

## 1. レイアウトの基本

Android でレイアウトを作成する方法には、「XML で定義する方法」と「プログラム上から作成する方法」の 2 つがある。

XML で定義する方が読みやすく変更も容易なので、基本的には XML でレイアウトを作成し、動的にレイアウトを変更したい場合にプログラムで記述するのが一般的である。

## 2. レイアウトの種類

ビューを配置するベースとなるレイアウトについて説明する。ここで紹介するレイアウト以外にも RelativeLayout (ビュー同士の相対位置で配置)、AbsoluteLayout (絶対位置で配置) があるのだが、

ここで説明する 3 つの組み合わせでほとんどのレイアウトが実現できるので、この 3 つを覚えておけば十分である。(なお、レイアウト画像は Codezine から転載させてもらった)

また、レイアウトは入れ子に出来る性質があり、複数のレイアウトを組み合わせる事が可能である。

### 2-1. LinearLayout

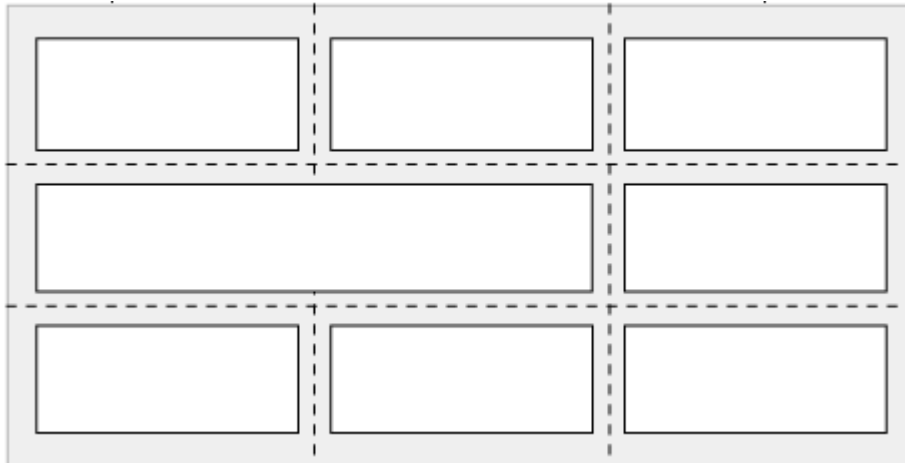
LinearLayout は任意のビューを縦／横方向に直線上に配置するレイアウトである。直線上に並ぶ性質であるため、ビューとビューが重なる事はない

縦方向(左)、横方向(右)



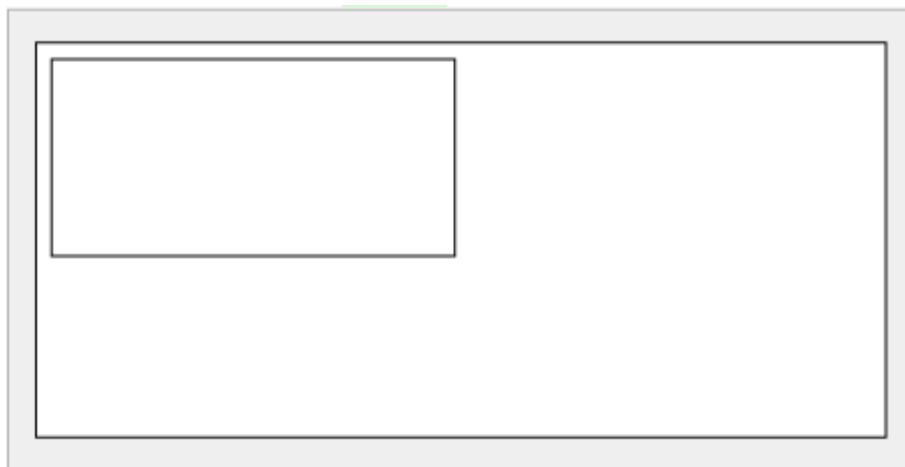
## 2-2. TableLayout

TableLayout は任意のビューを表形式で配置したい場合に利用する(HTML の Table に近いイメージ)。



## 2-3. FrameLayout

ビューを重ねて表示するためのレイアウト。後から配置した部品が前面に来る仕様となっている。



## 3. まとめ

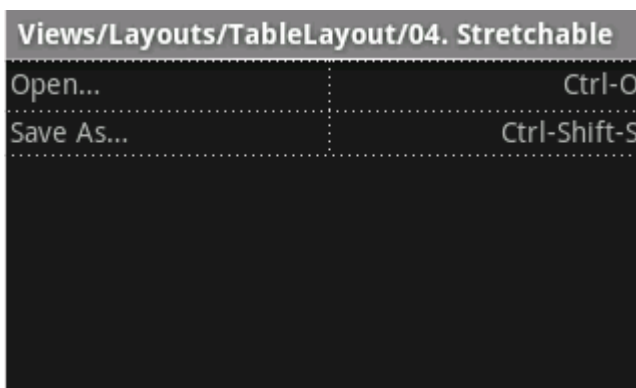
- レイアウトの作成は GUI エディタではなく、XML ファイルを直接編集する
- レイアウトは入れ子に出来るので、複数のレイアウトを組み合わせる事が出来る
- レイアウトは 3 つだけ覚えておけばよい
  - LinearLayout 直線上に並べる
  - TableLayout テーブル上に並べる
  - FrameLayout ビューとビューを重ねる

## ●テーブルレイアウトのサンプル

[TableLayout](#) は、子を行と列に配置します。TableLayout コンテナは、行、列、セルの境界線は表示しません。このテーブルの列の数は、セルの数が最も多い行と同じになります。テーブルのセルを空にすることはできますが、HTML では可能なセルの結合はできません。

[TableRow](#) オブジェクトは、TableLayout の子のビューです( TableRow はそれぞれそのテーブルの 1 行を定義しています )。それぞれの行は、0 個またはそれより多いセルを持ち、さまざまな種類のビューが定義されます。ということは、行のセルは、ImageView や TextView オブジェクトのような、さまざまな View オブジェクトで構成されるかもしれません。また、それが ViewGroup オブジェクトかもしれません( 例えば、もうひとつの TableLayout をセルとしてネストさせることも可能です )。

以下は、行と列がそれぞれふたつあるレイアウトのサンプルです。隣のスクリーンショットがその結果で、セルの境界線は点線で表示しています ( 見れるように追加しました ) 。



```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="1">
    <TableRow>
        <TextView
            android:text="@string/table_layout_4_open"
            android:padding="3dip" />
        <TextView
            android:text="@string/table_layout_4_open_shortcut"
```

```

        android:gravity="right"
        android:padding="3dip" />
    </TableRow>

    <TableRow>
        <TextView
            android:text="@string/table_layout_4_save"
            android:padding="3dip" />
        <TextView
            android:text="@string/table_layout_4_save_shortcut"
            android:gravity="right"
            android:padding="3dip" />
    </TableRow>

</TableLayout>

```

列は隠したり、伸ばして可能な限りスクリーンを埋める設定にしておいたり、テーブルがスクリーンに収まるように、列を強制的に縮む設定にしておくこともできます。詳しくは、[TableLayout リファレンス](#) のドキュメントを参照してください。

## ● 枠線を水平に入れるサンプル

```

<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"

    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="1">

    <TableRow>
        <TextView
            android:layout_column="1"

```

```

        android:text="Open..."
        android:padding="3dip" />
    <TextView
        android:text="Ctrl-O"
        android:gravity="right"
        android:padding="3dip" />
</TableRow>

<TableRow>
    <TextView
        android:layout_column="1"
        android:text="Save..."
        android:padding="3dip" />
    <TextView
        android:text="Ctrl-S"
        android:gravity="right"
        android:padding="3dip" />
</TableRow>

<TableRow>
    <TextView
        android:layout_column="1"
        android:text="Save As..."
        android:padding="3dip" />
    <TextView
        android:text="Ctrl-Shift-S"
        android:gravity="right"
        android:padding="3dip" />
</TableRow>

<View
    android:layout_height="2dip"
    android:background="#FF909090" />

<TableRow>
    <TextView

```

```

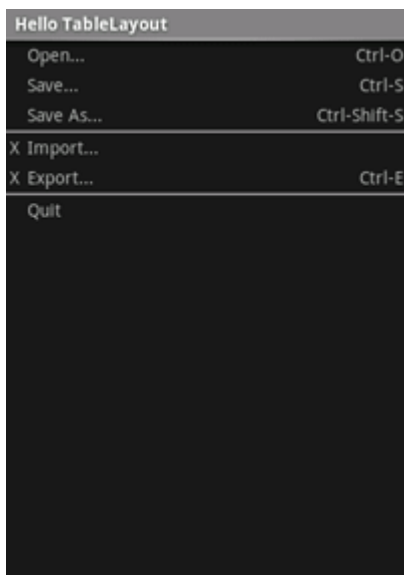
        android:text="X"
        android:padding="3dip" />
    <TextView
        android:text="Import..."
        android:padding="3dip" />
</TableRow>

<TableRow>
    <TextView
        android:text="X"
        android:padding="3dip" />
    <TextView
        android:text="Export..."
        android:padding="3dip" />
    <TextView
        android:text="Ctrl-E"
        android:gravity="right"
        android:padding="3dip" />
</TableRow>

<View
    android:layout_height="2dip"
    android:background="#FF909090" />

<TableRow>
    <TextView
        android:layout_column="1"
        android:text="Quit"
        android:padding="3dip" />
</TableRow>
</TableLayout>

```



## ●レイアウトの入れ子設定

### 概要

LinearLayout を入れ子に設定することにより、複雑な画面を設計することができるようになります。このサンプルは、XML で実現していて、Java ファイルは Eclipse で Android プロジェクトを生成したときに自動生成されるコードから変更しません。main.xml を次のように変更します。

### サンプルプログラム

```
<?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">
```

```
    <LinearLayout
```

```
        android:orientation="horizontal"
```

```
        android:layout_width="fill_parent"
```

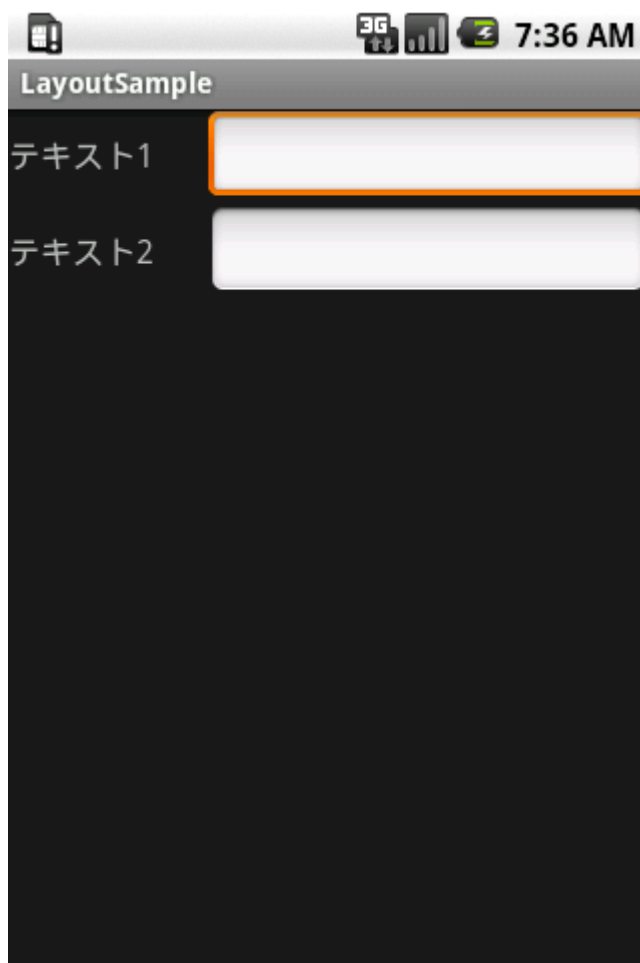
```
        android:layout_height="wrap_content">
        <TextView
            android:id="@+id/text01"
            android:text="テキスト 1"
            android:textSize="16dip"
            android:layout_width="100dip"
            android:layout_height="wrap_content" />
        <EditText
            android:id="@+id/edit01"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" />
    </LinearLayout>
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <TextView
            android:id="@+id/text02"
            android:text="テキスト 2"
            android:textSize="16dip"
            android:layout_width="100dip"
            android:layout_height="wrap_content" />
        <EditText
            android:id="@+id/edit02"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" />
    </LinearLayout>
```



</LinearLayout>

## 実行結果

サンプルプログラムを動作させると、エディットボックスが縦に 2 つ並んだ画面が出力します。XML ファイルを変更することで、ボタンやチェックボックスを横に並べることもできるようになります。



- [親ページ](#)
- [スタートページ](#)
- [ページ一覧](#)
- [ページ履歴](#)
- [最終更新](#)

## 概要

チェックボックスとラジオボタンをレイアウトファイルで指定するサンプル

処理内容は、[ユーザインターフェース/チェックボックスとラジオボタン](#)と同じである

## 実行結果

## 手順

### 1. プロジェクトの作成

- Eclipse を起動
- ファイル>プロジェクト>Android>Android プロジェクトを選択。以下を設定したら完了ボタンを押してプロジェクトを作成する

プロジェクト名	CheckRadioLayoutEx
ビルド・ターゲット	Android 1.6
アプリケーション名	CheckRadioLayoutEx
パッケージ名	net.easyjp.checkradiolayoutex 通常は自分のドメイン名を逆にしたものを使用する (例えば、easyjp.net → net.easyjp) ここでは適当に決めて構わない
CreateActivity	CheckRadioLayoutEx ここで指定した名前がプログラム本体のソースファイル名になる
Min SDK Version	4 (ここには、選択したビルド・ターゲットの API の数値を入れる。[[BR]]例: Android 1.6 = 4, Android 2.1 = 7, Android 2.2 = 8

### 2. レイアウト XML(res/main.xml)の編集

レイアウトファイルを以下のように修正する

- res/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"

    •    android:background="#FFFFFF"

    •    >

    •

    •    <!-- チェックボックス -->

    •    <CheckBox

    •        android:id="@+id/checkBox"

    •        android:layout_width="fill_parent"

    •        android:layout_height="wrap_content"

    •        android:text="@string/checkbox"

    •        android:checked="true"

    •        android:textColor="#FF000000"

    •    />

    •

    •    <!-- ラジオグループ -->

    •    <RadioGroup

    •        android:id="@+id/radioGroup"

```

- android:layout\_width="fill\_parent"
- android:layout\_height="wrap\_content"
- >
- <RadioButton
- android:id="@+id/radio0"
- android:layout\_width="fill\_parent"
- android:layout\_height="wrap\_content"
- android:text="@string/radio0"
- android:textColor="#FF000000"
- />
- 
- <RadioButton
- android:id="@+id/radio1"
- android:layout\_width="fill\_parent"
- android:layout\_height="wrap\_content"
- android:text="@string/radio1"
- android:textColor="#FF000000"
- android:checked="true"
- />
- 
- </RadioGroup>
-

- 
- <!-- ボタン -->
- <Button
- android:id="@+id/button"
- android:textSize="16sp"
- android:text="@string/button"
- android:layout\_width="wrap\_content"
- android:layout\_height="wrap\_content"
- />
- 
- </LinearLayout>

#### 4.res/string.xmlを修正する

- res/string.xml

- <?xml version="1.0" encoding="utf-8"?>
- <resources>
- <string name="hello">Hello World, CheckRadioLayoutEx!</string>
- <string name="app\_name">CheckRadioLayoutEx</string>
- <string name="checkbox">チェックボックス</string>
- <string name="radio0">ラジオボタン 0</string>
- <string name="radio1">ラジオボタン 1</string>
- <string name="button">結果表示</string>

- </resources>

## 5. CheckRadioLayoutEx.java の修正

- CheckRadioLayoutEx.java

```
•package net.easyjp.checkradiolayoutex;

•

•import android.app.Activity;

•import android.app.AlertDialog;

•import android.content.DialogInterface;

•import android.os.Bundle;

•import android.view.Window;

•import android.view.View;

•import android.widget.Button;

•import android.widget.CheckBox;

•import android.widget.RadioGroup;

•import android.widget.RadioButton;

•

•public class CheckRadioLayoutEx extends Activity

•    implements View.OnClickListener {

•    private Button button;

•    private CheckBox checkBox;

•    private RadioGroup radioGroup;
```

- 
- // 初期化
- @Override
- public void onCreate(Bundle savedInstanceState) {
- super.onCreate(savedInstanceState);
- requestWindowFeature(Window.FEATURE\_NO\_TITLE);
- setContentView(R.layout.main);
- 
- button = (Button)findViewById(R.id.button);
- button.setOnClickListener(this);
- 
- checkBox = (CheckBox)findViewById(R.id.checkBox);
- 
- radioGroup =
- (RadioGroup)findViewById(R.id.radioGroup);
- }
- 
- // ボタンのクリックイベント
- @Override
- public void onClick(View view) {
- // チェックボックスとラジオボタンの状態取得
-

```

•      // 選択されているラジオボタンを取得
•
•      RadioButton radio =
(RadioButton)findViewById(radioGroup.getCheckedRadioButto
nId());
•
•
•      showDialog(this, "状態", "チェックボックス: " +
checkBox.isChecked() + "¥n" +
•
•      "ラジオボタン: " + radio.getText().toString());
•
•      }
•
•
•      // ダイアログの表示
•
•      private void showDialog(final Activity activity, String
title, String message) {
•
•          AlertDialog.Builder ad = new
AlertDialog.Builder(activity);
•
•          ad.setTitle(title);
•
•          ad.setMessage(message);
•
•          ad.setPositiveButton("OK", new
DialogInterface.OnClickListener() {
•
•              public void onClick(DialogInterface dialog, int
which) {
•
•                  activity.setResult(Activity.RESULT_OK);
•
•              }
•
•          });
•
•          ad.create();

```



```
•      ad.show();  
•    }  
• }
```