# HELLO, ANDROID

*Android ハンズオンセミナー*

Eclipse で作成するプロジェクトのひな型が Hello, Android のプロジェクトになっています。新しいプロジェクトを作成しながら、内部の構造を確認していきましょう。

木南英夫

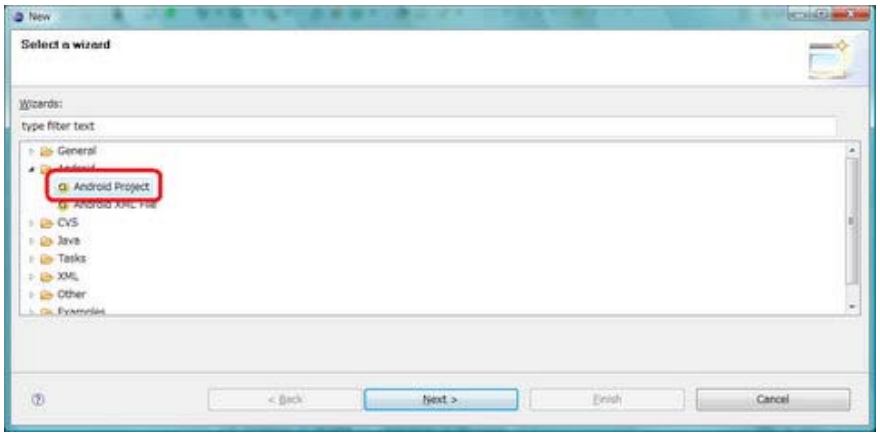
2009/08/06

# HELLO, ANDROID

*Android ハンズオンセミナー*

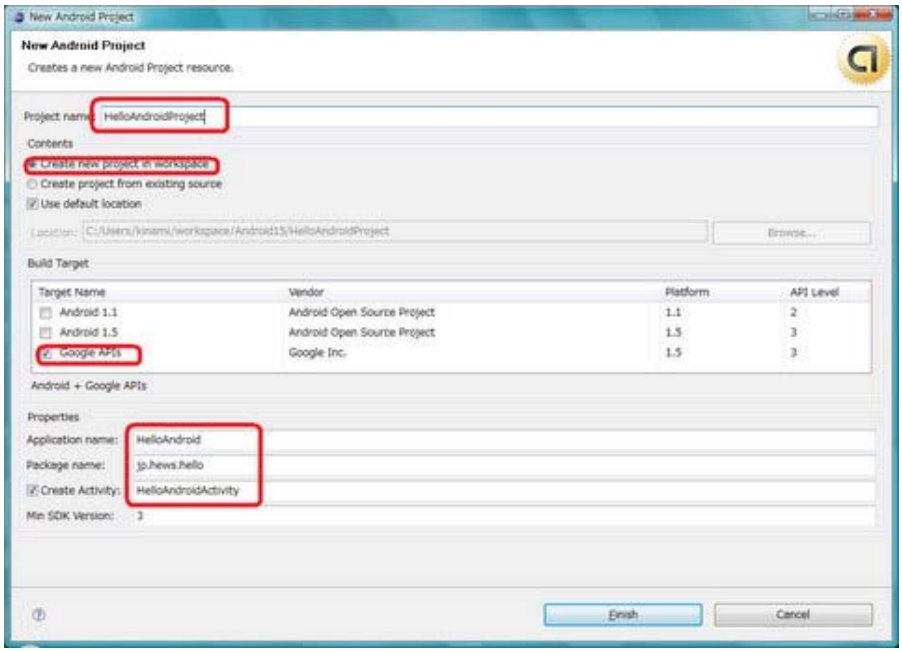
## ECLIPSEで新規のプロジェクトを作成する

新規のプロジェクトを作成します。File > New > Other で新規のプロジェクトの画面を表示します。



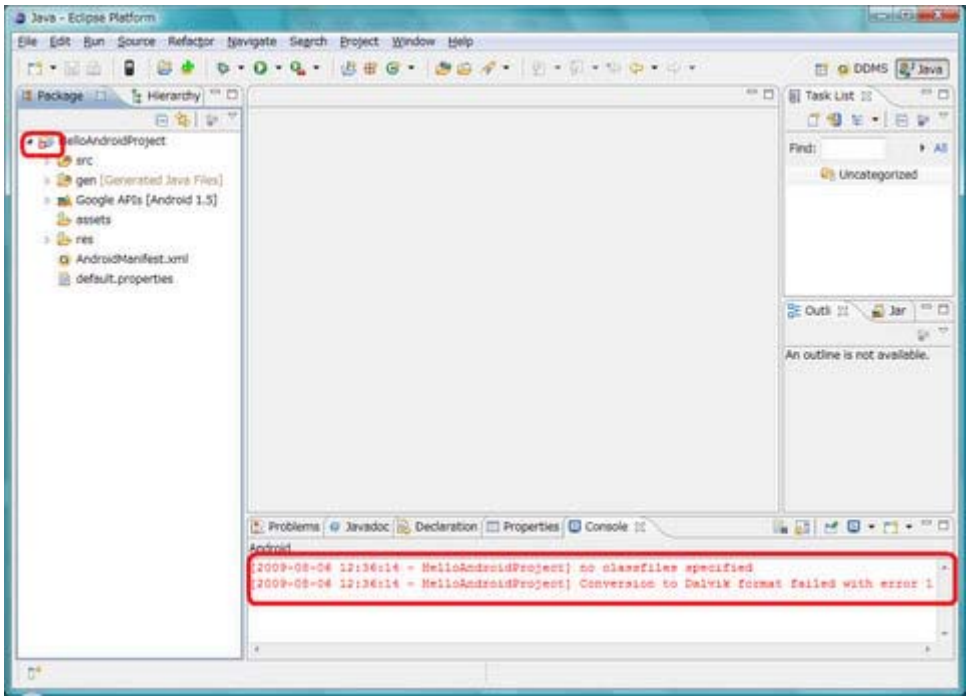

表示されたプロジェクトの画面で以下の項目を入力します。

| | |
|---|---|
| Project Name | HelloAndroidProject |
| Build Target | Google APIs |
| Application Name | HelloAndroid |
| Package Name | jp.hews.hello |
| Create Activity | HelloAndroidActivity |
| Min SDK Version | 3（Bild Target を指定すると自動的に設定される） |



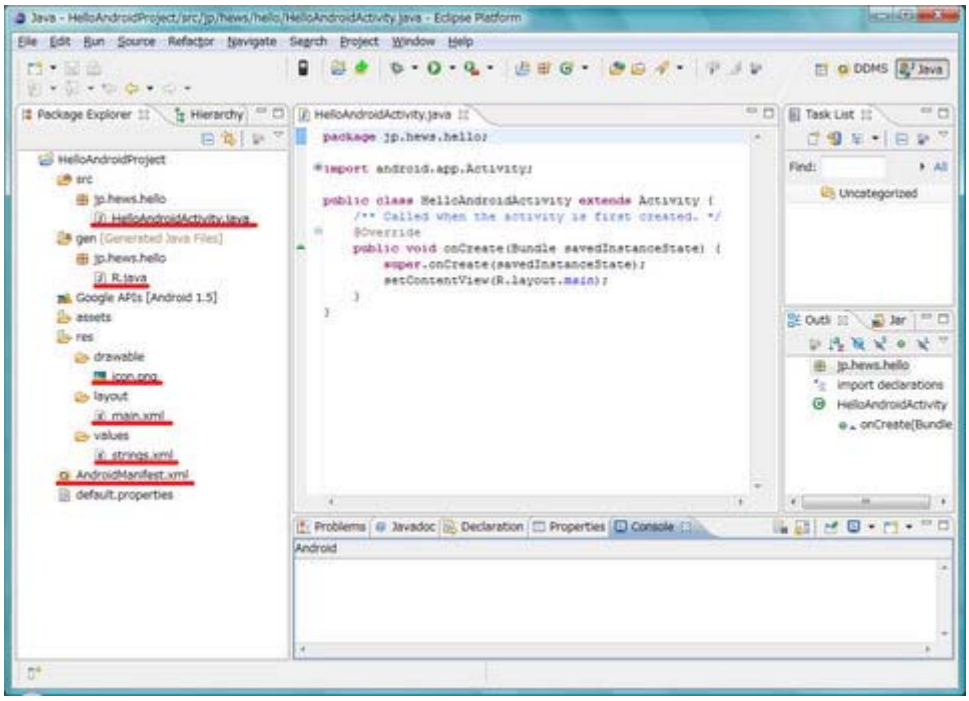
New Android Project の画面で Finish を押すとプロジェクトが作成されます。

初期状態では、クラスファイルが作成されていないので、エラーが発生する場合があります。このエラーは、Project > clean を選択するとエラーがクリアされます。



## プロジェクトの内容を確認する

それでは、プロジェクトの中に作成されているファイルを確認してみましょう。




フォルダー内に以下のクラスが作成されています。

| ファイル名 | 概要 |
|---|---|
| HelloAndroidActivity.java | アプリケーションの画面の制御 |
| res\drawable\icon.png | アプリケーションのアイコンのイメージ |
| res\values\strings.xml | 画面に表示する文字列の定義 |
| res\layout\main.xml | XML で定義した画面の構成 |
| gen\R.java | アイコンや文字列などのリソースへのアクセス用 ID |
| AndroidManifest.xml | アプリケーションの属性の定義 |

## HelloAndroidActivity.java

GUI を持つアプリケーションのメインのコントローラクラスです。

このクラスの、onCreate メソッドからアプリケーションが開始されます。onCreate メソッドでボタンのリスナーを登録したり、バックグラウンドの処理を開始します。

アプリケーションの終了時の処理や、画面が表示されなくなった時の処理は、ベースクラスの以下のメソッドをオーバーライドして実装します。

- onCreate アクティビティの開始時に呼び出されます
- onStart 画面がユーザーに表示される時に呼び出されます
- onResume アクティビティがユーザーと対話可能(キー操作の受付など)になるときに呼び出されます
- onPause ユーザーの操作を受け付けなくなった時に呼び出されます
- onStop 画面が表示されなくなったときに呼び出されます
- onRestart 非常時になった画面が再度表示される前に表示されます
- onDestroy アクティビティが終了するときに呼び出されます

## res\drawable\icon.png

ランチャーに表示されるアイコンです。このファイルを編集することでアイコンを変更することができます。

## res\values\strings.xml

画面に表示される文字列の定義です。Android では、プログラムのロジックと、表示される文字を分離することで国際化を容易にしています。

日本語用の文字列を res/values-jp/strings.xml に定義することで、システムの設定によって、日本語と英語の切り替えを行うことができるようになります。

## res\layout\main.xml

画面のレイアウトの定義です。Android では、プログラムのロジックと、画面の定義を分離することで、画面の大きさや見た目などのカスタマイズを容易にしています。

画面の定義は、縦方向と横方にならべる LinearLayout や、リスト形式に要素を並べる ListView、画像をスクロールしながら表示する Gallery などをしようすることができます。以下の URL におもなビューの一覧が示されています。

- http://developer.android.com/guide/tutorials/views/index.html

## gen\R.java

画像や文字列などをプログラムから参照するときに私用する ID が定義去れています。strings.xml を編集すると、Eclipse のプラグインが自動的に ID を採番して、このファイルを更新するので、直接編集してはいけません。

## AndroidManifest.xml

アプリケーションの属性を記述したファイルです。

アプリケーションのアイコンや、どのアクティビティのクラスを起動するかなどが定義されています。

# 演習問題

1) 表示されている文字列を「こんにちはアンドロイド」に変更してみましょう。

2) アプリケーションのアイコンの画像と、アイコンの下の文字を「ハロー」に変更してみましょう

3) 文字列の下にアナログ時計を配置してみましょう。

2009年8月6日

# ボタンを追加してみよう

*Android アプリケーション開発　ハンズオンセミナー*

ユーザー操作を受け取るために、画面上にボタンを作ってみましょう。ここでは、Android の Eclipse プラグインを用いてボタンを定義します。

日本 Android の会　木南英夫

2009/08/05

2009 年 8 月 5 日

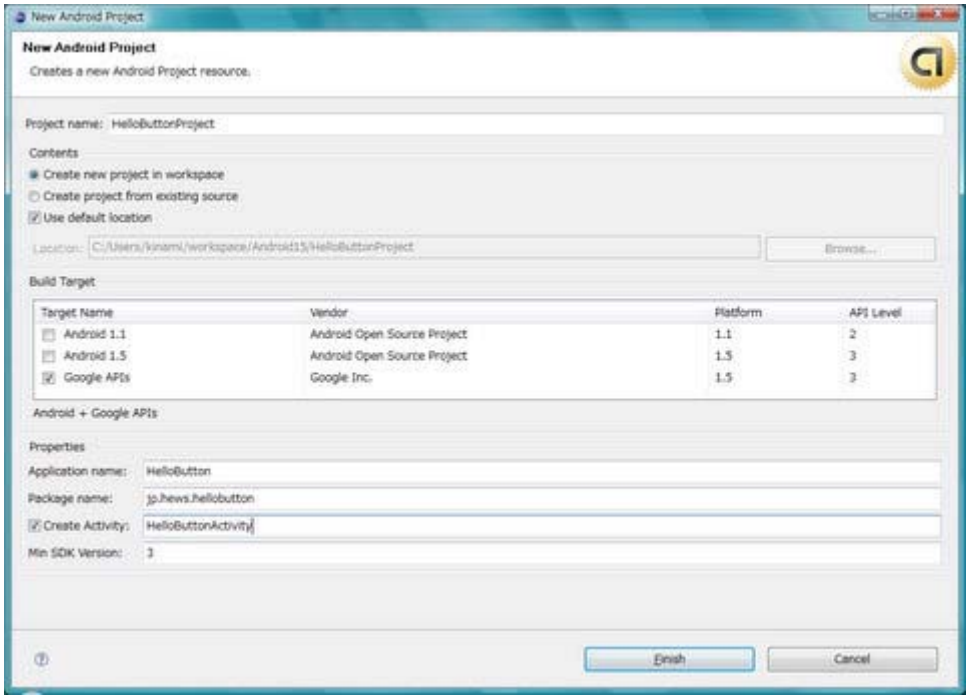
# ボタンを追加してみよう

*Android アプリケーション開発　ハンズオンセミナー*

## プロジェクトを作成する

必要に応じて、File > New > Android Project で新規のプロジェクトを作成します。

ここでは、以下のようなプロジェクトを作成してみます。



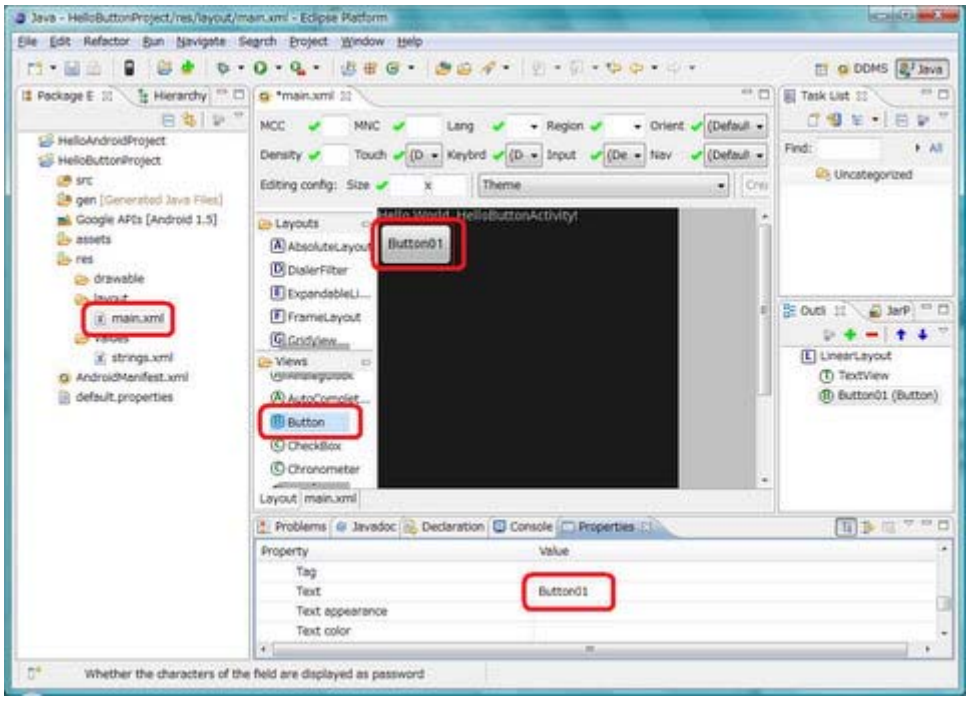

| | |
|---|---|
| Project Name | HelloButtonProject |
| Build Target | Google APIs |
| Application Name | HelloButton |
| Package Name | jp.hews.hellobutton |
| Create Activity | HelloButtonActivity |
| Min SDK Version | 3（Build Target を指定すると自動的に設定される） |

# 画面にボタンを配置する

画面にボタンを配置するには、レイアウトを定義した XML ファイルを編集します。

まず、レイアウトを定義した res/layout/main.xml を開いて、画面上にボタンを配置します。

この画面でボタンを配置するには、GUI で挿入する方法と、下部の「main.xml」タブから直接 XML を編集する方法があります。ここでは、GUI を用いて定義します。

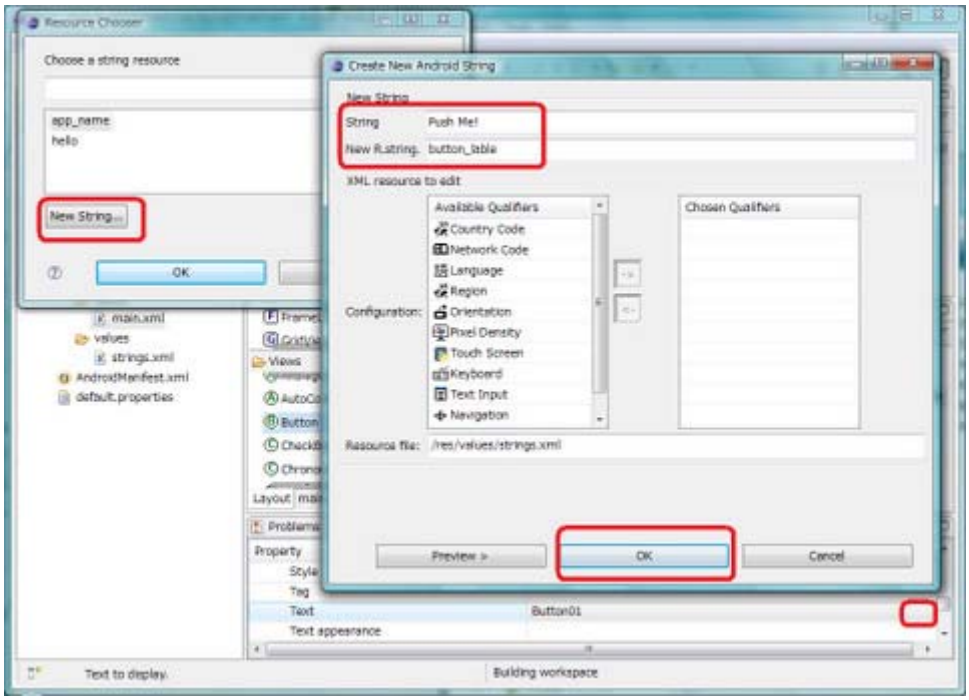



- Package Explorer で res/layout/main.xml をダブルクリックして main.xml を開きます。
- Views のフォルダーのなかの「Button」要素をドラッグして、画面に挿入します。
- Button01 という文字の表示されたボタンが配置されます

# ボタンに表示するラベルを定義する

ボタンのラベルは、res/values/strings.xml で定義します。ここでは、ボタンのプロパティから開かれる Resource Chooser を使用して、文字列を定義します。


- 下部の Properties タブで、Text 要素を選択して、Resource Chooser を選択します。




- 下部の「New String」を選択すると Create Android String の画面が開きます。
- String に表示する文字として「Push Me!」を入力します
- New R.string に、定義した文字列の ID として、「button_label」を入力します
- Preview を押して文字列が string.xml に反映されることを確認します。
- OK を押して、Resource Chooser の画面に戻ります

- いま定義した「button_lable」を選択して、「OK」を押します

# ボタンが押された時の処理を追加する

ボタンクラス（android.widget.Button）にリスナーを登録することで、ボタンが押された時の処理を追加することができます。

ボタンが押された時に呼び出されるリスナーの型は、android.view.View.OnClickListener です。

ここでは、メインのアクティビティクラスである HelloButtonActivity にリスナーを実装します。

- HelloButtonActivity をダブルクリックして開きます
- 先頭のインポート文の最後に実装に必要なボタンクラスとリスナークラスをインポートします。

```
import android.view.View;
import android.widget.Button;
```

- HelloButtonActivity クラスに「implements View.OnClickListener」を追加して、リスナーの実装を宣言します。

```
public class HelloButtonActivity extends Activity implements View.OnClickListener {
```

- HelloButtonActivity クラスに「public void onClick(View v)」メソッドを追加して、リスナーを実装します。
  - ✧ リスナーのメソッドの中では、finish()を呼び出します。このメソッドを呼び出すと、そのアクティビティを終了します。

```
  public void onClick(View v) {
    finish();
  }
```

- HelloButtonActivity クラスの onCreate メソッドで、ボタンにリスナーを登録します。
  - ✧ setContentView メソッドで、main.xml で定義した要素を表示します
  - ✧ findViewById メソッドで、main.xml で定義したボタンのインスタンスを取り出します。

- ✧ 取り出したボタンクラスの setOnClickListener メソッドに this を渡すことで、リスナーを登録します

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

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

- 作成したソースコードは、以下のようになっています。

```
package jp.hews.hellobutton;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class HelloButtonActivity extends Activity implements View.OnClickListener {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

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

    public void onClick(View v) {
        finish();
    }
}
```

## プログラムを実行する

修正したプログラムを実行して、表示されたボタンを押すことで、アクティビティが終了することが確認できます。


- Run メニューの Run で Run As 画面を表示する
- 「Android Application」を選択することで、アプリケーションが実行される
- 「Push Me!」というボタンを押すと画面が閉じて、ランチャーの画面が表示される

## 演習問題

- ボタンをふたつ横に並べて配置してみましょう。
- 二つのボタンのクリックを一つのリスナーのメソッドで受けて、どちらのボタンが押されたのかを判定してみましょう。
- イメージボタンや、ラジオボタンなど、別の種類のボタンを作ってみましょう。

# インテントを発行してみよう

*Android アプリケーション開発　ハンズオンセミナー*

インテントとは、アプリケーションから、新しいアクティビティを起動するためのパラメータです。起動するアクティビティは、同じアプリケーション内のアクティビティでも、他のアプリケーション内のアクティビティでも指定することができます。他のアプリケーションのアクティビティを指定するためには、アクションと、Uri という二つのパラメータを指定します。Android では、あらかじめ規定されたインテントにたいする動作が組み込まれています。

日本 Android の会　木南英夫

2009/08/06

2009 年 8 月 6 日

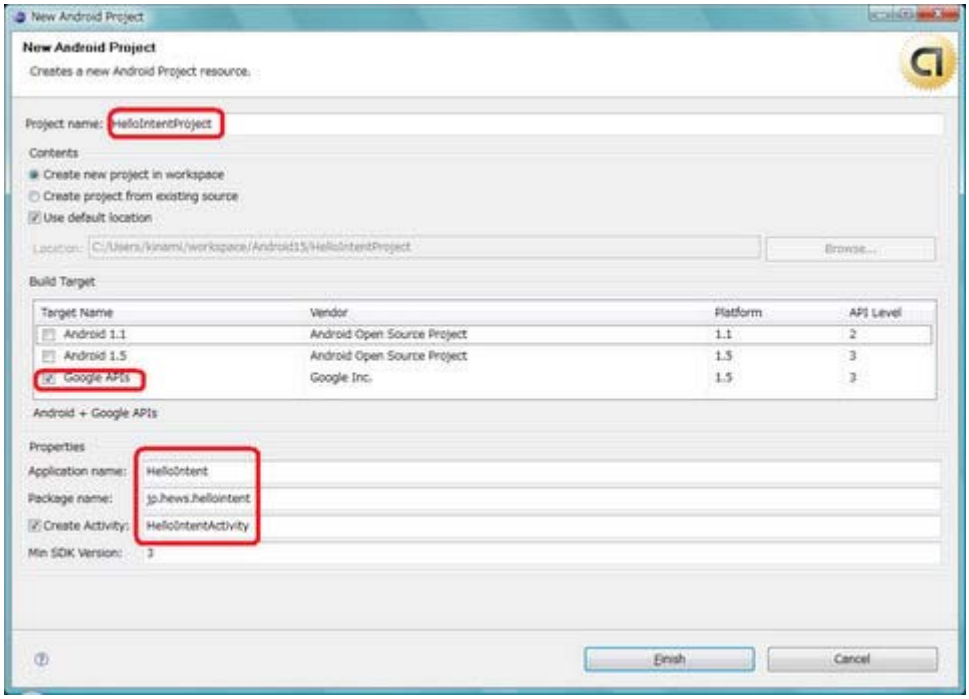
# インテントを発行してみよう

*Android アプリケーション開発　ハンズオンセミナー*

## プロジェクトを作成する

必要に応じて、File > New > Android Project で新規のプロジェクトを作成します。

ここでは、以下のようなプロジェクトを作成してみます。



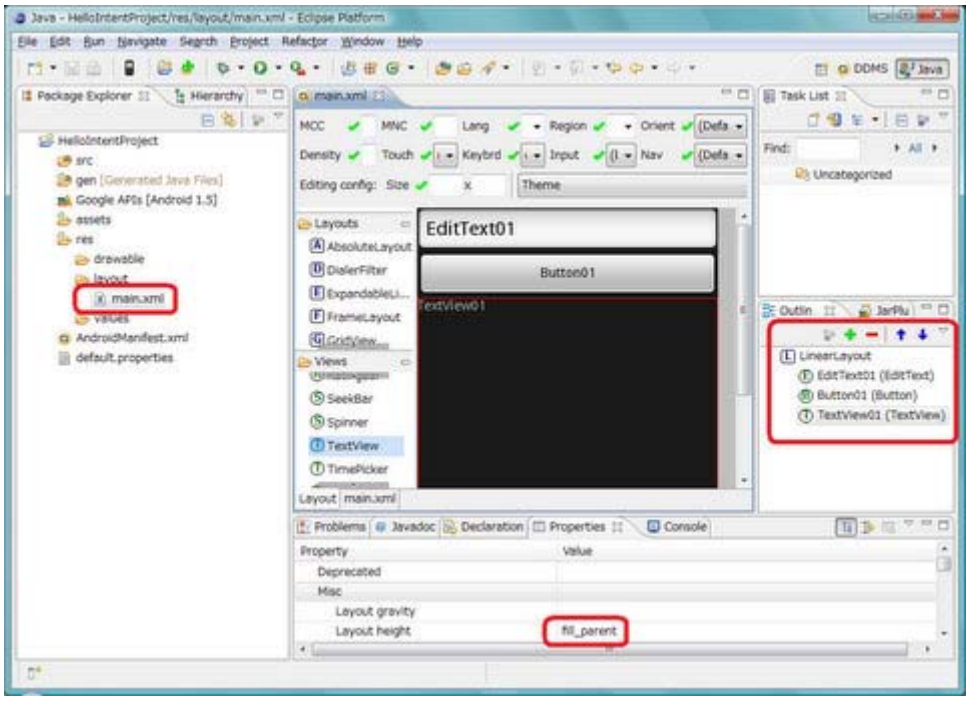

| | |
|---|---|
| Project Name | HelloIntentProject |
| Build Target | Google APIs |
| Application Name | HelloIntent |
| Package Name | jp.hews.hellointent |
| Create Activity | HelloIntentActivity |
| Min SDK Version | 3（Build Target を指定すると自動的に設定される） |

# 画面を定義する

画面にボタンを配置するには、レイアウトを定義した XML ファイルを編集します。

まず、レイアウトを定義した res/layout/main.xml を開いて、画面上にボタンを配置します。

この画面でボタンを配置するには、GUI で挿入する方法と、下部の「main.xml」タブから直接 XML を編集する方法があります。ここでは、GUI を用いて定義します。




- LinierLayout の最初に EditText(id:EditText01)を挿入します。layout_width を fill_parent に変更します。
- 2番目に Button(id:Button01)を挿入します。layout_width を fill_parent に変更します。
- 3番目に TextView(id:TextView01)を挿入します。layout_width と layout_hight を fill_parent に変更します。

# インテントを作成して発行するメソッドを定義する

EditText から文字列を取り出して、ACTION_VIEW のインテントを作成して、startActivity を呼び出します。

判定できない URI の場合には、エラーの内容をテキストビューに表示します。

以下のメソッドを HelloIntentActivity.java に定義してみましょう。

```
private void sendIntent() {
    try {
        EditText et = (EditText)findViewById(R.id.EditText01);
        Intent i = new Intent(Intent.ACTION_VIEW,
                        Uri.parse(et.getText().toString()));
        startActivity(i);
    } catch (Exception e) {
        TextView t = (TextView)findViewById(R.id.TextView01);
        t.setText(e.toString());
    }
}
```

- EditText の内容を Uri に変換して、インテントを作成します。
- 作成したインテントを引数にして startActivity を呼び出します。
- エラー発生時は、TextView にエラーを表示します。
- 必要なクラスのインポートは、Ctrl-Shift-O で挿入できます。

## ボタンにリスナーを登録する

ボタンが押された時に、定義した sendIntent を呼び出すリスナーを登録します。

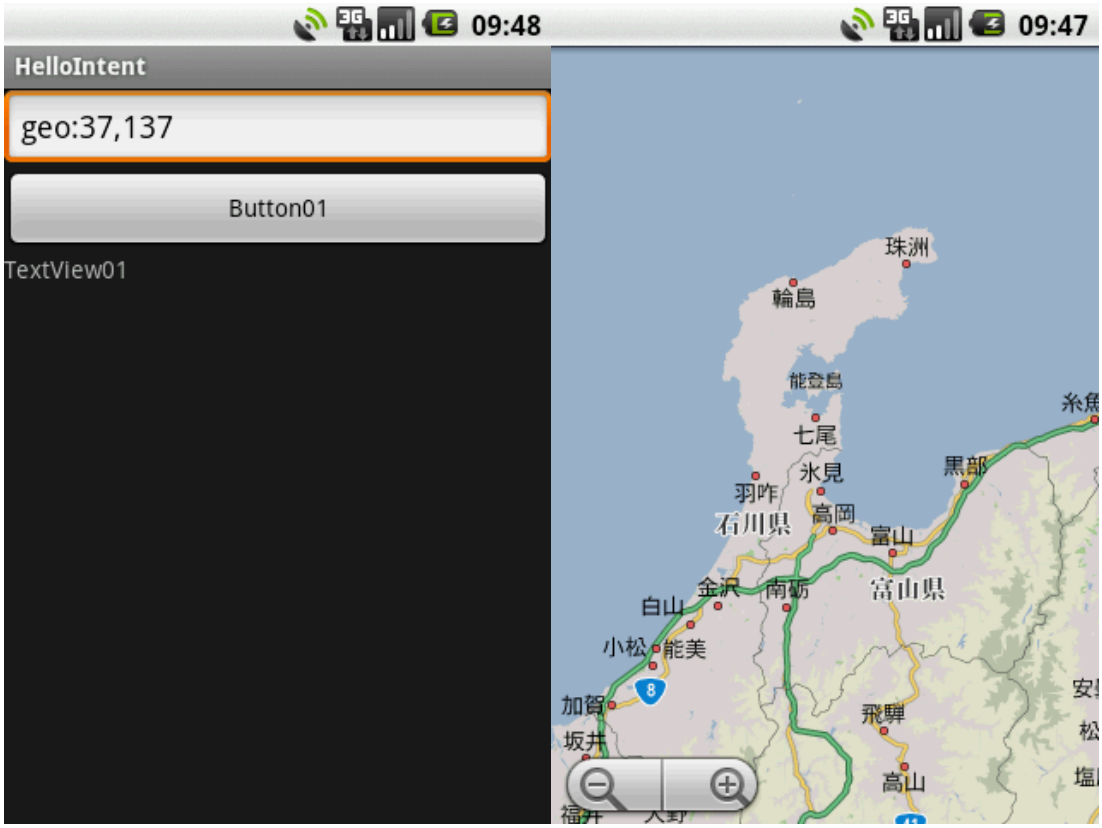
```
public class HelloIntentActivity extends Activity implements View.OnClickListener {

    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

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

    public void onClick(View arg0) {
       sendIntent();
    }
```

- HelloIntentActivity.java を開いて、onCreate メソッドでボタンを取り出して、リスナーを登録します。
- ボタンリスナーを作成して、リスナー内の onClick メソッドで次で定義する sendIntent メソッドを呼び出します。
- onClickListener でリスナーを登録します。

# URIを入力する

URI を入力して、あらかじめ組み込まれている Uri に対応するアクティビティを起動してみます。

| Uri | 動作 | 例 |
|---|---|---|
| http://web_address | ブラウザを起動する | http://www.google.com/ |
| tel:phone_number | ダイヤル画面を表示します | tel:123456 |
| geo:latitude,longitude | 地図を表示します | geo:37,137 |
| content://contacts/people | コンタクトリストを表示します | content://contacts/people/1 |



## ソースコード

作成したソースコードは、以下のようになっています。

```
package jp.hews.hellointent;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class HelloIntentActivity extends Activity implements View.OnClickListener  {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

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

    public void onClick(View arg0) {
        sendIntent();
    }

    private void sendIntent() {
        try {
            EditText et = (EditText)findViewById(R.id.EditText01);
            Intent i = new Intent(Intent.ACTION_VIEW,
                        Uri.parse(et.getText().toString()));
```

```
            startActivity(i);
        } catch (Exception e) {
            TextView t = (TextView)findViewById(R.id.TextView01);
            t.setText(e.toString());
        }
    }
}}
```

# 演習問題

- ActrionView 以外のアクションを画面から選択できるようにしてみましょう。
- 自分自身で定義したアクティビティをインテントで起動してみましょう。