# Google Map を使ってみよう

*Android アプリケーション開発　ハンズオンセミナー*

Android で Google Maps を使用したアプリケーションを作成してみましょう。
Android には、MapView, MapActivity という Google Map を簡単使用するクラスが用意されています。

日本 Android の会　木南英夫

2009/08/07

2009 年 8 月 7 日

# GOOGLE MAP を使ってみよう

*Android アプリケーション開発　ハンズオンセミナー*

## 作成するアプリケーション

Android 上に地図を表示して、タップした位置にアイコンを描画するアプリです。以下のページを参考にしながら、作成することができます。

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

ここでは、上記のページの一部を簡略化したサンプルを作成してみます。

Google Map API のドキュメントは、以下のサイトで参照することができます。

- http://code.google.com/intl/ja/android/add-ons/google-apis/

## 作成手順の概略

アプリケーションを作成する手順の概略です。

- Google Maps の API キーを取得します
  - アプリに署名する証明書が必要です
- 新規のプロジェクトを作成します
  - Google API を選択します
- マニフェストファイルを設定します
  - ライブラリやパーミッションの設定が必要です
- MapView で地図を表示します
  - 取得した API キーを使用します
- MapActivity で地図の表示を制御します
  - ズームコントローラで地図の拡大縮小できます
- タップされた位置にアイコンを描画します
  - Overlay クラスで地図上に描画できます

# GOOGLE MAPS の API キー取得

Google Maps は Google が提供する Web 上のサービスです。Android では、このサービスにアクセスするための専用のライブラリが提供されています。

このライブラリは、Google が提供しているものであり、オープンソースの Android の一部ではありません。このため、このライブラリを使用するには、Google に開発者としての登録を行う必要があります。登録を行うと、API キーという文字列が発行されます。この文字列をアプリケーション内に組み込むことで、サービスにアクセスすることが可能になります。

API キーの取得には、アプリケーションの開発時に使用する証明書のハッシュが必要になります。証明書とは、開発者の名前や所属に、秘密鍵で署名したものです。


Android のアプリケーションのパッケージには、開発者自身の証明書による署名が必要になります。デバッグには、開発ツールが自動的に作成した証明書が使用されているのであまり意識することがありませんが、エミュレータや携帯端末の実機に転送する前に常に署名がおこなわれています。

証明書の MD5 のハッシュは、Sun の JDK に含まれている keytool という証明書操作用のツールを使用します。まずは、keytool がインストールされているか、以下のコマンドを実行して確認しましょう。

```
> keytool -help
```

もし、コマンドが見つからない場合には、JDK をインストールしたディレクトリの bin ディレクトリを確認してみましょう。以下の、%JAVA_HOME%を JDK をインストールしたディレクトリに置き換えて実行してみましょう。うまく実行できれば、PATH 環境変数に、%JAVA_HOME%\bin を追加しておきます。

```
> "%JAVA_HOME%\bin\keytool" -help
...
```

つぎに、使用している証明書の MD5 のハッシュを取ります。デバッグ時に使用される証明書は、使用している OS に応じて、以下の場所に格納されています。

- - OS X ~/.android/debug.keystore
- - Linux ~/.android/debug.keystore
- Windows Vista C:\Users\%USERNAME%\.android\debug.keystore
- Windows XP C:\Documents and Settings\%USERNAME%\.android\debug.keystore

keytool の-list コマンドで証明書の一覧を表示します。例えば、SDK1.5 の Vista の場合には、以下のコマンドを実行します。パスワードは、入力しないで単純にリターンを押すだけで大丈夫です。（入れたい人は、"android"と入れてください）

```
>  keytool -list -keystore C:\Users\%USERNAME%\.android\debug.keystore
キーストアのパスワードを入力してください: android

キーストアのタイプ: JKS
キーストアのプロバイダ: SUN

キーストアには 1 エントリが含まれます。

androiddebugkey, 2008/10/14, PrivateKeyEntry,
証明書のフィンガープリント (MD5): 4B:28:72:FB:D3:2C:86:D5:A5:F8:B4:BA:09:C3:90:29
```

ここで表示されたフィンガープリントを、以下のページで入力します。

- http://code.google.com/intl/ja/android/maps-api-signup.html

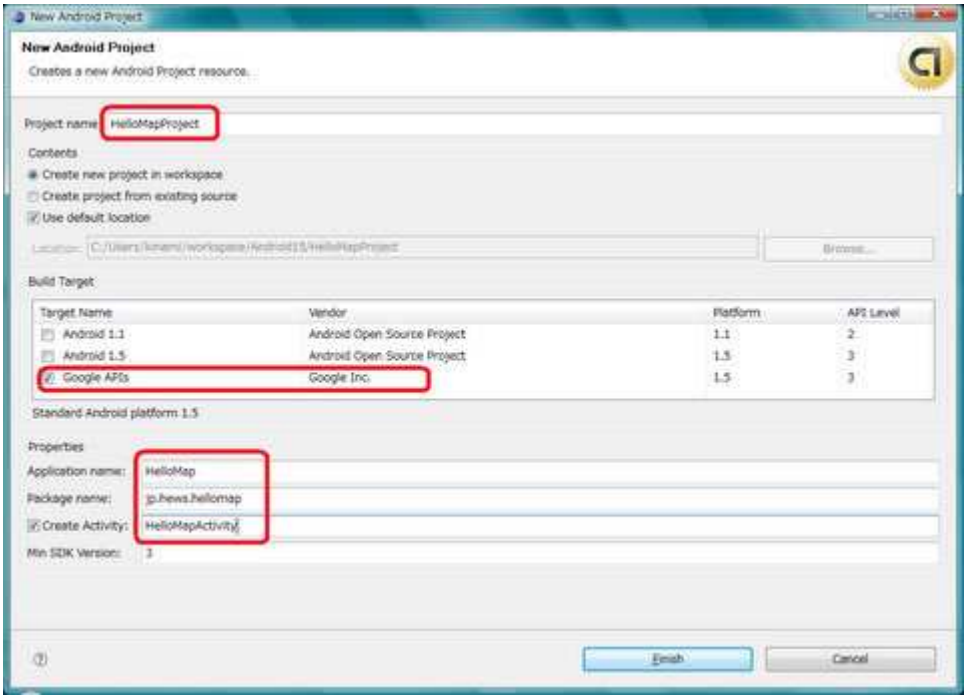
生成したフィンガープリントは、後で使用するので、控えておきます。

アプリケーションの署名は、端末のインストール時と、マーケットへのアップロード時に確認されます。

# 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。



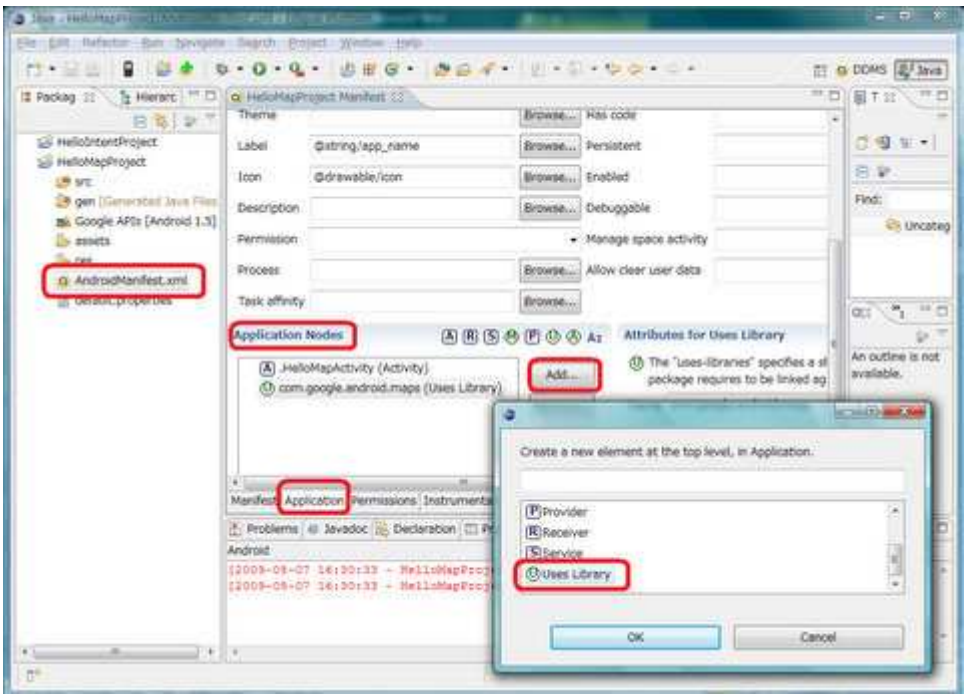

| | |
|---|---|
| Project Name | HelloMapProject |
| Build Target | Google APIs |
| Application Name | HelloMap |
| Package Name | jp.hews.hellomap |
| Create Activity | HelloMapActivity |
| Min SDK Version | 3（Build Target を指定すると自動的に設定される） |

# マニフェストファイルの設定

地図を使用するには、マニフェストの application 内にライブラリを使用する宣言をします。

Eclipse の GUI では、AndroidManifest の Application タブの下の「Application Nodes」で設定します。

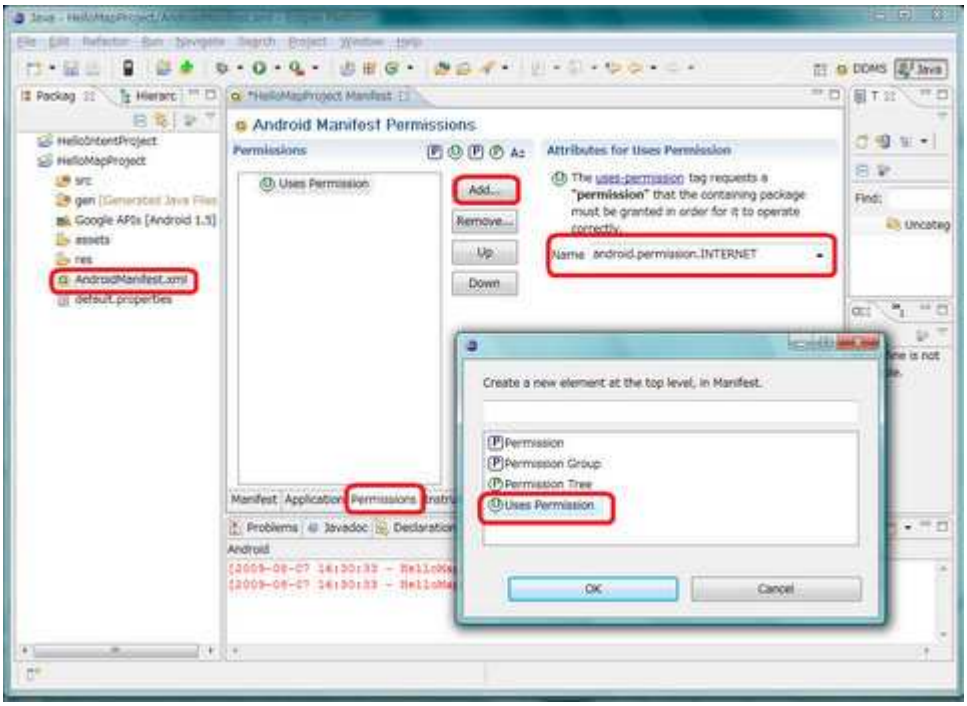
- <uses-library android:name="com.google.android.maps" />




また、インターネットに接続するために、同じくマニフェスト内で以下のパーミッションの使用を宣言します。

Eclipse の GUI では、AndroidManifest の Permission タブで設定します。

- <uses-permission android:name="android.permission.INTER




## レイアウトの作成

- レイアウトリソースファイル（res/layout/main.xml）を以下のように書き換えます。

```
<?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" >

    <com.google.android.maps.MapView
        android:id="@+id/mapview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:clickable="true"
        android:apiKey="取得した API Key" />
 </LinearLayout>
```

## MAPVIEW クラスを作成する

- MapView クラスを表示して、MapActivity を継承させます。これによって、MapView の制御をこのクラスが行えるようになります。(Shift-Ctrl-O で com.google.android.maps.MapActivity をインポートします。)

```
public class HelloMapView extends MapActivity
```

- Add unimplement method を使用して、isRouteDisplayed メソッドをオーバーライドします。

```
@Override
protected boolean isRouteDisplayed() {
    // TODO Auto-generated method stub
    return false;
}
```

ここで、アプリケーションを実行して、地図が表示されることを確認しましょう。もし、地図が表示されてない場合には、以下の項目を確認してみましょう。

- SDK にあらかじめインストールされている Map アプリケーションで地図が表示されるか確認する。
  - 表示されてなけば、ネットワークの設定など OS の設定を確認する。Proxy が設定されていると、表示できないので注意する。
- Map の API キーが正しく入力されているか確認する。API キーの取得は何度でも行えるので、再度取得し直してみる。
- 使用している証明書が正しいか確認する。Windows では、SDK1.1 から SDK1.5 で証明書の位置が変更になっているので、注意する

## ズームコントローラを有効にする

onCreate で、ZoomControl を有効にします。

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

```
    MapView mapView = (MapView) findViewById(R.id.mapview);
    mapView.setBuiltInZoomControls(true);
}
```

実行して、地図を表示してみましょう。タップするとズームコントローラが表示されます。

## 初期値を設定する

以下のフィールドを定義します。

```
// ログ出力用のタグ
static final String TAG = "HelloMapActivity";

// 地図の初期値
static final int INITIAL_ZOOM_LEVE = 16;
static final int INITIAL_LATITUDE = 36564000;
static final int INITIAL_LONGITUDE = 136661000;
```

onCreate の後に、以下の行を追加します。

```
    // 位置とズームレベルの初期状態を設定する
    MapController controller = mapView.getController();
    GeoPoint point =
        new GeoPoint(INITIAL_LATITUDE, INITIAL_LONGITUDE);
    controller.setCenter(point);
    controller.setZoom(INITIAL_ZOOM_LEVE);
```

実行してみましょう。

# タップされた位置にアイコンを描画する

Overlay クラスを使用して、地図上に描画できます。

コンストラクタの最後に以下の呼び出しを追加します。

```
    // イメージを地図上に表示する
    setOverlay(point);
```

以下のプライベートメソッドを定義します。

ここでは、アプリケーションのアイコンを地図上に描画するので、アプリケーションのアイコンを好きな柄に書き換えておきましょう。

```
private void setOverlay(GeoPoint point) {

    // Overlay クラスを生成する
    Bitmap icon = BitmapFactory.decodeResource(getResources(),
                                        R.drawable.icon);
    IconOverlay overlay = new IconOverlay(icon, point);

    // 生成した Overlay クラスを追加する
    MapView map_view = (MapView) findViewById(R.id.mapview);
    List<Overlay> overlays = map_view.getOverlays();
    overlays.add(overlay);
}

// 地図上に表示されるオーバーレイ
private class IconOverlay extends Overlay {

    // 描画するアイコン
    Bitmap mIcon;
    int mOffsetX;
    int mOffsetY;

    // アイコンを表示する位置
    GeoPoint mPoint;

   IconOverlay(Bitmap icon, GeoPoint initial) {
       // アイコンと、アイコンの中心のオフセット
```

```
        mIcon = icon;
        mOffsetX = 0 - icon.getWidth() / 2;
        mOffsetY = 0 - icon.getHeight() / 2;
        mPoint = initial;
    }

    // 地図のタップ時に呼び出される
    @Override
    public boolean onTap(GeoPoint point, MapView mapView) {
        // タップされた位置を記録する
        mPoint = point;
        Log.i(TAG, "Point = " + point.getLatitudeE6() + " , " + point.getLongitudeE6());
        return super.onTap(point, mapView);
    }

    // 地図の描画時に、shadow=true, shadow=false と 2 回呼び出される
    @Override
    public void draw(Canvas canvas, MapView mapView,
                boolean shadow) {
        super.draw(canvas, mapView, shadow);
        if (!shadow) {
            // 地図上の場所と、描画用の Canvas の座標の変換
            Projection projection = mapView.getProjection();
            Point point = new Point();
            projection.toPixels(mPoint, point);
            point.offset(mOffsetX, mOffsetY);
            // アイコンを描画
            canvas.drawBitmap(mIcon, point.x, point.y, null);
        }
    }
};
```

実行してみましょう。

# 演習問題

- タップされた緯度経度情報をもとに、グルメ検索などの Web サービスから情報を取得してみましょう
- GPS でロケーションを取得して、Map 上に現在地を表示してみましょう