！という事とします。

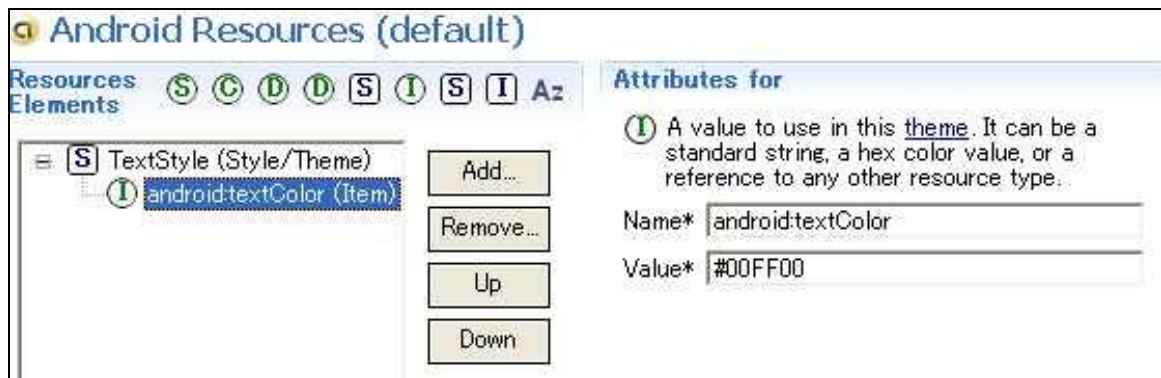
TextView の文字色は、「**android:textColor**」

TextView の文字サイズは、「**android:textSize**」

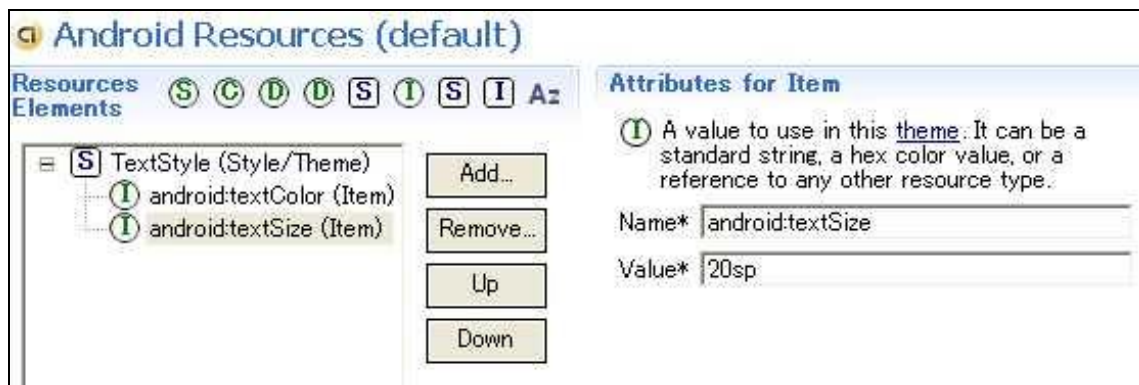
これが Name に使う文字列になります。

例えば文字色を緑で統一したい！という場合は以下のようにします。





文字サイズを 20sp に統一したい！という場合は以下のようにします。



これでスタイルの定義は OK です。

次に、レイアウト用の XML ファイルの<TextView>タグ内に style 属性を追加します。

#### ●main.xml

<TextView

```
style="@style/TextStyle"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/hello" />
```

## Android アプリ全体で、文字色や背景色を統一させる方法 (Theme)

Android には、**テーマ**というものを定義する事ができます。

スタイルをご存知でしょうか。

テーマとは、このスタイルを Android アプリ全体の初期値として使用する方法です。

### Android アプリのテーマの使用方法

テーマの定義方法は、基本的にスタイルと同じです。

「[Android アプリで使う文字列のスタイルを統一する方法](#)」を参照してください。

この方法で XML にスタイルを定義するまでは、全く同じです。

スタイルは、各 View の Style 属性で、定義したスタイルを参照させていましたが、Theme (テーマ) として参照させる場合は、以下のようにします。

---

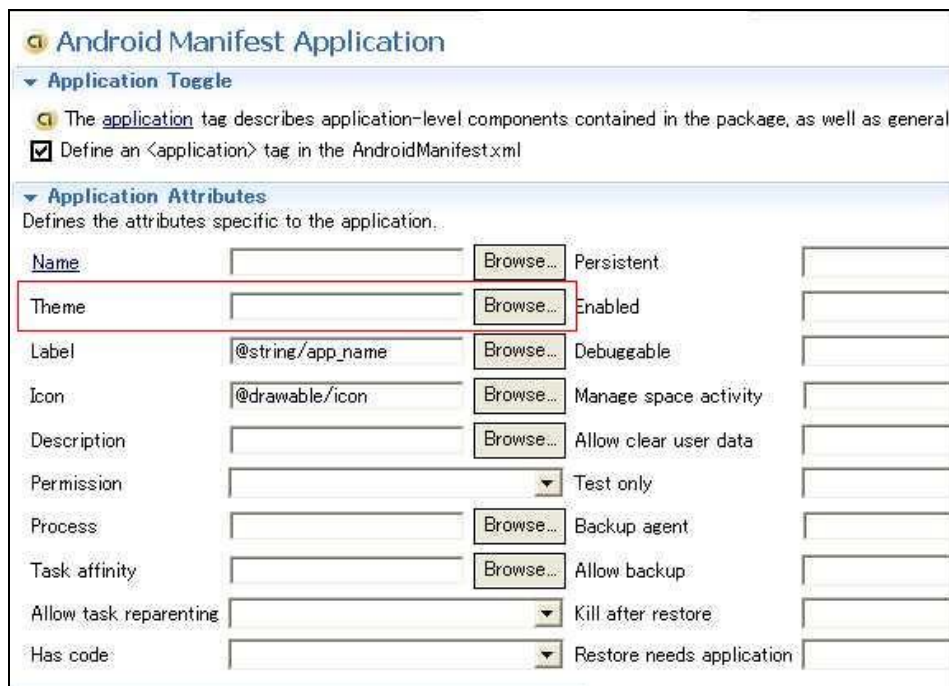
Eclipse を使った方法を説明します。

まず、テーマを指定したいアプリの、AndroidManifest.xml を Android Manifest Editor で開きます。

通常は関連付けられているので、ダブルクリックで OK です。

そして、Application タブを開きます。

以下のような画面になりますね。



赤枠で囲っているところを見てください。

「Theme」とあります。

この Browse ボタンを押すと、以下の画面が表示されます。



ここには、定義しているスタイルが一覧表示されますので、テーマとして使いたいスタイルを選択すれば OK です。

AndroidManifest.xml へ以下の太字部分が追記されます。

---

```
<application android:icon="@drawable/icon" android:label="@string/app_name"
android:theme="@style/TextStyle" >
```

---

Eclipse を使わずに、この太字部分を手書きしても OK です。

なお、Android アプリで使う文字列のスタイルを統一する方法では、文字色、文字サイズ、文字スタイルについて説明していましたが、それ以外にテーマとして使うスタイルによく使うプロパティをいくつか紹介します。

**android:windowBackground** : 背景画像を指定します。

**android:windowTitleStyle** : アプリケーションのタイトル文字のスタイルを指定します。

**android:textColorPrimary** : 文字色 (プライマリー)

**android:textColorSecondary** : 文字色 (セカンダリー)

上記が、基本的によく使うプロパティだと思います。