

Hello, Android!

第一印象は重要だ。それは、あなたが、このAndroidというフレームワークを手にして、"Hello, World!"を書いたときに受ける第一印象だ。そう、Androidにおいて、それはとても簡単なのだ。下記を見て欲しい。

プロジェクトを作成する。

UIを構築する。

コードを走らせる: Hello, Android

以下のセクションでそれをつまびらかに語っていきこう。

UIをXMLのレイアウトにアップグレードする。

プロジェクトをデバッグする。

Eclipseなしでプロジェクトを作成する。

さあ行こう。

プロジェクトを作成する

プロジェクトを作成することはできる限り簡単にしてある。Eclipseプラグインで、Androidの開発環境のセットアップを作成することが可能だ。[Eclipse 3.3以上\(Europa\)](#)と、[Eclipse用のAndroidプラグイン](#)は用意してあるかい？それらをインストールしてから次に進んで欲しい。

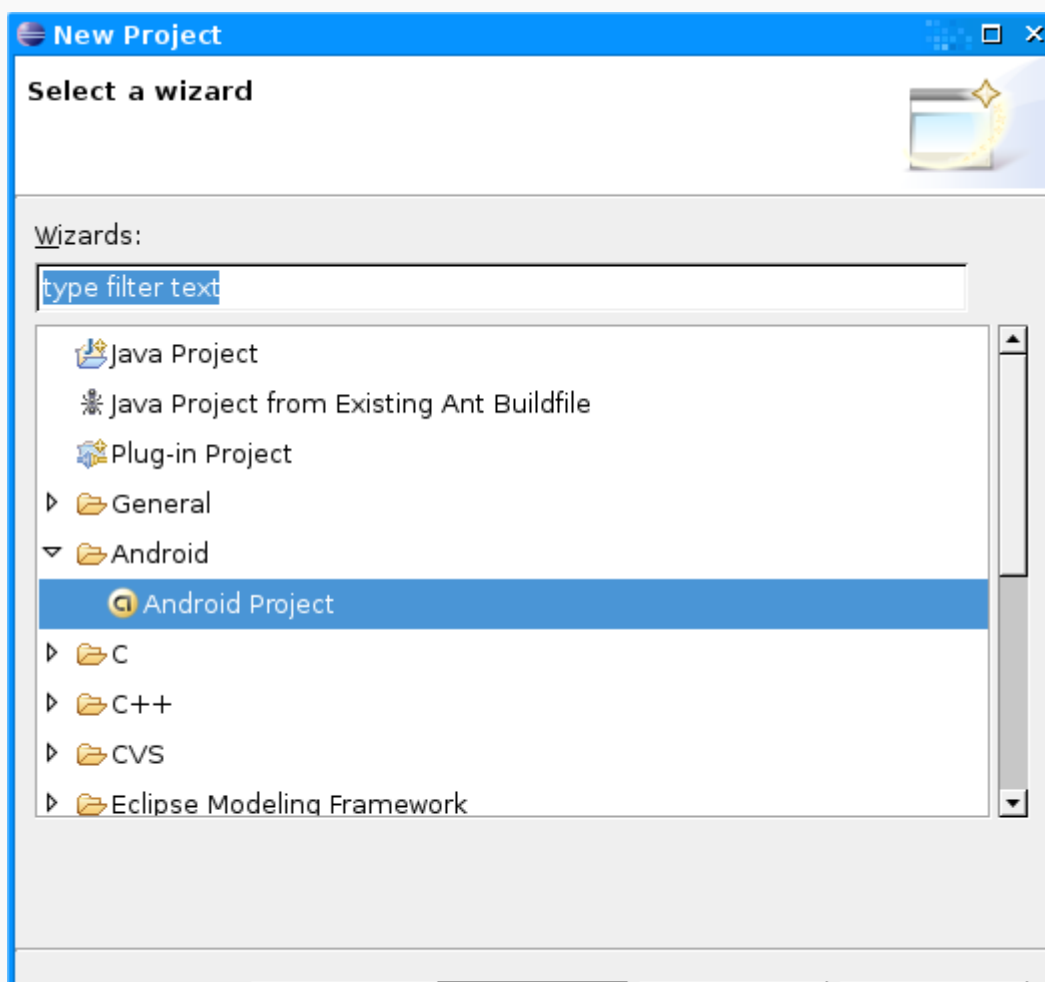
最初に、"Hello, World!"をビルドするための、簡単な要約を述べておこう。

1. File->New->Project menuから、"Android Project"を新しく作成する。
2. New Android Project ダイアログで、プロジェクトの詳細を埋める。
3. なにかを表示するための自動生成されたテンプレートコードを編集する。

さあ、行こう！以下でそれぞれのステップの詳細を説明しよう。

1 . 新しい"Android Project"を作成する。

Eclipseから、File->New->Projectと選択して欲しい。もし、EclipseのAndroidプラグインがきちんとインストールされているなら、表示されるダイアログの中に、"Android"と名前のついたフォルダがあり、その中には、"Android Project"があるはずだ。



"Android Project"を選択し、"Next"を押そう。

2 . New Android Project ダイアログで、プロジェクトの詳細を埋める。

次の画面で、プロジェクトに関する詳細を入力する。たとえば次の例のように：

New Android Project

Creates a new Android Project resource.

Project name: HelloAndroid

Contents

Create new project in workspace

Create project from existing source

Use deault location

Location: /opt/workspace/HelloAndroid

Properties

Package name: com.google.android.hello

Activity name: HelloAndroid

Application name: Hello, Android

それぞれの入力欄が意味するところは次のようになる。

Project Name	プロジェクトを保存したいディレクトリもしくはフォルダの名前
Package Name	これはパッケージの名前空間だ。ちょうどJavaのように、あなたのソースコードは全てここより下位におかれるようにする。ここでは、自動生成されたスタブのパッケージ名がすでにセットされているはずだ。パッケージ名は、システムにインストールされるすべてのパッケージ間で、ユニークである必要がある。というわけで、あなたのアプリケーションに標準的なドメイン命名スタイルを使うことはとても重要だ。上述の例では、パッケージ名として、ドメイン"com.google.android"を使用している。あなたの所属する組織にみあった、唯一の名前を使用するといいたいだろう。
Activity Name	ここでは、プラグインによって生成されるスタブクラスの名前が書かれている。これは、AndroidのActivityクラスのサブクラスである。Activityは単純なクラスで、それ自体で実行させ、処理させることができる。希望するならUIも作れるが、そうしなくても構わない
Application Name	ここにはユーザーが目にするアプリケーションのタイトルを入力する。

"Use default location"チェックボックスをONにすることで、プロジェクトファイルの保存場所を変更することができる。

3 . 自動生成コードを編集する。

After the plugin runs, you'll have a class named HelloAndroid that looks like this: プラグインを実行すると、下記のようなHelloAndroidクラスが出来上がっているがわかるだろう。

```
public class HelloAndroid extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
```

```
        super.onCreate(icicle);
        setContentView(R.layout.main);
    }
}
```

次のステップで、これを修正していこう！

UIを構築する。

プロジェクトをセットアップしたあとは、当然、それを修正していく。以下がその完成品だ。1行ずつ解剖していこう。

```
public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        TextView tv = new TextView(this);
        tv.setText("Hello, Android");
        setContentView(tv);
    }
}
```

このサンプルのコンパイルをするために、インポートセクションに、"import android.widget.TextView;"を追加するしなければいことを注意しよう。

Androidでは、ユーザーインターフェースは、Viewsと呼ばれるクラス階層で成り立っている。Viewはシンプルな描画オブジェクトである。たとえば、ラジオボタンであるとか、アニメーションであるとか、（今回のケースは）テキストラベルなどだ。テキストを扱えるViewのサブクラスの名前は、単に、TextViewとなる。

下記が、TextViewクラスを生成する方法だ。

```
TextView tv = new TextView(this);
```

TextViewクラスのコンストラクタに渡している引数は、Android Contextのインスタンスである。Contextは単にシステムへ渡すハンドルである。そのハンドルは、リソースを解決したり、データベースや設定などにアクセスするために供給されている。ActivityクラスはContextから派生している。それゆえ、HelloAndroidクラスはActivityクラスのサブクラスであり、コンテキストであるのだ。だから、"this"参照をTextViewに渡すことができる。

一度TextViewを生成してしまえば、何を表示するのか伝えてあげる必要がある。

```
tv.setText("Hello, Android");
```

とくに特筆すべきことはないだろう。

ここまでで、TextViewを生成し、どんなテキストをディスプレイに表示すべきかを伝えた。最後のステップは、実際のディスプレイに、TextViewをつなぐことである。こんな感じに。

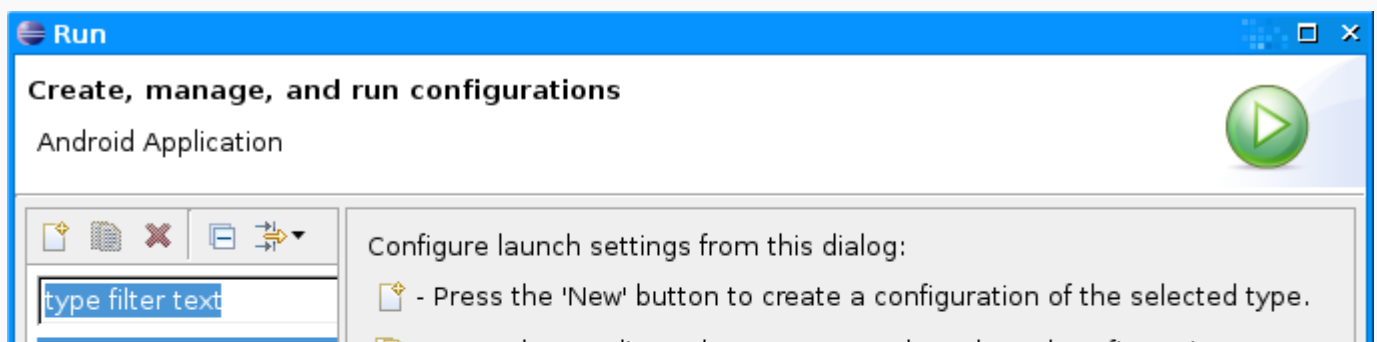
```
setContentView(tv);
```

ActivityのsetContentViewメソッドは、ActivityのUIにどのViewが関連付けられるべきかをシステムに通知する。もし、Activityがこのメソッドをコールしないなら、UIは何も表示されないし、システムは真っ白けの画面を表示することだろう。今のところの目的は、何でもいからテキストを表示することなので、作ったばかりのTextViewを渡してしまえばよい。

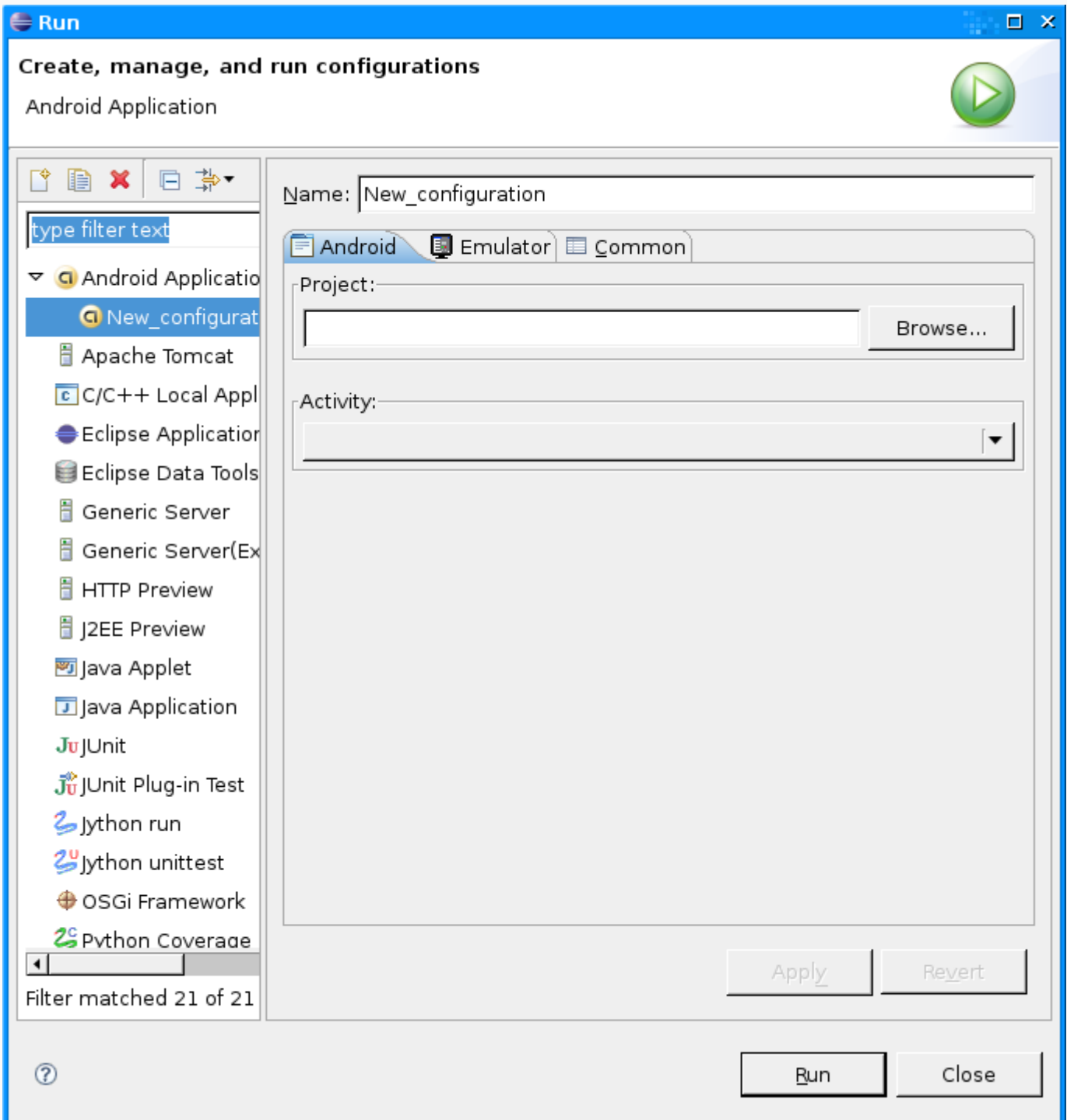
これで、アンドロイドでの"Hello, World"のコーディングは完了だ。もちろん、つぎは、実行させるてみよう。

コードの実行:Hello, Android

Eclipseプラグインのおかげで、とても簡単にあなたのアプリケーションを実行することができる。メニューからRunを選択すると、下のようなダイアログが表示される。

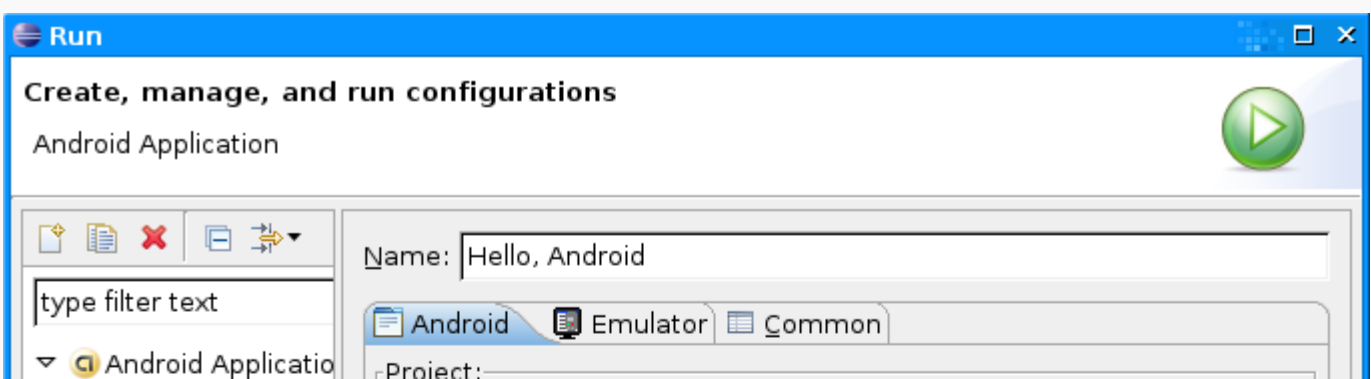


次に、"Android Application"を選択しよう。そして、アイコンの左上をクリックしよう(+印とともに、画面に描画されているやつだ)。それとも、単に、"Android Application"をダブルクリックするだけでいい。"New_configuration"と名づけられた新しいランチャーダイアログが表示されるはずだ。



名前を何か適当なもの、たとえば、"Hello, Android"と変更して、"Browse"ボタンを押下して、あなたのプロジェクトを選択しよう。(もしあなたが2個以上のAndroidプロジェクトをEclipseで開いていたら、正しいものを選択しているかどうか確かめてほしい) プラグインは、自動的に、あなたのプロジェクトからActivityのサブクラスをスキャンして、"Activity:"ラベルの下のドロップダウンリストに追加してくれる。デフォルトでは、あなたは、"Hello, Android"プロジェクトしか作っていないから、単に続けるだけでいい。

"Apply"ボタンを押下しよう。こういう風になる。



これで成功だ。"Run"ボタンを押してみよう。Androidエミュレータがスタートするはずだ。起動完了したら、あなたのアプリケーションが表示されるだろう。今までいったことが全部できていれば、次のような画面を目にすることができるはずだ。



これが、Androidにおける、「Hello, World」だ。とっても簡単だったろう？チュートリアル次のセクションでは、Androidについて、より詳細な価値ある情報を知ることができるだろう。

UIをXMLレイアウトにアップグレードする。

さっき終わらせた、「Hello, World」サンプルは、いわゆる「programmatic」なUIレイアウトだ。このことは、UI記述をソースコードに直接書いてビルドしているってことだ。UIプログラミングがおわっても、変更にもりやりかただってことはわかるだろう。たとえば、ちょっとしたUIのレイアウトの変更が、大きなソースコードの変更につながったりとか、Viewクラス同士のつながりは忘れやすいし、それがデバッグに時間を浪費することにつながる。

そんなわけで、Androidでは、もうひとつのUI構築のモデルを提供している。それが、XMLベースのレイアウトファイルだ。このコンセプトを説明するには一例をあげるのが一番だね。ここに、今終わらせたプログラミングベースのものと同じ振る舞いをするXMLレイアウトファイルを用意しよう。

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="Hello, Android"/>
```

たいていの、Android XMLレイアウトファイルの構成はシンプルだ。タグのツリーからなっており、それぞれのタグは、Viewクラスの名前になっている。この例で言えば、TextView一要素だけからなるシンプルなツリー構成だ。XMLレイアウトファイルには、タグ名として、Viewクラスを継承したものなら、自作のものでも何でも使える。これは、Webの構築モデルからインスパイアされたものなんだ。ちょうど、UIの表示とデータを処理するアプリケーションロジックを分離できるみたいに、この例では、4つのXML属性がある。以下が、その意味の要約だ。

Attribute	意味
xmlns:android	XML名前空間定義だ。これは、Android名前空間で定義された、共通の属性を参照するということをAndroidツールに知らせている。
android:layout_width	この要素は、このViewが消費する画面幅がどれくらいなのかを定義する要素だ。この場合で言えば、"fill_parent"を使っているが、画面全体の幅を指定していることになる。
android:layout_height	android:layout_widthと同じようなものだが、これは高さを意味する。
android:text	これは、TextViewの内容をセットするものだ。この例でいえば、いつもの"Hello, Android"だ。

そう、XMLレイアウトはざっとこんな感じだ。けど、どうやってそれを組み込むと思う？ resディレクトリの下に入れればOKだ。"res"は"resources"をはしょったもので、そのディレクトリには、アプリケーションに必要なコード以外の一式を詰め込んでおけばいい。たとえば、イメージや、ローカライズされた文字列や、XMLレイアウトファイルだ。

Eclipseプラグインは、XMLファイルを作成してくれる。上の例では単にそれを使わなかっただけだ。Package Explorerで、resフォルダの内容を開いて、main.xmlファイルに編集して、上のテキストをコピーして、変更を保存しよう。

Package Explorerのソースコードフォルダから、R.javaファイルを開いてみよう。次のようなものが表示されるはずだ。

```
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=0x7f040000;
    };
};
```

プロジェクトのR.javaファイルに全てのリソースにインデックスが定義されている。このクラスをソースコードのなかで、プロジェクトで使用しているリソースへの参照の速記方法の一つとしてつかえばいい。これはEclipseのようなコードコンプリート機能を持つIDEでは特にパワフルだ。なぜなら、その機能のおかげで、すばやく、インタラクティブに探しているリソース参照を配置することができるからだ。

この例で注意すべきは、"layout"と命名されたインナークラスと、"main"と命名されたフィールドだ。新しいXMLレイアウトファイルを追加したら、Eclipseプラグインは通知し、R.javaファイルを再生成するだろう。つまり、他のリソースファイルをプロジェクトに追加するなら、R.javaファイルも更新されるのがわかるだろう。

最後に、あなたのHelloAndroidコードを、ハードコーディングされたバージョンからXMLのUIを使うように修正する必要がある。新しいクラスはこのようになるはずだ。見ればわかるように、ソースコードはよりシンプルになった。

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

この変更を加える際に、コピーアンドペーストをしないことだ。Rクラスに働くコードコンプリート機能を試してみよう。これがかなり使える機能だってことがわかるだろう。

変更を加えたら、アプリケーションを再起動してみよう。することと言えば、緑のRunと書かれた矢印アイコンをクリックし、Run -> Run Last Launchedをメニューから選択するだけだ。すると、さっき見たのと同じものが表示されるはずだ！結局のところ、2つの違ったレイアウトのアプローチは同じ結果をもたらすわけだ。

XMLレイアウトの作成はもっと説明することがあるんだが、今ここで説明してもしょうがない。このアプローチのもっと詳細な情報のためには"Implementing a User Interface"ドキュメントを読んで欲しい。

プロジェクトのデバッグ

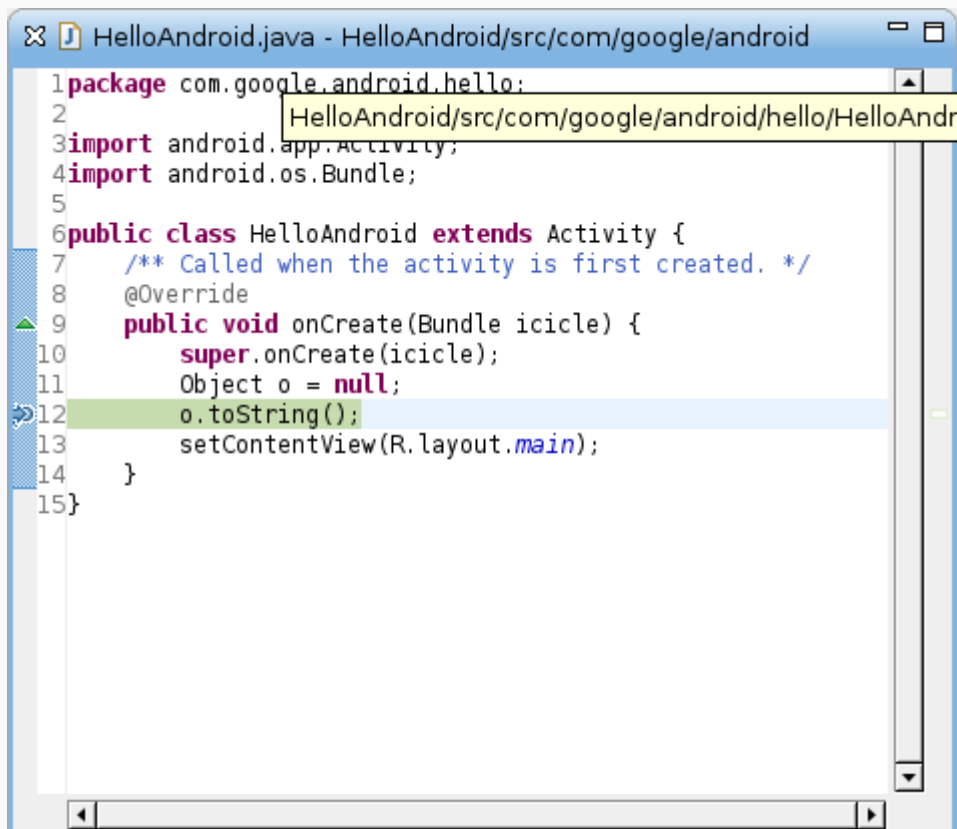
EclipseのAndroidプラグインは、Eclipseデバッガにうまく統合されてもいる。それをデモするために、コードにバグを混入させてみよう。次のようにHelloAndroidのソースコードに変更を入れて欲しい。

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

これは単に、NullPointerExceptionを入れてみたただけだ。もう一度アプリケーションを実行させると、すぐに次のような画面を見るだろう。



何かおかしかったのかを特定するために、ソースコードの"Object o = null;"と書かれた行にブレークを貼ってみよう。(ブレークポイントを貼るためには、Eclipseの行番号の左の領域をダブルクリックすればいい)それから、Run -> Debug Last Launched を選択してデバッグモードに入ろう。エミュレータが再起動すると、セットしたブレークに到達した時点でアプリケーションは中断する。EclipseのDebug Perspectiveを通してどんなアプリケーションでもステップ実行ができる。



```
1 package com.google.android.hello;
2
3 import android.app.Activity;
4 import android.os.Bundle;
5
6 public class HelloAndroid extends Activity {
7     /** Called when the activity is first created. */
8     @Override
9     public void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         Object o = null;
12         o.toString();
13         setContentView(R.layout.main);
14     }
15 }
```

Eclipseなしでプロジェクトを作成する。

もし、あなたがEclipseを使用しないなら(たとえば別のIDEがいいだとか、そもそも単にテキストエディタとコマンドラインツールが使いたいなら) Eclipseプラグインはあなたの助けにはならないだろう。でも心配しなくていい、Eclipseを使用しないからといって、あなたは何も便利さを失わないのだ。

EclipseのAndroidプラグインは、Android SDKに同梱されているツール群のラッパーにすぎないからだ。(ツール群とは、エミュレータや、aapt,adb,ddmsなどだ。ほかにもどこかにドキュメントがあるだろう) それゆえ、他のツールを使って、それらをラッピングすることは可能だ。たとえば'ant'のビルドファイルを使うなどして。

Android SDKは、"activityCreator.py"とネーミングされたPythonスクリプトも含んでいる。それは、antと互換性のあるbuild.xmlファイルはもちろんプロジェクトに必要なソースコードとスタブディレクトリを全て作成してくれる。これは、コマンドラインでプロジェクトをビルドできることと、あなたの好きな他のIDEと統合できることを意味する。

たとえば、Eclipseを通してさっき作ったものと似たようなHelloAndroidプロジェクトを作成するためには、次のコマンドを使えばいい。

```
activityCreator.py --out HelloAndroid com.google.android.hello.HelloAndroid
```

プロジェクトをビルドするためには、'ant'コマンドを走らせればいい。コマンドが成功すれば、'bin'フォルダの下にHelloAndroid.apkと名づけられたファイルがあるはずだ。この.apkファイルは、Android Packageで、'adb'ツールを使ってエミュレータにインストールして実行できる。

これらのツールについての使用方法をもっと知りたければ、上で紹介されたドキュメントを読んで欲しい。