

ANDROID開発基礎から 設定画面まで

日本Androidの会

赤井 忠昭

アジェンダ

- ▶ Hello Androidについて
 - ▶ これから始める人を対象とします。
 - ▶ プロジェクトの中身について説明します。
- ▶ Activityについて
 - ▶ Activityを簡単に説明します。
- ▶ Intentについて
 - ▶ Intentを簡単に説明します。
- ▶ 設定画面を作ろう
 - ▶ 一応、今回のメインテーマです。

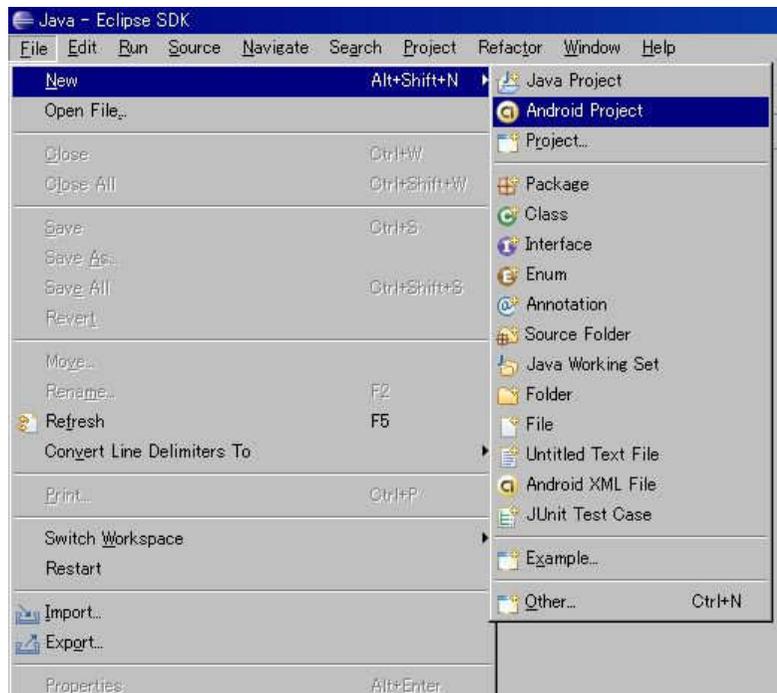
Androidアプリ開発に必要なもの

- ▶ J2 SE 5.0
- ▶ Eclipse 3.4以降
- ▶ Android SDK(最新は2.0)
- ▶ ADT(Android Development Tool)

開発環境の作り方はいろいろなサイトで記載されているので、ここでは説明しません。

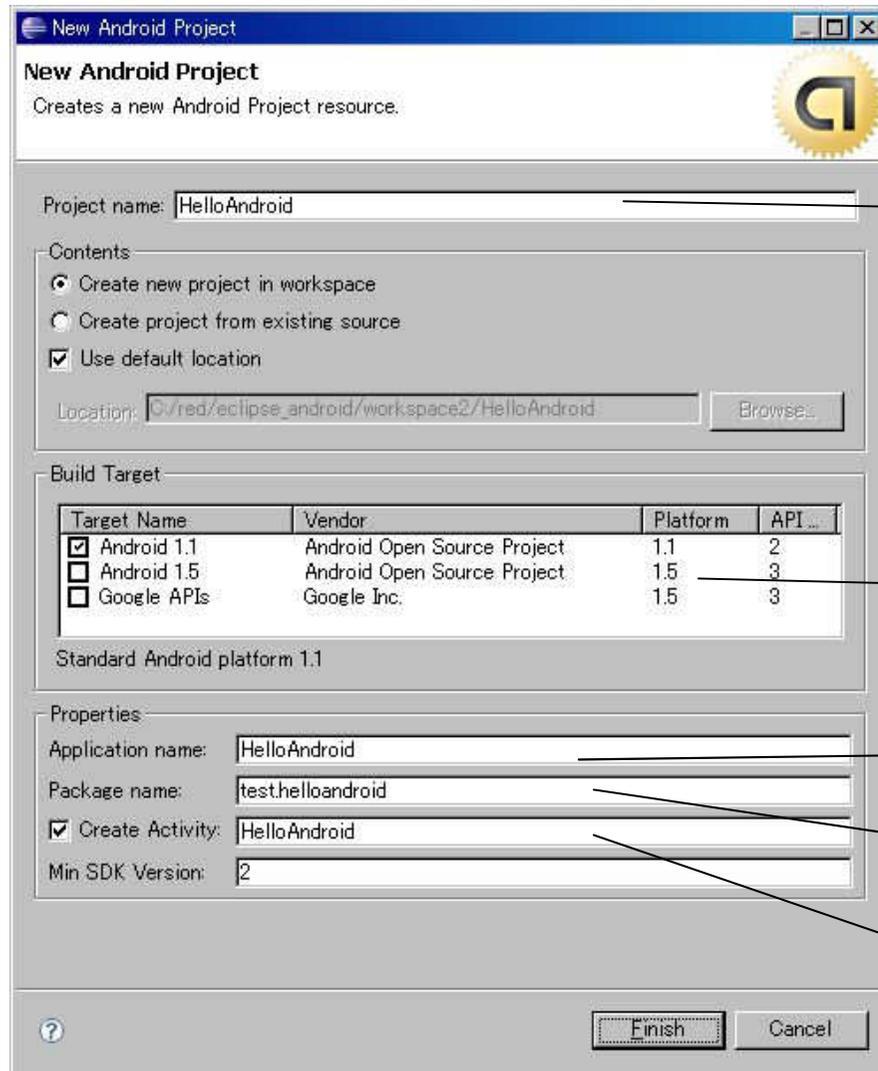
Hello Androidの表示

▶ プロジェクトを作成しよう



File⇒New⇒Android Project
を選択

Hello Androidの表示 続き



※Android SDK 1.5の場合

→ プロジェクト名を指定

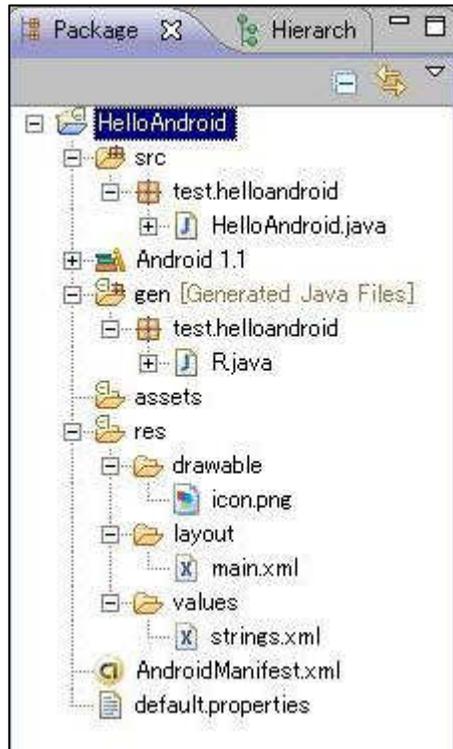
→ Androidの対象バージョンを指定

→ アプリケーション名を指定

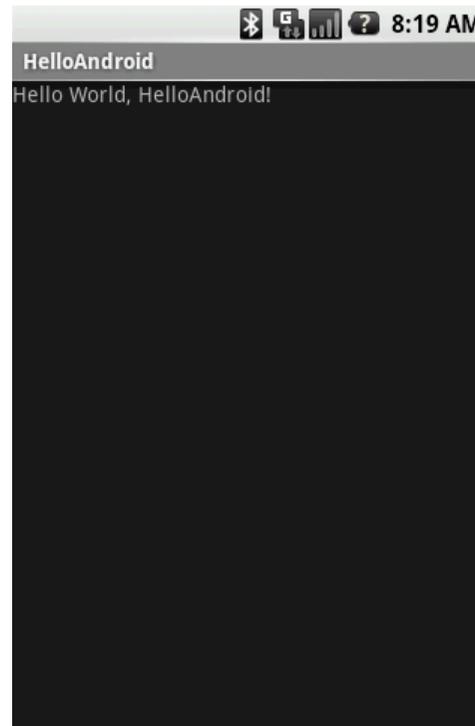
→ パッケージ名を指定
(最低でも1つは区切る必要あり)

→ Activityを作る場合は
Activity名を指定

Hello Androidの表示 続き

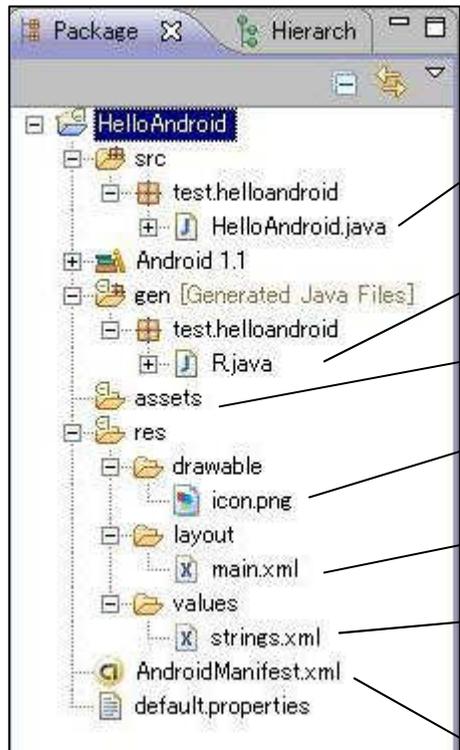


- ▶ プロジェクトを作成すると左のような構成ができあがります。
- ▶ これだけで Hello Androidは表示されます。



実行結果

パッケージエクスプローラー簡単解説



Javaファイル(ここにプログラムを書いていきます。)

XMLで記述した内容がこのファイルに自動的に反映されます。
このファイルは触ってはいけません。

アプリケーション内で使用するデータを格納します。

表示する画像関係を格納します。

デザインを定義するXMLを格納します。

アプリケーションに表示する文字等を格納します。

アプリケーションのアクティビティやサービスの挙動・どの
ようなデータを処理するか・どのようにして起動するかな
どの様々な情報を登録します。

とても重要なファイルです。

Androidアプリ作成の基本

Androidアプリは基本的には以下のようにして作成します。

- ▶ res/layout/hoge.xmlでレイアウトをデザイン
- ▶ res/values/strings.xmlで文字の表示
(簡単に多言語対応できるようにするため)
- ▶ Javaファイルでアプリケーションの振る舞いを記述
- ▶ AndroidManifest.xmlでアプリケーションに関する様々な設定を記述

開発を始めた当初はAndroidManifest.xmlの記述漏れがよくあります。注意しましょう。

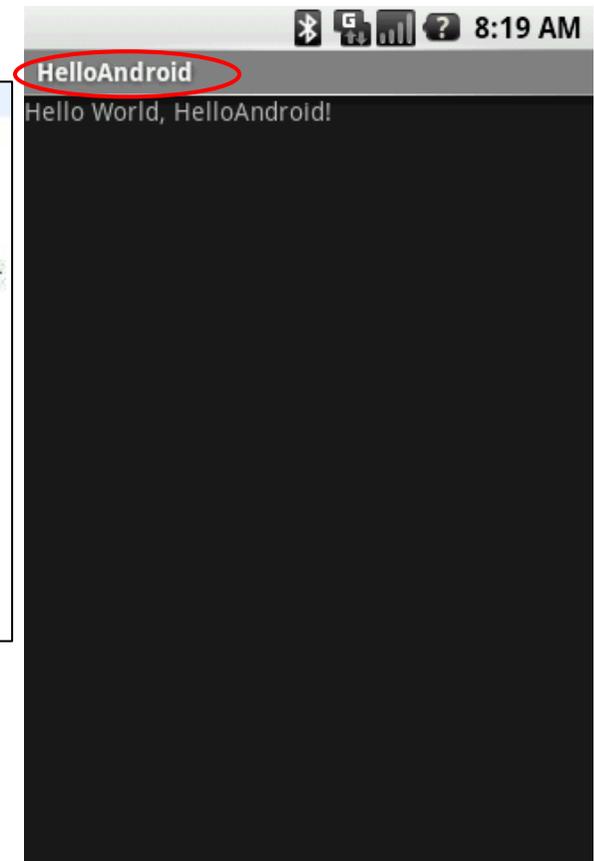
Hello Androidを確認しよう

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="test.helloandroid"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".HelloAndroid"
            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>
    <uses-sdk android:minSdkVersion="2" />
</manifest>
```

res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, HelloAndroid!</string>
    <string name="app_name">HelloAndroid</string>
</resources>
```



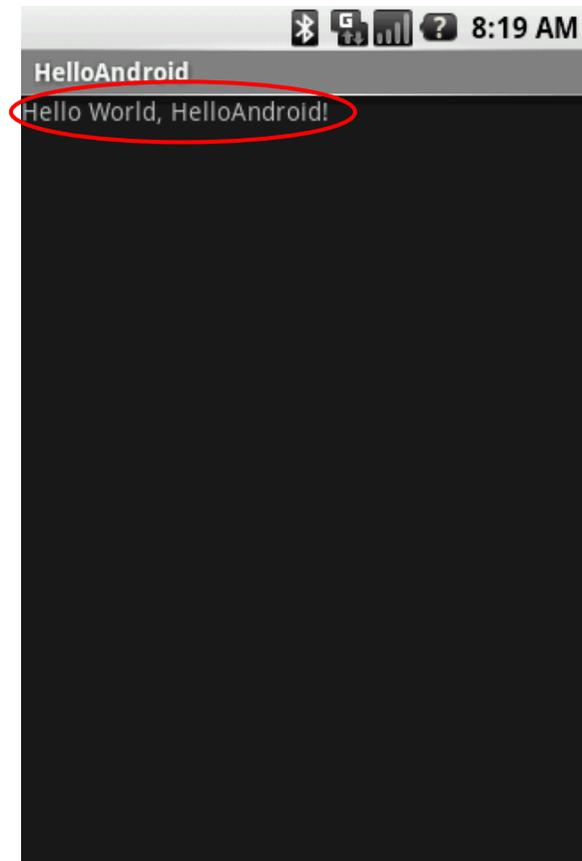
Hello Androidを確認しよう 続き

res/layout/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"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```

res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, HelloAndroid!</string>
    <string name="app_name">HelloAndroid</string>
</resources>
```



時間の関係上、レイアウトの書き方等は今回は省略します。

Hello Androidを確認しよう 続き

res/layout/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"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```

R.java

```
/* AUTO-GENERATED FILE. DO NOT MODIFY.

package test.helloandroid;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

HelloAndroid.java

```
package test.helloandroid;

import android.app.Activity;
import android.os.Bundle;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

setContentViewで使用する
レイアウトを決定

Activityって何？

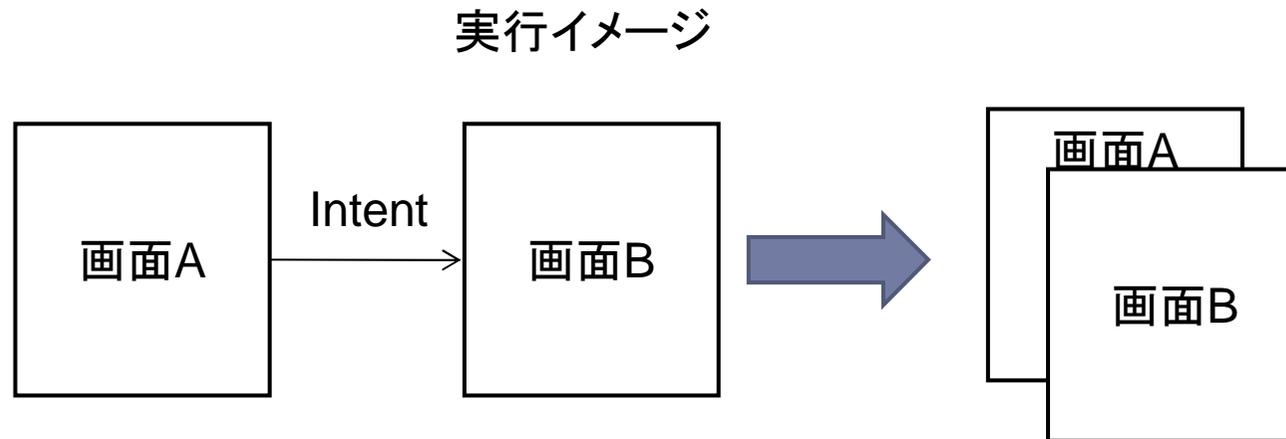
- ▶ アプリの画面に相当するもの
- ▶ Activityにはライフサイクルがある
(ライフサイクルは重要ですが、時間の関係上省略します)
- ▶ AndroidManifest.xmlで定義する必要がある。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="test.helloandroid"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".HelloAndroid"
            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>
    <uses-sdk android:minSdkVersion="2" />
</manifest>
```

Androidによる画面の遷移

- ▶ 複数の画面を持つアプリケーションで画面遷移を行うには新しいActivityを実行する
- ▶ 新しいActivityを実行するためにIntentを使用する
- ▶ 実行されたActivityは呼び出し元Activityの上に重なるようにして表示される



Intentって何？

- ▶ 和訳すると「意図」とか「意思」という意味
- ▶ Activityから異なるActivityを呼び出すときのトリガー
- ▶ Intentはデータのやりとりをすることもできる
- ▶ Intentをやり取りするActivityが異なるpackageの場合はアプリからアプリを起動する
- ▶ 同一packageの場合は画面遷移
- ▶ Intentには明示的Intentと暗黙的Intentがある

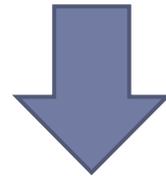
時間の関係上、Intentの細かい説明は省略します。

設定画面を作ってみる(仕様)

- ▶ 大半のアプリケーションは設定画面が存在するため、今までの説明をもとに設定画面を作ってみる。
- ▶ 画面にボタンを配置して、そのボタンを押下すると設定画面のActivityを呼び出す。
- ▶ 設定画面で値を変更すると呼び元画面では変更された値が反映されることを確認する。

設定画面を作ってみる(実装方法?)

- ▶ 今までの話からいくと起動元Activityと設定画面Activityを作成する。
- ▶ Intentを使って起動元Activityから設定画面Activityを起動する。
- ▶ 設定画面Activityと起動元ActivityはIntentでデータのやり取りを行う。



できなくはないと思うが大変そう。
最適なデータの共有方法を考えてみる。

データ共有方法

- ▶ Androidはいくつかのデータ共有方法がある。
 - ▶ Intent
 - ▶ 関数呼び出しの引数のような使い方
 - ▶ データ共有というよりはデータの受け渡し
 - ▶ SharedPreferences
 - ▶ Activity間やアプリケーション間でデータを共有
 - ▶ 実はXMLファイル
 - ▶ Content Provider
 - ▶ 他アプリケーションにデータを提供できる
 - ▶ SQLを使用する
 - ▶ Network
 - ▶ Networkを經由してサーバ上にデータを保持し共有する
 - ▶ 処理に時間がかかる

SharedPreferencesで簡単設定画面

- ▶ 起動元Activityと設定画面Activityを作成する。
- ▶ 設定画面ActivityはPreferenceActivityを継承する。
- ▶ Intentを使って起動元Activityから設定画面Activityを起動する。
- ▶ 設定画面Activityで設定された情報はSharedPreferencesで保存され、起動元Activityはその保存された値を参照する

起動元Activityの設定

- ▶ res/layout/main.xmlにボタンを追加
(ついでにTextViewも変更)

```
<?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:id="@+id/text_view"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/settings_value"
    />
<Button
    android:id="@+id/settings_button"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/settings"
    />
</LinearLayout>
```

Javaファイルから操作できるようにするためにidを追加

わかりやすいようにStringの名称を変更

Button追加

起動元Activityの設定 続き

▶ HelloAndroid.javaを修正

▶ 必要クラスをimport

```
import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
```

▶ ボタン押下時にIntentを発行してPreferenceActivityを実行

```
public class HelloAndroid extends Activity implements OnClickListener {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button settingsButton = (Button)findViewById(R.id.settings_button);
        settingsButton.setOnClickListener(this);
    }

    public void onClick(View v) {
        startActivity(new Intent(this, SettingsActivity.class));
    }
}
```

起動元Activityの設定 続き

▶ HelloAndroid.javaを修正 続き

▶ onResume時にSharedPreferencesの値を取得して表示

```
@Override
public void onResume() {
    super.onResume();
    SharedPreferences mPrefs = PreferenceManager.getDefaultSharedPreferences(this);

    TextView tv = (TextView) this.findViewById(R.id.text_view);
    String value = mPrefs.getString("settings_value", "");
    if( !value.equals("") ) {
        tv.setText(value);
    }
}
```

設定画面Activityの設定 (SettingsActivity)

- ▶ res/layout/settings.xmlにListPreferenceを登録

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:title="@string/settings">

  <ListPreference
    android:key="settings_value"
    android:title="@string/settings_title"
    android:summary="@string/settings_summary"
    android:dialogTitle="@string/settings_list_dialogtitle"
    android:entries="@array/settings_list_entries"
    android:entryValues="@array/settings_list_entryvalues" />
</PreferenceScreen>
```

ルートのタグはPreferenceScreenとなる

設定画面Activityの設定 (SettingsActivity) 続き

- ▶ SettingsActivity.javaを作成してView用xmlを紐付ける

```
package test.helloandroid;

import android.os.Bundle;
import android.preference.PreferenceActivity;

public class SettingsActivity extends PreferenceActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.layout.settings);
    }
}
```

→ PreferenceActivityを継承

→ XMLよりPreferenceを設定

設定画面Activityの設定 続き

- ▶ AndroidManifest.xmlに新しいActivityを定義

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="test.helloandroid"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".HelloAndroid"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="SettingsActivity" android:label="@string/settings">
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="2" />
</manifest>
```

文字列の設定

- ▶ res/values/strings.xmlに表示文字をセット

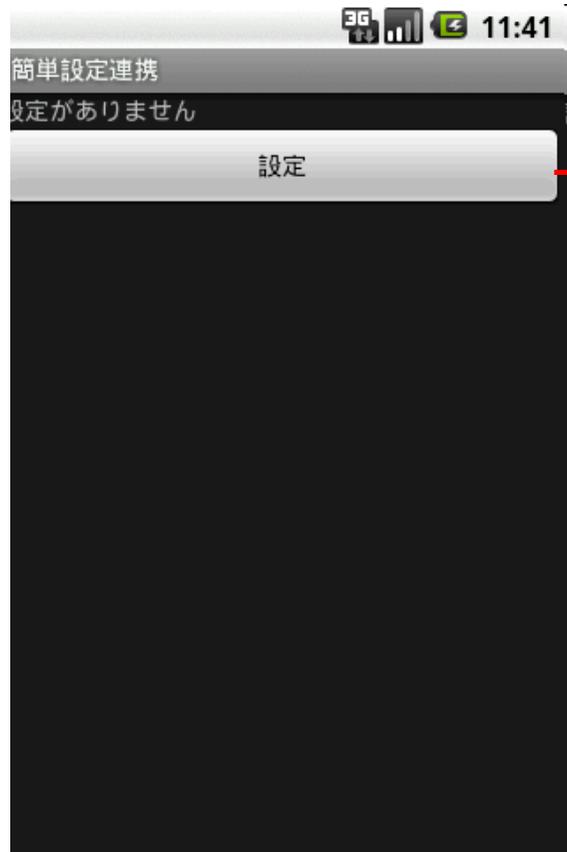
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="settings_value">設定がありません</string>
  <string name="app_name">簡単設定連携</string>
  <string name="settings">設定</string>
  <string name="settings_title">設定項目タイトル</string>
  <string name="settings_summary">設定項目サマリー</string>
  <string name="settings_list_dialogtitle">リストダイアログタイトル</string>
</resources>
```

文字列の設定 続き

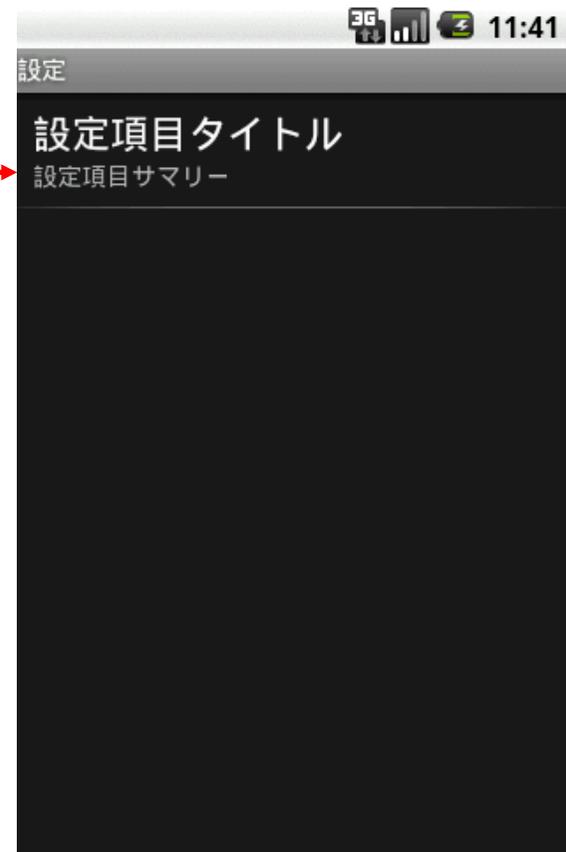
- ▶ res/values/arrays.xmlに選択する値リストを設定

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string-array name="settings_list_entries">
    <item>リスト1</item>
    <item>リスト2</item>
    <item>リスト3</item>
    <item>リスト4</item>
  </string-array>
  <string-array name="settings_list_entryvalues">
    <item>list1</item>
    <item>list2</item>
    <item>list3</item>
    <item>list4</item>
  </string-array>
</resources>
```

実行結果



ボタン押下

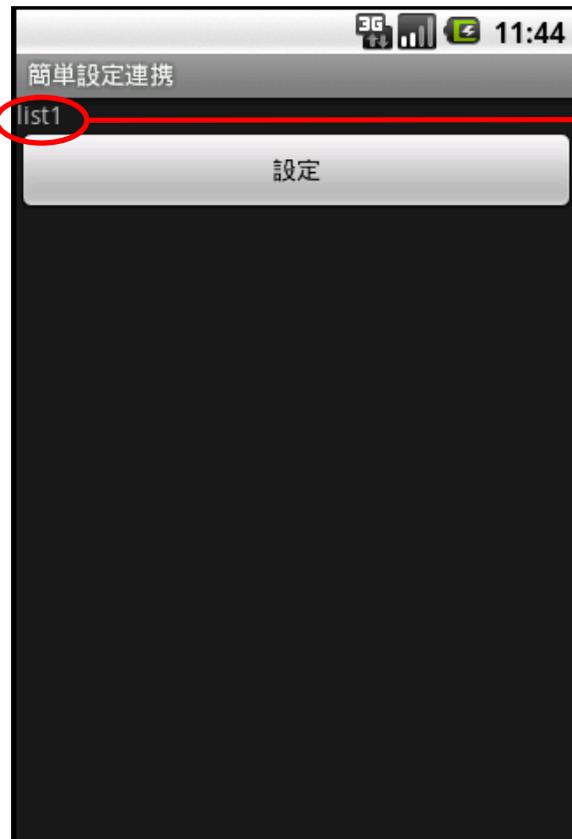


選択

実行結果 続き



実行結果 続き



リスト1の値である
list1が表示される

SharedPreferencesは値をXMLで保持するため、次回以降に起動してもlist1の値は保持されたままです。

PreferenceScreenで設定できるもの

- ▶ 今回利用したListPreference以外にもいろいろと設定できる種類があります。
 - ▶ CheckBoxPreference
 - ▶ チェックボックスを表示
 - ▶ EditTextPreference
 - ▶ 入力ダイアログを表示
 - ▶ PreferenceCategory
 - ▶ グループिंगして見やすくするためのもの(文字列を設定)
 - ▶ PreferenceScreen
 - ▶ 別の設定画面を呼び出す

最後に

- ▶ 今回、説明した方法はあくまでも一例であり、一番いいということではありません。
- ▶ 例えばボタン押下時の処理にしても複数の方法があり、どの方法がいいかは場合によると思います。
- ▶ しかし、SharedPreferencesを使った設定画面の作成は簡単ですので、ぜひ試していただきたいと思います。

ご清聴ありがとうございました