Android-A-Yan 分科会 第１回

Android アプリをつくろう

地縛伝言® ver 0.4

地縛伝言® アプリの作成

今回は、入門プログラム的な一つの機能に的を絞ったサンプルを作るのではなく、ある程度実践的なアプリを作ってみたいと思います。アプリ作成の流れ全体を通して、なるべく多くのキーワードを出すことで、後で調べたり学習するためのとっかかりになることを狙っています。その中で適宜、概念や手法、Tips など、自分がこの半年で得たことを最大限伝えられたらと考えています。

地縛伝言® アプリの概要

まず全体像を把握しておくために、どんなアプリをつくっていくかを説明します。

1. 地縛伝言アプリはメッセージを GAE アプリケーションに投稿
2. GAE アプリに、投稿されたメッセージは、投稿された位置とともに保存される
3. 地縛伝言アプリは、端末の位置が変わると自端末からある距離内のメッセージを GAE アプリに要求
4. GAE アプリは、要求された範囲にあるメッセージを返す
5. 地縛伝言アプリは、取得したメッセージを表示

では、以下の時間で、上記を作り上げていきたいと思います。

ちなみに、開発環境は作成できていることが前提です。

1．単体で動くアプリを作る（45 分)

2．位置情報を利用して WEB アプリと連携（45 分)

3．サービスをつくる（45 分)

4．その他諸々の機能（45 分)

5．最後に

1.単体で動くアプリを作る

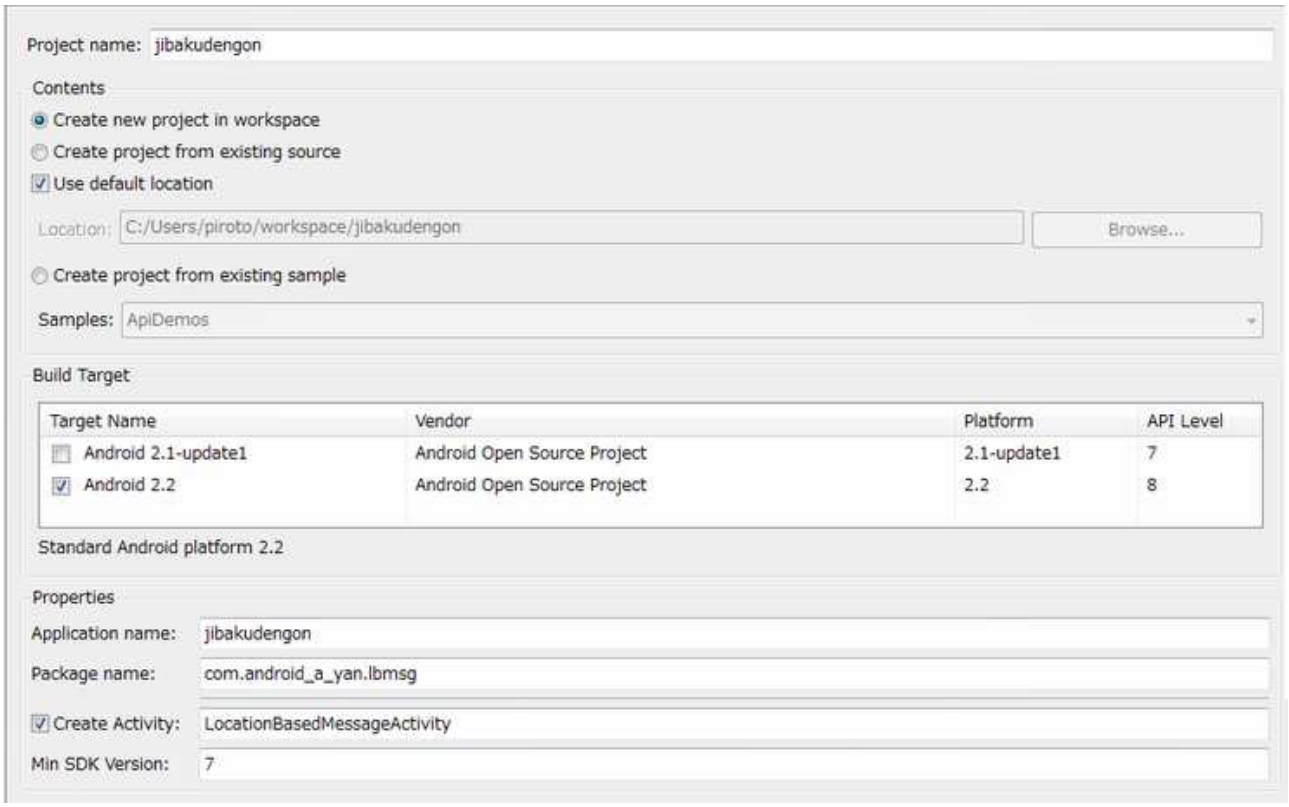
まずは、単体で動くアプリケーションの基本的な作成手順の、画面作成からイベントの実装までを流してみたいと思います。

1.1.プロジェクトの作成

まず、Eclipse に Android アプリのプロジェクトを作成します。

1.1.1 File – New – Other – Android Project を選択

New Android Project



API Level：Android プラットフォームの各バージョンの API を一意に識別

Min SDK Version ： サポートする最低の API レベルを指定すればいい。Build Target と一致する必要はない。例えば、Android2.1 端末に対応させるために Min SDK Version を 7 としておいて、実際には Android 2.2 SDK を利用して開発することが可能。Android2.1 に対応させつつ、2.2 端末では、2.2 からサポートされた、アプリケーションを SD カードへ移動させることが可能になる。

1.1.2 いったん動かしてみる

実機をつなぐか、エミュレータを作成して、プロジェクトのコンテキストメニューから、Debug As – Android Application で一度、アプリを動かしてみましょう。

1.2.画面をつくろう

1.2.1 Activity

プロジェクトの作成で、Create Activity にチェックを入れていると、自動で Activity が作成されているはずです。Activity とは、Windows アプリケーションでのウィンドウのようなもので、ユーザーとのインターフェースを受け持ちます。ウィンドウとは違って、一度に１つしか表示できません。新しい Activity を起動すると、どんどん既存の Activity の上に積み重なっていき、一番上(最前面)の Activity を終了すると、下の Activity があらわれます。要するにスタック構造です。

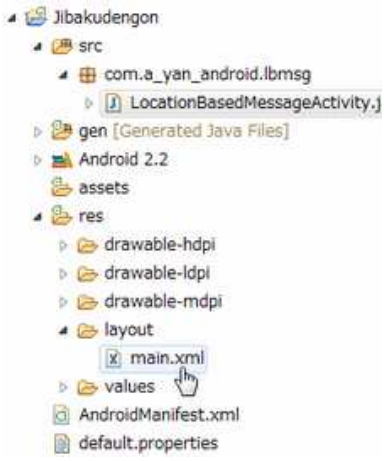
```
package com.a_yan_android.lbmsg;

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

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

Activity は、ウィンドウのようなものといいましたが、ウィンドウとおなじようにボタンやテキストボックスを配置することができます。Swing を使って画面を書くように、ソースコードでごりごり画面を生成するコードを記述することもできますが、XML で画面レイアウトを定義することもできます。校舎の方が、レイアウトとロジックが分離でき、個人的には好みです。で、上記ソースコードのなかで、マーカーでマークした、setContentView(R.layout.main) が、XML のレイアウト定義を 読み込んで、Activity の画面を作成する部分です。 “R”というのは、SDK により、コンパイル時に自動で生成される(/gen フォルダを確認してみてください)クラスです。XML による画面定義を含む様々なリソース(アイコン、文字列、色・・・)の ID 管理を行ってくれます。これにより、画面定義がしてある main.xml という物理ファイルを意識せずに各所にて利用できるようになっています。ちなみに、

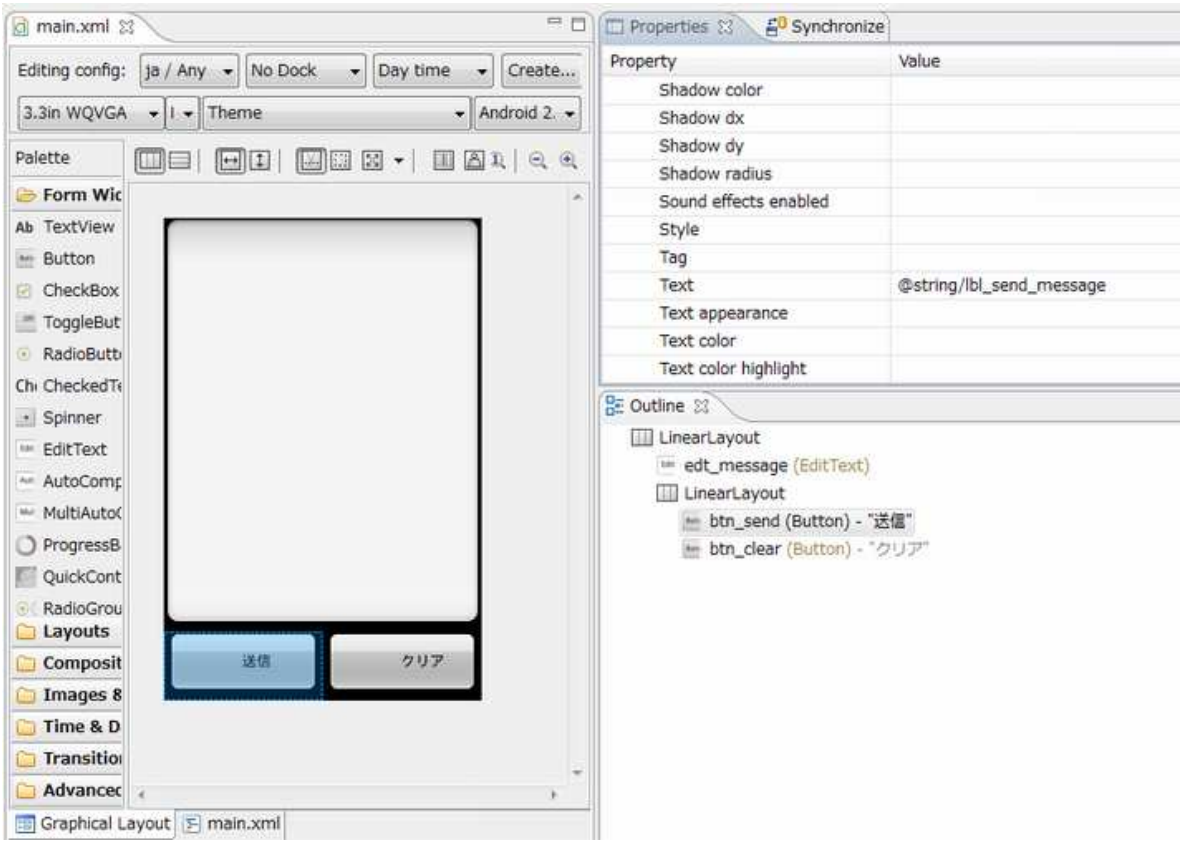
android.R クラス(パッケージ名に注意)を使うと、SDK がもともともっているリソースを利用することができます。

1.2.2 main.xml の編集

では、実際に main.xml を編集してみます。



/res/layout/main.xml が、XML 画面定義の実体です。

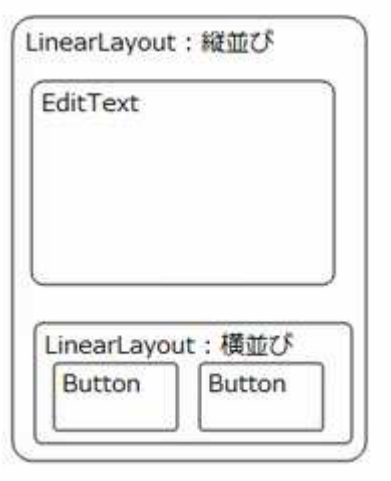


ダブルクリックすると、デザイナが起動します。下のタブで main.xml を選択すると、XML を直接編

集できます。コンポーネントを貼り付けて、プロパティを設定して上図のような画面を作ってみてください。 (もしくはサンプルソースの XML をコピーしてみてください)

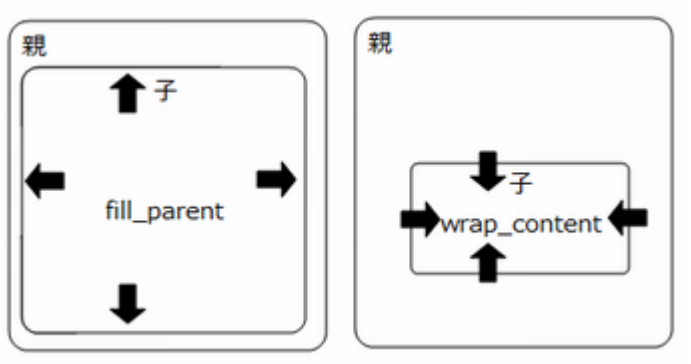
デザイナは使いやすいとは言いにくいですが、それでもあるとだいぶ楽です。

プロパティビューと、アウトラインビューを表示させておくと、コンポーネントの構造もわかりやすく、便利。上図の画面レイアウトの構造は、まさにアウトラインビューにあるとおりです。(あたりまえですが・・・)



1.2.3 レイアウトとプロパティ

LienarLayout は、コンポーネントを順番に並べるレイアウトで、縦向きに並べるか、横向きに並べるかなど指定できます。



特徴的なプロパティとして、コンポーネントには、layout_width や layout_height というプロ

パティがあるんですが、fill_parent を指定すると、親要素いっぱいに、wrap_content を指定すると、内容にフィットしたサイズにレイアウトされます。

サイズ関係のプロパティですが、Android では、いろいろな単位が出てきます。

http://y-anz-m.blogspot.com/2010/05/androiddimension.html

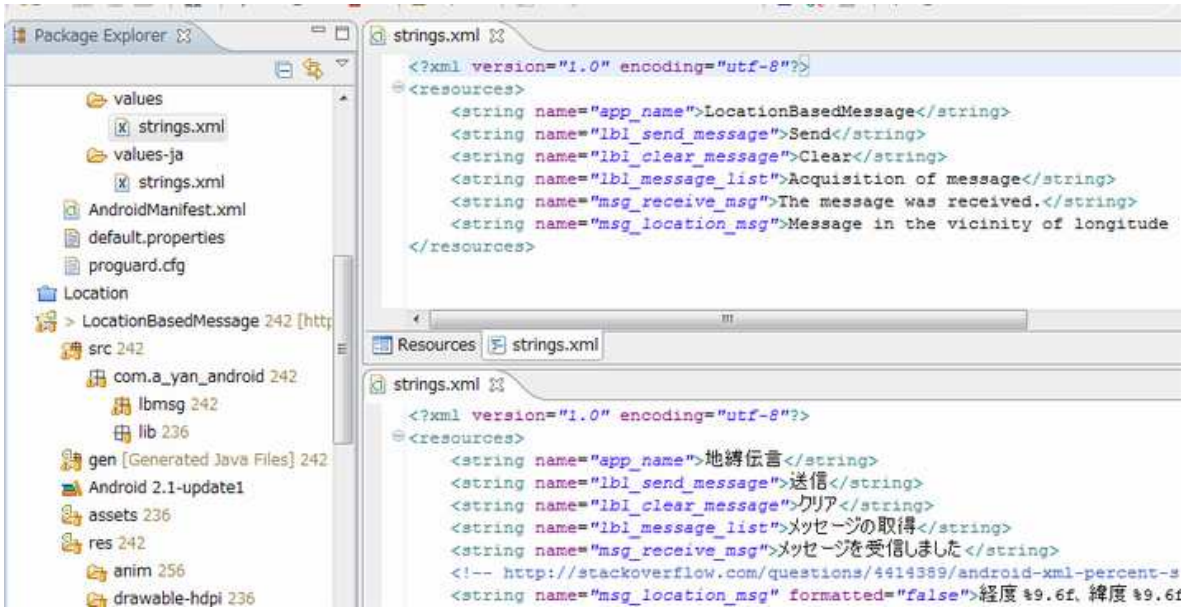
の辺りを参考にしてください。一ついえるのは、ピクセルは、PC と違って、モバイルでは機種によって実際に表示されるサイズがかなり変わってくるようなのでその他の論理的なサイズを使った方がいいでしょう。

1.2.4 文字列リソースの国際化

例えば、ボタンのテキストプロパティは、以下の様にしています。

```
android:text="@string/lbl_send_message"
```

このように、文字列をハードコーディングするのではなく、文字列リソースを参照するという形にすることで、アプリケーションの多言語対応など容易にできるようになります。



/res/values/string.xml がプロジェクト作成時から存在しますが、values-ja を作成し、その中に string.xml を作っておくと、端末のロケール切り替えにより、自動的に適切なリソースファイルが使われるようになります。また app_name には、アプリケーション名を設定しておきます。ちな

みに、-ja の部分は ISO 639-1 準拠らしいです。string.xml もサンプルソースからコピーしておいてください。

1.2.5 ID リソース

ボタンコンポーネントの、ID は、以下のような記述になっています。

```
android:id="@+id/btn_send"
```

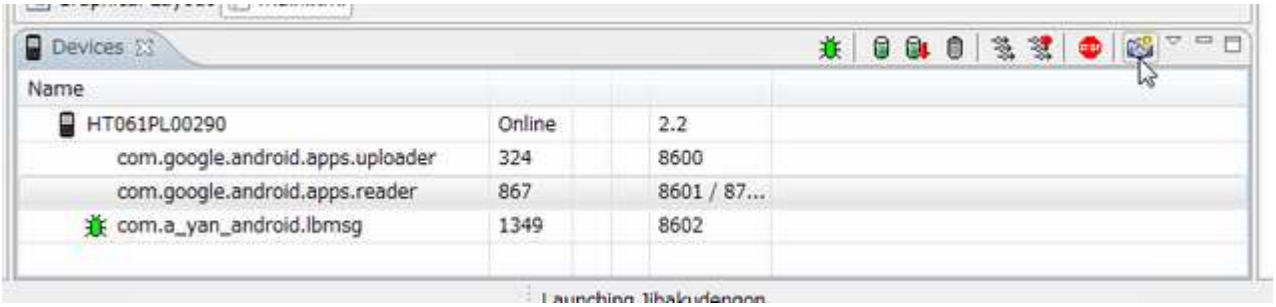
@+ という記述は、「当該リソースがない場合には新規で作成する」的な意味だったかと思います。後ほど、ソースコードから、コンポーネントを参照するために必要になります。

1.2.6 動かしてみよう

一度動かしてみよう。

Eclipse の Window – Show View – Other から、Android – Devices ビューを起動すると、デバイスの状態を確認したり、画面のキャプチャを取ることができるようになる。

Android 端末は基本的にアプリケーションから画面のキャプチャを取ることができないので、開発以外でも、意外と重宝する。



1.3 Activity の実装

画面の定義もできたので実装をしていこう。

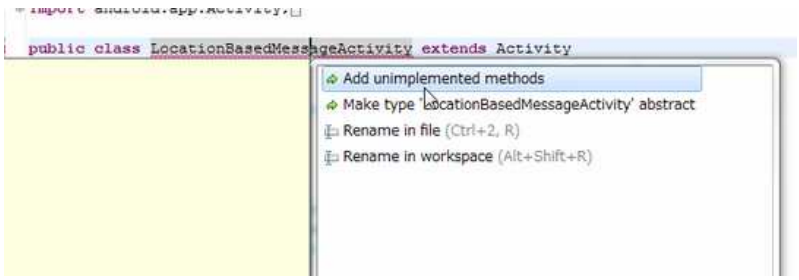
```
public class LocationBasedMessageActivity extends Activity
                                    implements View.OnClickListener {

    private EditText edtMessage;
    private Button   btnSend;
    private Button   btnClear;
```

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

        edtMessage = (EditText) findViewById(R.id.edt_message);
        btnSend    = (Button)   findViewById(R.id.btn_send);
        btnClear   = (Button)   findViewById(R.id.btn_clear);

        btnSend.setOnClickListener(this);
        btnClear.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
        case R.id.btn_send:

            // TODO
            break;

        case R.id.btn_clear:
            edtMessage.setText("");
            break;

        default:
            break;
        }
    }

}
```

上から順に、

1.3.1 View.OnClickListener を Activity に実装して、ボタン押下イベントハンドラを記述

Eclipse では、エラーがある場合、Ctrl＋1 にて、解決策を提示してくれる。Activity に対して、implement View.OnclickListener とタイプすると、Activity は 実装すべきメソッド onClick を実装していないので、不完全な状態として、クラス名に波線が入る。ここにカーソルを当てて、Ctrl + 1 とすると、



と、修正候補がでるので、ここでは、Add unimplemented method を選択し、未実装メソッドを自動で記述させる。

1.3.2 onCreate() で、コンポーネントの参照をメンバー変数に保持しておく

先ほど、XML でコンポーネントに付与したリソース ID を用いて、

findViewByID(リソース ID)

を行うことで、XML で定義したコンポーネントのインスタンスを取得し、操作が可能になる。

この手のことは、基本的に onCreate()で行う。Activity のライフサイクルは、また癖があるので、一度しっかり確認しておいたほうがよい。

http://developer.android.com/reference/android/app/Activity.html

1.3.3 ボタンのイベントハンドラに処理を記述

クリアボタンの押下にて、エディットボックスの内容をクリアする処理を記述。

1.3.4 メニューの作成

Activity にオプションメニュー(メニューボタン押下で表示される)を追加しておきます。また、イベントハンドラを作成しておきます。

```
private static final int MENU_MSG_LIST = Menu.FIRST;
private ISeekMessageService seekMessageService;
private AtomicBoolean isBinded = new AtomicBoolean();

 略

@Override
public boolean onCreateOptionsMenu(Menu menu) {
  menu.add(Menu.NONE,MENU_MSG_LIST, Menu.NONE, R.string.lbl_message_list)
    .setIcon(android.R.drawable.ic_menu_mylocation);
  return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
  switch(item.getItemId()) {
  case MENU_MSG_LIST:
    // サービスを作成したら非コメント化
```

```
      // if (isBinded.get() && seekMessageService != null) {
      //   try {
      //     seekMessageService.refleshMessages();
      //   } catch (RemoteException e) {
      //     // TODO Auto-generated catch block
      //     e.printStackTrace();
      //   }
      // }
      break;
    default:
      return false;
    }
    return true;
  }
```

ここまでで、画面作成の大枠は、理解できたのではないでしょうか(きっと・・・)。

2.位置情報を利用して WEB アプリと連携

位置情報や各種センサーを利用できるのが、モバイルアプリの利点であり、位置情報を活用するという視点が新しいサービスを着想するヒントとなり得ると思います。ここでは、位置情報を WEB アプリと連携させてみたいと思います。

2.1 位置情報の利用

2.1.1 端末の位置を取得

位置情報を使うには、LocationManager を利用します。Android では、何か機能を利用するときに、

なんちゃら Manager を宣言し、システムレベルのサービスを利用することが多いですが、ここでは、

位置情報サービスを利用します。

```
private LocationManager locationManager;

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

    locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);

    Location loc =
        locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
    Log.i("MyApp",
          String.format("lat:%f,
              lon:%f",loc.getLatitude(),loc.getLongitude()));

    edtMessage = (EditText) findViewById(R.id.edt_message);
    btnSend    = (Button)   findViewById(R.id.btn_send);
    btnClear   = (Button)   findViewById(R.id.btn_clear);

    btnSend.setOnClickListener(this);
    btnClear.setOnClickListener(this);
}
```

では、LocationManager を宣言して、Activity の onCreate で、位置情報サービスを取得、緯度

経度をログにはき出してみましょう。

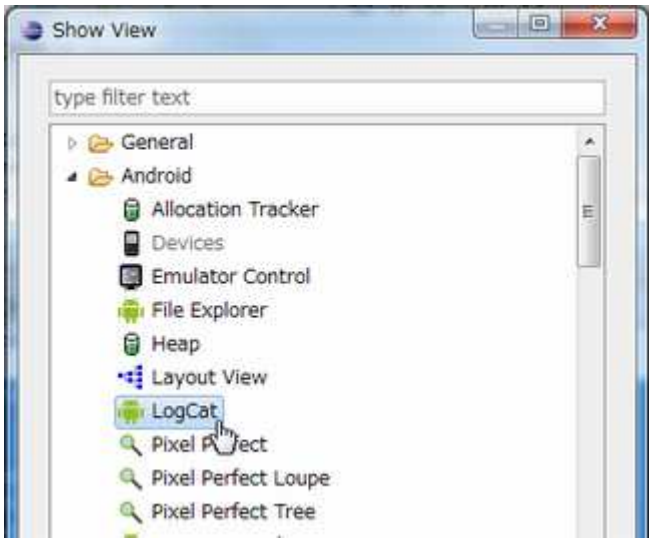
LocationManager.getLastKnownLocation で最後に取得した

2.1.2 ログに書き出す

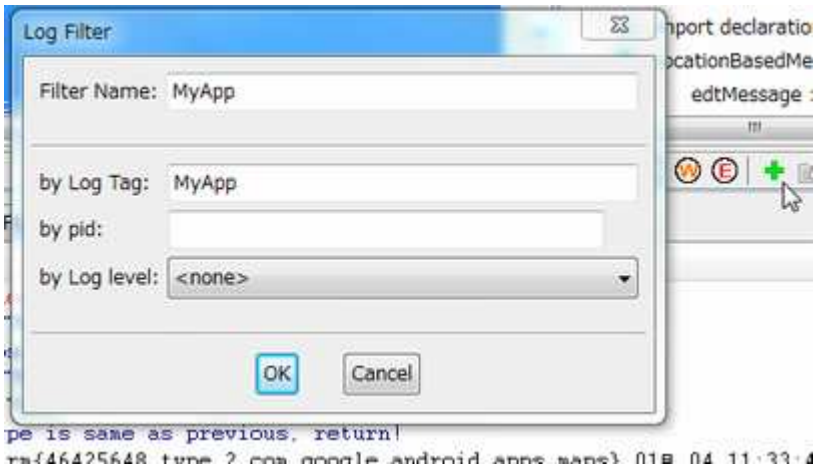
デバッグ情報などをログに書き出すには、android.util.Log クラスを使います。

Eclipse から、ログを確認するには、Window – Show View – Other から、LogCat を表示しま

す。



LogCat では、端末のログを全部拾ってしまうので、タグをつけることで、フィルターすることができます。LogCat の右上の+ボタンを押すとフィルターを追加できます。

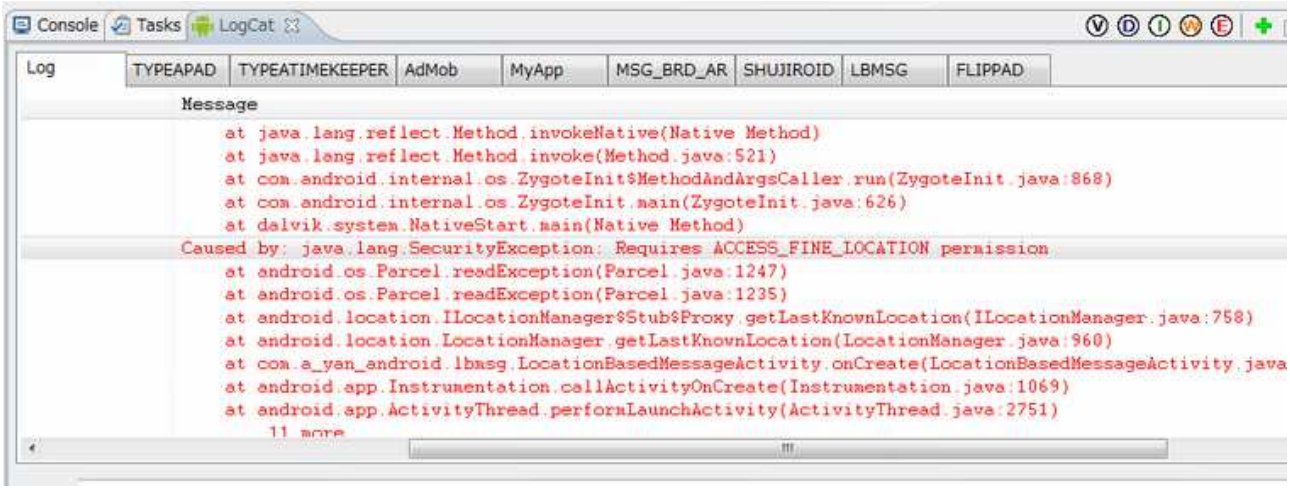


2.1.3 一旦実行

ここで、一旦実行してみましょう。

きちんと(?)エラーになったでしょうか？

LogCat をざっとさかのぼって見てみてください。赤字で例外が表示されているので、わかると思いますが、SecurityException が発生しています。



自分の場合、コーディングしてコードはどう見直してもおかしくないはずなのに、エラーになってしまうときなど、結構悩んだ末に、実は適切なセキュリティが設定してないことが原因だったということがままあります。システムサービスを利用する場合、確実にパーミッションの設定が必要ですので、適切に設定する必要があります。

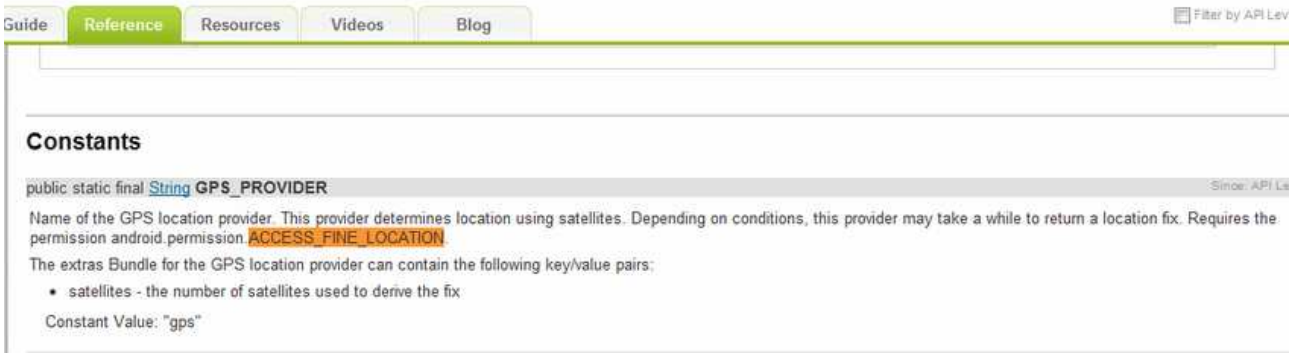
2.1.4 パーミッションの設定

では、パーミッションの設定をしてみます。

上記エラーログでは、ACCESS_FINE_LOCATION パーミッションが必要というエラーになってます。

使用するサービスによって、必要なパーミッションは、基本的に SDK の API ドキュメントに書かれています。

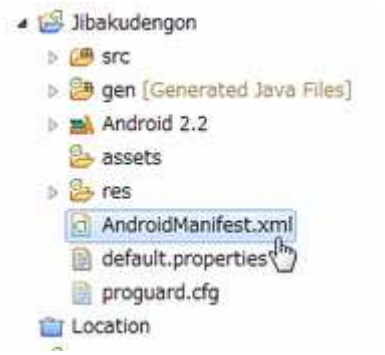
http://developer.android.com/reference/android/location/LocationManager.html



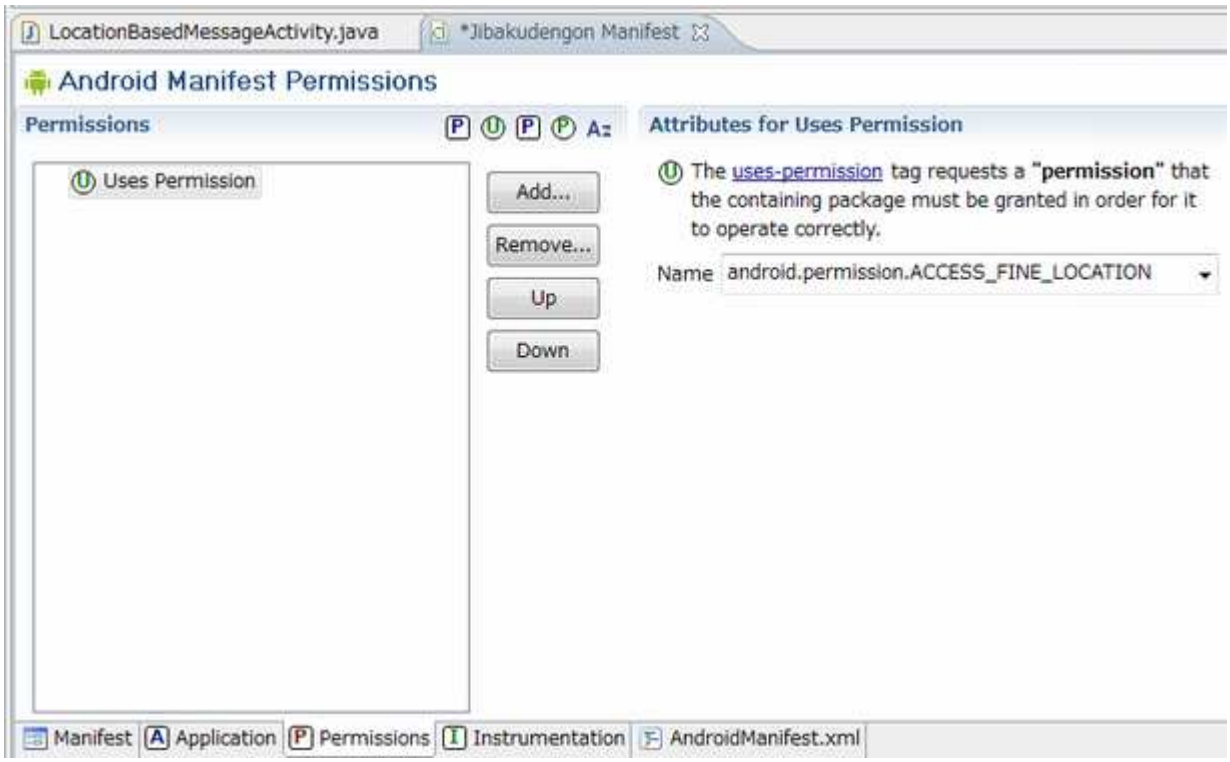
位置情報を利用するには、GPS を利用する GPS プロバイダーと、基地局や WiFi のアクセスポイントから位置を割り出す、ネットワークプロバイダーがあるのですが、GPS プロバイダーを使う場合、ACCESS_FINE_LOCATION パーミッションが必要という記述があります。

パーミッションの設定をしてみましょう。

AndroidManifest.xml を開きます。



Permissions タブから、Add を選択すると、エレメントの種類を選ぶことができるので、Uses Permission を選択します。すると、右側にパーミッションを選択するプルダウンがでてきますので、適切なものを選択して保存します。



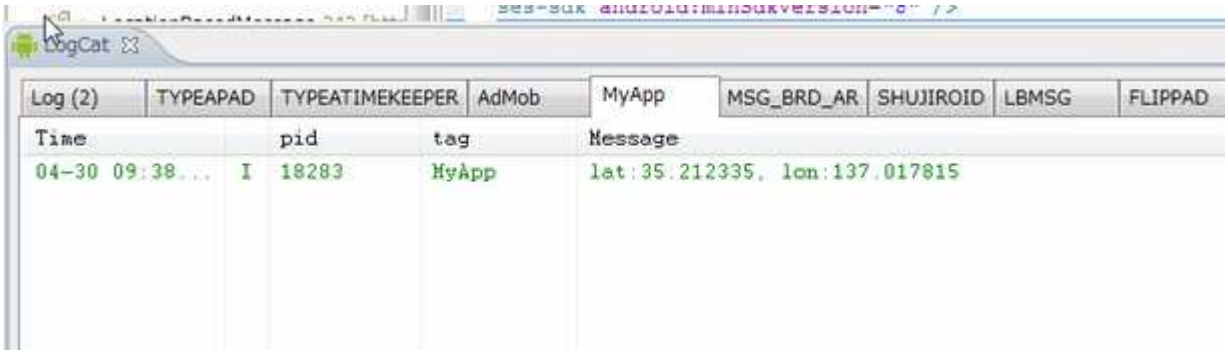
これで、OK のはずです。

```
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"></uses-permission>
```

上記が、AndroidManifest.xml に追記されます。

再度実行してみましょう。

LogCat の “MyApp” タグに緯度、経度は出力されたでしょうか？



2.2　WEB アプリを利用

Android アプリから WEB アプリを使うことができると、アイディアの幅が大きくひろがるのではない

でしょうか。今では Google App Engine のような PaaS サービスが無料や廉価で利用できますし、

こういうことをやるにはいい時代だと思います。

2.2.1 HTTP POST リクエストを投げる

Android から、HTTP リクエストを送って、結果を得る流れは、大きく以下の様な感じになります。

```
HttpPost post
   =  new HttpPost("http://typea-android-apps.appspot.com/lbmsg/insert");

List<NameValuePair> parms = new ArrayList<NameValuePair>();
parms.add(new BasicNameValuePair("lat", String.valueOf(loc.getLatitude())));
parms.add(new BasicNameValuePair("lon", String.valueOf(loc.getLongitude())));
parms.add(new BasicNameValuePair("message", msg));
post.setEntity(new UrlEncodedFormEntity(parms, HTTP.UTF_8));

DefaultHttpClient httpClient = new DefaultHttpClient();
HttpResponse response = httpClient.execute(post);

InputStream in = response.getEntity().getContent();
BufferedReader reader = new BufferedReader(new InputStreamReader(in));
String l = null;
while((l = reader.readLine()) != null) {
   buf.append(l + "¥r¥n");
}
if (response.getStatusLine().getStatusCode() != HttpStatus.SC_OK) {
   Log.e(LocationBasedMessageApplication.TAG, buf.toString());
}
```

POST リクエストを生成して、パラメータをセットして、リクエストを送信して、レスポンスを受け取るかたちになります。日本語をリクエストパラメータに使用する場合、UrlEncode を適切に指定する必要があります。

2.2.2 Google App Engine (Python) の内容を確認

今回、緯度、経度とメッセージを登録する機能と、緯度、経度の範囲を指定して、その範囲にあるメッセージを JSON として取得するサービスを作成しています。

これだけの機能なら、たかだか数百行のコードでかけてしまうので、Python(および GAE) の生産性は高いなぁと思いつつ、動的な部分でコード補完されなかったり、文法はやはり Java の方が単純なので、比較すると、Python の場合、久々に触ったときに感が戻るまでにちょっと時間がかかるなぁなどとも思います。今回は関係ないですが、今後の参考に、おもな部分(ほとんどですが・・・)のソースを掲げ

ておきます。

typea_android_apps.py

```
from google.appengine.ext import webapp
from google.appengine.ext.webapp.util import run_wsgi_app

from lbmsg import location_based_messge_req_handler as lbmsg_hander

class MainPage(webapp.RequestHandler):
    def get(self):
        self.response.headers['Content-Type'] = 'text/plain'
        self.response.out.write('Hello, TYPEA.INFO APPS')

application = webapp.WSGIApplication([
              ('/', MainPage),
              ('/lbmsg/',        lbmsg_hander.AuthCheck),
              ('/lbmsg/insert', lbmsg_hander.InsertLocalBasedMessage),
              ('/lbmsg/list',    lbmsg_hander.LocalBasedMessageList),
              ], debug=True)
def main():
    run_wsgi_app(application)

if __name__ == "__main__":
    main()
```

local_based_message.py

```
from google.appengine.ext import db

class LocalBasedMessage(db.Model):
    owner = db.UserProperty(auto_current_user_add=True)
    geo_pt = db.GeoPtProperty()
    message = db.TextProperty()
    create_date = db.DateTimeProperty()

    def to_dict(self):
        return {'key':         str(self.key()),
                'owner':       str(self.owner),
                'lat':         str(self.geo_pt.lat),
                'lon':         str(self.geo_pt.lon),
                'message':     self.message.encode('utf-8'),
                'create_date':  str(self.create_date),
                }
```

location_based_message_req_handler.py

```
from google.appengine.api import users
from google.appengine.ext import webapp
from google.appengine.ext import db
import json

from datetime import datetime

from local_based_message import LocalBasedMessage

MAX_MESSAGES = 100

class AuthCheck(webapp.RequestHandler):
```

```
    def get(self):
        self.post()

    def post(self):
        user = users.get_current_user()
        if user:
            self.response.headers['Content-Type'] = 'text/html'
            ret = '<html><body><a href="%s">logout</a></body></html>' %
(users.create_logout_url(self.request.uri))
            return self.response.out.write(ret)
        else:
            #Unauthorized
            #return self.response.set_status(401)
            self.redirect(users.create_login_url(self.request.uri))

class InsertLocalBasedMessage(webapp.RequestHandler):
    def get(self):
        self.post()

    def post(self):
        user = users.get_current_user()
        if user:
            lat = self.request.get('lat')
            lon = self.request.get('lon')
            message = self.request.get('message')

            lbmsg = LocalBasedMessage(geo_pt=db.GeoPt(lat, lon),
                                message=message,
                                create_date=datetime.now()
                                )
            lbmsg.put()
            return self.response.out.write(json.write(lbmsg.to_dict()))
        else:
            #Unauthorized
            #return self.response.set_status(401)
            self.redirect(users.create_login_url(self.request.uri))

class LocalBasedMessageList(webapp.RequestHandler):
    def get(self):
        self.post()

    def post(self):
        user = users.get_current_user()
        if user:
            from_geo_pt=db.GeoPt(self.request.get('from_lat'),
                                 self.request.get('from_lon'))
            to_geo_pt=db.GeoPt(  self.request.get('to_lat'),
                                 self.request.get('to_lon'))

            q = LocalBasedMessage.gql("WHERE geo_pt >=:1 AND geo_pt <=:2"
                                 ,from_geo_pt
                                 ,to_geo_pt)

            messages = q.fetch(MAX_MESSAGES)

            ret = []
            for message in messages:
                ret.append(message.to_dict())
            return self.response.out.write(json.write(ret))

        else:
```

```
            #Unauthorized
            #return self.response.set_status(401)
            self.redirect(users.create_login_url(self.request.uri))
```

2.2.3 メッセージを送信する

位置情報も取得でき、WEB へのリクエストの送信もできるようになったので、メッセージを送信してみようと思います。ただ、上記 GAE で、確認したように今回、Google のアカウントを認証に使用しています。Android 端末は Google のアカウントと親和性が高いので、この仕組みを押さえておくと簡単に認証が必要なアプリを開発できるだろうという目論見です。Google アカウントを使用するには、先ほどの POST リクエストだけではなく、認証部分処理を作り込む必要があります。

2.2.3.1 Google アカウントを利用するクラス

発展途上ですが、Google アカウントの認証部分をまとめたクラスを作成して、利用しますので少し長いですが、ここからコピーするか、サンプルからファイルをとりこんでください。以下のパーミッションを追加する必要があります。GET_ACCOUNTS、MANAGE_ACCOUNTS、USE_CREDENTIALS

```
package com.a_yan_android.lib.google;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.params.ClientPNames;
import org.apache.http.cookie.Cookie;
import org.apache.http.impl.client.DefaultHttpClient;

import android.accounts.Account;
import android.accounts.AccountManager;
import android.accounts.AccountManagerCallback;
import android.accounts.AccountManagerFuture;
import android.accounts.AuthenticatorException;
import android.accounts.OperationCanceledException;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
```

```
import com.a_yan_android.lbmsg.LocationBasedMessageApplication;

/**
 * @author piroto
 *
 * @see http://typea.info/blg/glob/2010/08/android-gae-windows7.html
 * @see http://typea.info/blg/glob/2010/08/android-google-gae.html
 * @see http://typea.info/blg/glob/2010/09/android-gae.html
 * @see http://typea.info/blg/glob/2011/04/android-googlegaeserver-error.html
 */
public class GoogleServiceUtil {

  private GoogleServiceUtil() {}

  public enum GOOGLE_ACCOUNT_TYPE {
    GAE    { public String toString() {return "ah";    } },
    CALENDAR  { public String toString() {return "cl";    } },
    GMAIL    { public String toString() {return "mail";    } },
    READER    { public String toString() {return "reader";  } },
    TALK    { public String toString() {return "talk";    } },
    YOUTUBE   { public String toString() {return "youtube";   } },
  }

  /**
   * GAE での認証用URIを生成
   * @param appPath
   * @param authToken
   * @return
   */
  public static String defaultGAELoginUrl(String hostname, String appPath, String authToken)
{
    return "http://" + hostname + "/_ah/login?continue=" + appPath + "&auth=" + authToken;
  }

  /**
   * Google のサービスを実行するクラス
   *
   * @author piroto
   */
  public static class Executer {
    public static final String GOOGLE_ACCOUNT_TYPE = "com.google";

    private Context context;
    private AccountManager accountManager;

    /**
     * @param context
     */
    public Executer(Context context) {
      this.context = context;
    }

    /**
     * @return
     */
    private AccountManager getAccountManager() {
      if (accountManager == null) {
        accountManager = AccountManager.get(this.context);
      }
      return accountManager;
    }
```

```
    /**
     * Google アカウントを取得
     * @return
     */
    public Account[] getGoogleAccounts() {
      return getAccountManager().getAccountsByType(GOOGLE_ACCOUNT_TYPE);
    }

    /**
     * 認証を行った後、Google サービスを実行させる
     * @param account Google アカウント
     * @param type アカウントタイプ
     * @param callback 認証の要求が完了したときに呼び出されるコールバック
     */
    public void execute(Account account, GOOGLE_ACCOUNT_TYPE type,
AbstractGoogleServiceCallback callback) {

      getAccountManager().getAuthToken(
           account,
           type.toString(),
           false,
           callback,
           null   // nullの場合、callback はメインスレッド
           );
    }

    /**
     * アカウントを削除
     * @param account
     * @param type
     */
    public void removeAccount(Account account) {
      getAccountManager().removeAccount(account, null, null);
    }
  }

  /**
   * Google サービスの認証部を実装したコールバック
   * @author piroto
   *
   */
  public static abstract class AbstractGoogleServiceCallback implements
AccountManagerCallback<Bundle> {
    private Context context;

    /**
     * @param context
     */
    public AbstractGoogleServiceCallback(Context context) {
      this.context = context;
    }

    /**
     * @return
     */
    public Context getContext() {
      return this.context;
    }

    /**
```

```
     * @param bundle
     * @param authToken
     */
    public void removeAuthTokenCache(Bundle bundle, String authToken) {
      String accountType = bundle.getString(AccountManager.KEY_ACCOUNT_TYPE);
      AccountManager manager = AccountManager.get(getContext());
      manager.invalidateAuthToken(accountType, authToken);
    }

    /**
     * ACSID を取得する
     * @param bundle
     * @return
     */
    private String getACSID(Bundle bundle){
      final int RETRY_MAX = 3;
      boolean isValidToken = false;

      int  retry = 0;
      String acsid = null;
      try {
        DefaultHttpClient httpClient = null;
        while (!isValidToken) {

          String authToken = bundle.getString(AccountManager.KEY_AUTHTOKEN);
          httpClient = new DefaultHttpClient();
          httpClient.getParams().setBooleanParameter(ClientPNames.HANDLE_REDIRECTS,
false);

          String uri = getAuthenticateUri(authToken);
          HttpGet httpAuthGetRequest = new HttpGet(uri);
          HttpResponse httpAuthResponse = httpClient.execute(httpAuthGetRequest);
          int status = httpAuthResponse.getStatusLine().getStatusCode();

          // 認証が成功した場合、リダイレクトのために302 が返される
          if (status == HttpStatus.SC_MOVED_TEMPORARILY) {
            isValidToken = true;
          } else {
            // 認証エラーの場合、ログにレスポンスの内容を出力
            try {
              StringBuilder buf = new StringBuilder();
              buf.append(String.format("status:%d¥n",status));
              InputStream in = httpAuthResponse.getEntity().getContent();
              BufferedReader reader = new BufferedReader(new InputStreamReader(in));
              String l = null;
              while((l = reader.readLine()) != null) {
                buf.append(l + "¥n");
              }
              Log.e(”MyApp”,buf.toString());
            } catch(Exception e) {
            }
            // 認証トークンキャッシュの削除
            //   期限切れ、もしくは、認証リクエストが無効になるような、認証トークンが見つかった場合、
            //   キャッシュのクリアを行う
            removeAuthTokenCache(bundle, authToken);
          }

          retry++;
          if (retry > RETRY_MAX) {
            break;
          }
```

```
    }
    if (isValidToken) {
      // 認証エラーでなければ、Cookie から ACSIDを取得する
      for (Cookie cookie : httpClient.getCookieStore().getCookies()) {
        if ("SACSID".equals(cookie.getName()) ||
          "ACSID".equals(cookie.getName())) {
          acsid = cookie.getName() + "=" + cookie.getValue();
        }
      }
    }
  } catch (ClientProtocolException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
  } catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
  }
  return acsid;
}

@Override
public void run(AccountManagerFuture<Bundle> future) {

  HttpResponse httpBodyResponse = null;
  try {
    Bundle bundle = future.getResult();
    Intent intent = (Intent)bundle.get(AccountManager.KEY_INTENT);
    if (intent != null) {
      // 認証されていない場合、認証画面を起動
      this.context.startActivity(intent);
    } else {

      DefaultHttpClient httpClient = new DefaultHttpClient();
      String acsid = getACSID(bundle);
      // 認証が正常に行われ、ACSIDが取得できたら、サービスのリクエストをPOSTし、
      // レスポンスを取得
      if (acsid != null) {
        HttpPost httpPost = request();
        httpPost.setHeader("Cookie", acsid);
        httpBodyResponse = httpClient.execute(httpPost);
      }
    }

  } catch (OperationCanceledException e) {
    // TODO
    e.printStackTrace();
  } catch (AuthenticatorException e) {
    // TODO
    e.printStackTrace();
  } catch (IOException e) {
    // TODO
    e.printStackTrace();
  } finally {
    // コールバックする。認証などでエラーが発生している場合、レスポンスは null
    callback(httpBodyResponse);
  }
}

/**
 * 認証を行うURLを指定
 * たとえばGAEであれば、以下のようなURLを返す
```

```
     * <pre>"http://typea-msg-brd.appspot.com/_ah/login?continue=/&auth=" +
authToken;</pre>
     * @param authToken
     * @return
     */
    public abstract String getAuthenticateUri(String authToken);

    /**
     * 認証に成功した後、実際に行いたいリクエストを実装する
     * @return
     */
    public abstract HttpPost request();

    /**
     * request() レスポンスを受け取る
     * @param response
     */
    public abstract void callback(HttpResponse response);
  }
}
```

2.2.3.2 メッセージを送信するメソッドの実装

Activity にメッセージを送信する処理を実装します。

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

  private EditText edtMessage;
  private Button   btnSend;
  private Button   btnClear;

  private LocationManager locationManager;

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

    locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);

    edtMessage = (EditText) findViewById(R.id.edt_message);
    btnSend    = (Button)  findViewById(R.id.btn_send);
    btnClear   = (Button)  findViewById(R.id.btn_clear);

    btnSend.setOnClickListener(this);
    btnClear.setOnClickListener(this);
  }

  @Override
  public void onClick(View v) {
    switch (v.getId()) {
    case R.id.btn_send:
      sendLocationBasedMessage(edtMessage.getText().toString());
      break;
    case R.id.btn_clear:
      edtMessage.setText("");
      break;
```

```
    default:
      break;
    }
  }

  /**
   * もっとも最近取得した位置情報を元に、メッセージをGAEアプリに送信、登録
   * @param message
   */
  private void sendLocationBasedMessage(String message) {
    final String msg = message;
    final Location loc =
locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);

    GoogleServiceUtil.Executer exec = new GoogleServiceUtil.Executer(this);
    Account[] accounts = exec.getGoogleAccounts();
    if (accounts == null) {
      // TODO
      return;
    }

    exec.execute(accounts[0],
       GOOGLE_ACCOUNT_TYPE.GAE,
       new GoogleServiceUtil.AbstractGoogleServiceCallback(this) {

      @Override
      public HttpPost request() {
        HttpPost post
          =   new HttpPost("http://typea-android-apps.appspot.com/lbmsg/insert");
        try {
          List<NameValuePair> parms = new ArrayList<NameValuePair>();
          parms.add(new BasicNameValuePair("lat", String.valueOf(loc.getLatitude())));
          parms.add(new BasicNameValuePair("lon", String.valueOf(loc.getLongitude())));
          parms.add(new BasicNameValuePair("message", msg));
          post.setEntity(new UrlEncodedFormEntity(parms, HTTP.UTF_8));
        } catch (UnsupportedEncodingException e) {
          // TODO Auto-generated catch block
          e.printStackTrace();
        }
        return post;
      }

      @Override
      public String getAuthenticateUri(String authToken) {
        return GoogleServiceUtil.defaultGAELoginUrl("typea-android-apps.appspot.com",
                           "/lbmsg",
                           authToken);
      }

      @Override
      public void callback(HttpResponse response) {
        if (response != null) {
          int status = response.getStatusLine().getStatusCode();

          StringBuilder buf = new StringBuilder();
          buf.append(String.format("status:%d", status));
          try {
            InputStream in = response.getEntity().getContent();
            BufferedReader reader = new BufferedReader(new InputStreamReader(in));
            String l = null;
            while((l = reader.readLine()) != null) {
```

```
            buf.append(l + "¥r¥n");
          }
          if (status != HttpStatus.SC_OK) {
            Log.e("MayApp", buf.toString());
          }

        } catch(Exception e) {
          e.printStackTrace();
        }
        (Toast.makeText(getContext(), buf.toString(), Toast.LENGTH_LONG)).show();
      }
    }
  });
 }
}
```

まず、sendLocationBasedMessage() メソッドを Activity 実装し、送信ボタンのクリックイベントにて呼び出します。

スタティックインナークラス

実際に呼び出す箇所は、先ほど作成した GoogleServiceUtil.Executer クラス()の execute() メソッドの引数に、無名クラスを渡すという仕組みになっています。GoogleServiceUtil.Executer のように、<クラス>.<クラス> という書き方はあまりなじみがないかもしれないですが、スタティックインナークラスといって、クラスの内部にスタティックにクラスを宣言します。通常のインナークラスでは、存在自体が外側のクラスに依存してしまいますが、スタティックであるため、外側のクラスで修飾した型を宣言することで、他のクラスと同じように利用することができます。基本的に類似するクラスをまとめるのは Java の場合パッケージですが、特定のクラスに結びついたクラスを使いたい場合、利用箇所が、その特定のクラスからだけの場合通常のインナークラスで良いですが、他のクラスからも利用する場合で、その関係を明示しておきたい場合など非常に便利です。この考え方は、インターフェースでも同様で、実は、View.OnClickListener は、インナーインターフェースとなっています(インターフェースは暗黙でスタティック)し、Android SDK にて多用されていますので、覚えておいた方が良いでしょう。

Toast

最後に、GAE からのレスポンスを、Toast という機能を利用して画面に表示します。

ちょっとしたメッセージをユーザーに表示するのに便利です。

2.2.4 さて実行して、メッセージを送信してみましょう

上図の様に結果が返ってきて、Toast に表示されたでしょうか？

されてないと思います。LogCat を見てみると、例外が出力されている筈です。

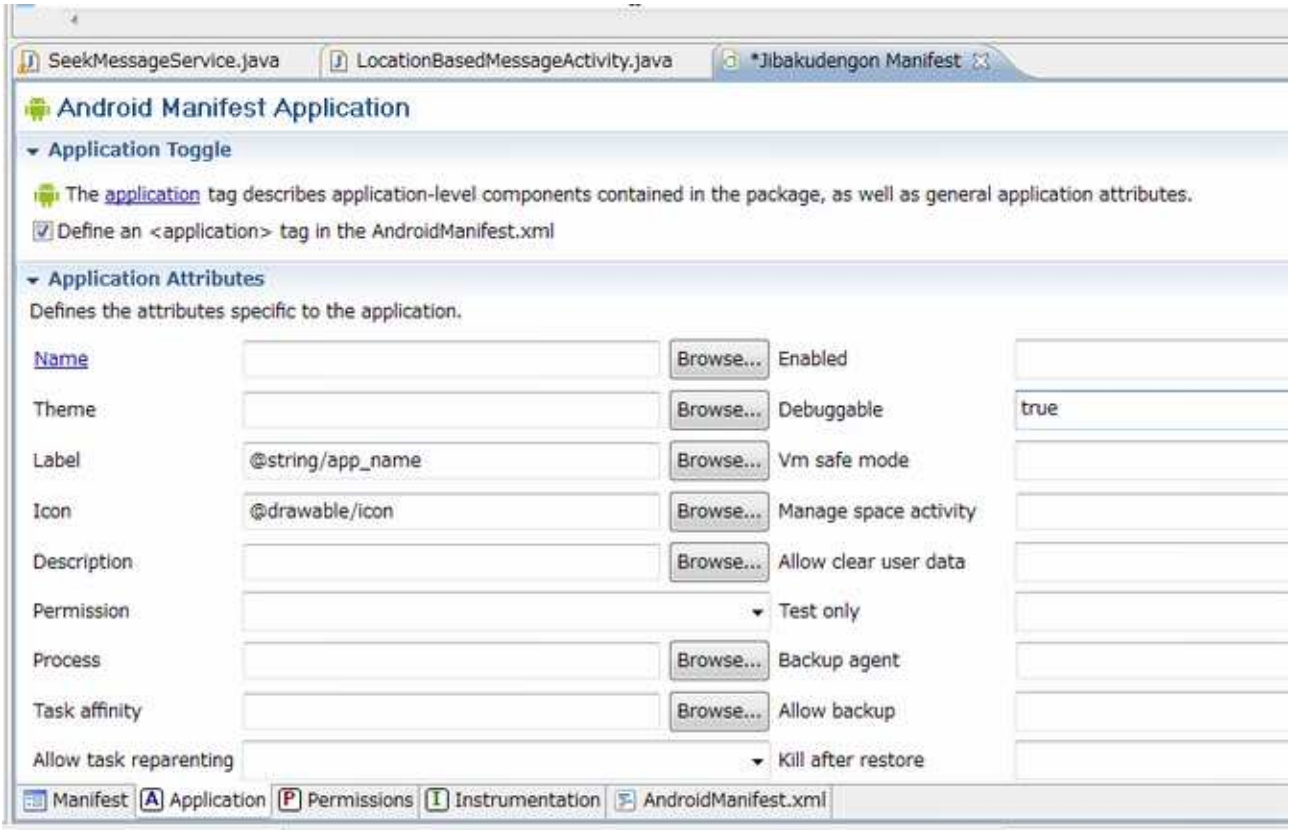
そうです。インターネットアクセスにもパーミッションが必要です。

```
<uses-permission android:name="android.permission.INTERNET"/>
```

設定して実行してみてください。

2.2.5ステップ実行してみましょう

デバッグのときに、ステップ実行したり、途中で止めて変数の中身を見たい場合、ブレークポイントを設定しますが、ただ、ブレークポイントを Eclipse 上から設定し、Debug – As で起動しても、ブレークポイントでは実行は止まってくれません。



AndroidManifest の Application タブ、Debuggable に true を設定し保存、デバッグを開始する必要があります。

Debuggable を true として実行すると、アプリ起動時にデバッガと接続するため、若干起動が遅くなるので、必要なときだけ true にするのが吉ではないかと思います。

3.サービスをつくる

サービスを作成し、バックグラウンドでうごかして範囲内のメッセージを取得し、結果を通知領域に表示させてみます。

5.1 サービスを作成

サービスには 2 種類あって、一つはクライアントとサービスが同じプロセスで動作するもの、もう一つは別プロセスで動作するものです。別プロセスで動作させてプロセス間で通信を行うためには、AIDL(Android Interface Definition Language)という仕組みを使う必要があります。

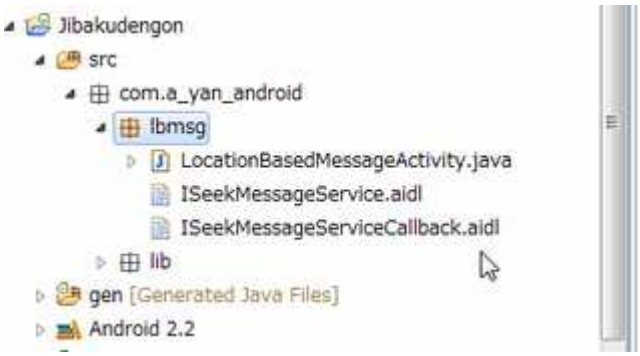
プロセス間通信とか、AIDL とかなにやらややこしそうですが、若干手順と書き方が異なるだけで、非常に簡単に作成することができます。

このアプリでは、端末の位置が変わったりした場合に、その付近にあるメッセージを取得し、通知を行うという別プロセスで動くサービスを作ってみます。

3.1.1 AIDL ファイルの作成

SeekMessageService というサービスを作成しようと思います。そのサービスを利用するためのインターフェースとして、ISeekMessageService.aidl というインターフェースファイルを作成します。また、サービスからコールバックするインターフェースとして、ISeekMessageServiceCallback.aidl というインターフェースファイルも作成します。

作成したい場所(パッケージ)のコンテキストメニューから、New – File で、上記 2 つの aidl ファイル名を 2 つ作成し、以下のコードをタイプしてください。



ISeekMessageService.aidl

```
package com.a_yan_android.lbmsg;
import com.a_yan_android.lbmsg.ISeekMessageServiceCallback;
```

```
oneway interface ISeekMessageService {
   void registerCallback(ISeekMessageServiceCallback callback);
   void unregisterCallback(ISeekMessageServiceCallback callback);
   void refleshMessages();
}
```

ISeekMessageServiceCallback.aidl

```
package com.a_yan_android.lbmsg;

oneway interface ISeekMessageServiceCallback {
}
```

registerCallback()、unregisterCallback()はコールバックを登録するためのメソッド、refleshMessages() は、メッセージを取得し直すためのメソッドです。

ここまでタイプして、保存を行うと、各 AIDL ファイルから、自動的に Java クラスが生成されて利用可能になります。(gen 配下に自動生成されたクラスがあります。R クラスもここにありました)

3.1.2 サービスの作成

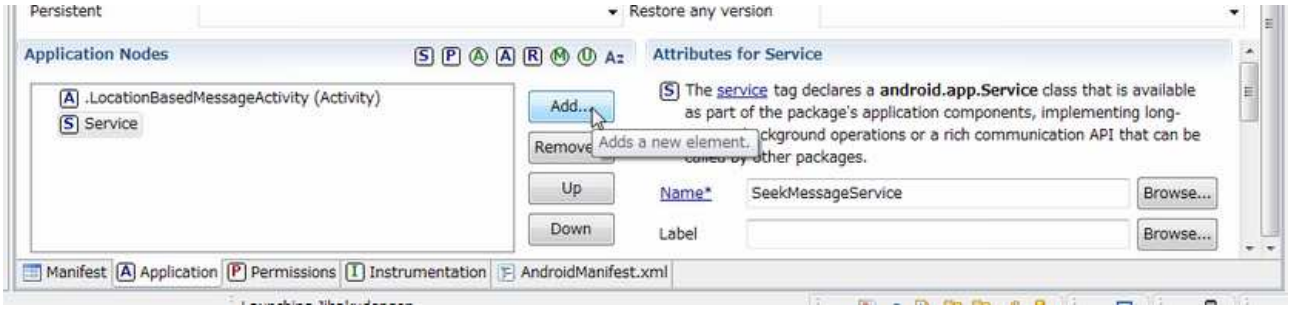
では、これらを利用するサービスを作成します。パッケージのコンテキストメニューから、New – Class として、SeekMessageService を作成してください。継承元として、android.app.Service を指定します。また、端末の位置が変更されたことを受け取るリスナーインターフェース LocationListener を実装しておきます。

```
package com.a_yan_android.lbmsg;

public class SeekMessageService extends Service implements LocationListener {

  private final RemoteCallbackList<ISeekMessageServiceCallback> callbacks
    = new RemoteCallbackList<ISeekMessageServiceCallback>();

  @Override
  public void onCreate() {
    super.onCreate();
  }
```

```
  @Override
  public IBinder onBind(Intent intent) {
    return binder;
  }

  private ISeekMessageService.Stub binder = new ISeekMessageService.Stub() {
    @Override
    public void registerCallback(ISeekMessageServiceCallback callback)
        throws RemoteException {
      callbacks.register(callback);
    }

    @Override
    public void unregisterCallback(ISeekMessageServiceCallback callback)
        throws RemoteException {
      callbacks.unregister(callback);
    }

    @Override
    public void refleshMessages() throws RemoteException {
      Location location = null; //TODO
      SeekMessageService.this.getMessages(location);
    }
  };

  @Override
  public void onLocationChanged(Location location) {
    getMessages(location);
  }

  /**
   * @param location
   */
  public void getMessages(Location location) {
    //TODO
  }

  @Override
  public void onProviderDisabled(String provider) {
    // TODO Auto-generated method stub
  }

  @Override
  public void onProviderEnabled(String provider) {
    // TODO Auto-generated method stub
  }

  @Override
  public void onStatusChanged(String provider, int status, Bundle extras) {
    // TODO Auto-generated method stub
  }
}
```

Import 文は長くなるので掲げていませんが、Eclipse のエディタにて、Shift + Ctrl + o とすることで、ほとんどの場合、適切な inport 文が自動で作成されます。

また、Service は、Activity 同様、AndroidManifest に登録しておかないと使えません。



AndroidManifest の GUI エディタの Application タブから、Application Nodes の Add を選択し、Service を選択すると、右側の部分が表示されますので、Name を Brows して、サービスを設定、保存してください。

3.1.3 サービスに接続

では、アクティビティを作成したサービスに接続してみます。

```
public class LocationBasedMessageActivity extends Activity
                                    implements View.OnClickListener {
          略
  private ISeekMessageService seekMessageService;
  private AtomicBoolean isBinded = new AtomicBoolean();

  public void onCreate(Bundle savedInstanceState) {
          略
      bindLocationService();
  }

  private ISeekMessageServiceCallback callback = new ISeekMessageServiceCallback.Stub() {
  };

  private ServiceConnection conn = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
      seekMessageService = ISeekMessageService.Stub.asInterface(service);
      isBinded.set(true);

      try {
        seekMessageService.registerCallback(callback);
      } catch (RemoteException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
      }
    }

    @Override
    public void onServiceDisconnected(ComponentName name) {
      isBinded.set(false);
      seekMessageService = null;
    }
```

```
    };

    public void bindLocationService() {
      bindService(
          new Intent(this,SeekMessageService.class),
          conn,
          Context.BIND_AUTO_CREATE);
    }

    public void unBindLocationService() {
      if (isBinded.get()) {
        unbindService(conn);
      }
    }
          略
  }
```

conn メンバー変数では、ServiceConnection インターフェースを実装し、接続時にサービスを取得しコールバックを登録します。

callback メンバー変数では、今のところは、サービス側からのコールバックを作成していないので、空なのすが、必要に応じて、ISeekMessageServiceCallback.aidl に、コールバックメソッドのインターフェースを記述することによって、サービス側からコールバックすることができるようになります。

また、**Intent** によって Service を生成、バインドしています。Intent も Android では非常に重要な概念ですので、後述します。

5.2　位置情報の機能をサービスに追加

ここまででは、サービスを呼び出しても何も起こらないので、退屈ですが、もう少しコードを追加したいと思います。

3.2.1 位置情報のユーティリティクラスを作成

LocationUtil クラスを作成し、ある緯度、経度を基準に指定距離の開始位置、終了位置を取得する処理を実装してみます。これも以下からコピーするか、サンプルから取得してみてください。

```
  package com.a_yan_android.lib.location;
```

```
public class LocationUtil {
   public static final double WGS84_MAJOR_AXIS     = 6378137.0d;    // m
   public static final double WGS84_SEMI_MAJOR_AXIS = 6356752.3142d;  // m

  private LocationUtil() {
  }

  /**
   * @author piroto
   *
   */
  public static class GeoPt {
    private double lat;
    private double lon;
    public GeoPt(double lat, double lon) {
      setLat(lat);
      setLon(lon);
    }
    public double getLat() {
      return lat;
    }
    public void setLat(double lat) {
      if (lat < -90d || lat > 90d) {
        throw new IllegalArgumentException();
      }
      this.lat = lat;
    }
    public double getLon() {
      return lon;
    }
    public void setLon(double lon) {
      if (lon < -180d || lon > 180d ) {
        throw new IllegalArgumentException();
      }
      this.lon = lon;
    }
    @Override
    public String toString() {
      return String.format("%f,%f",getLat(),getLon());
    }
  }

  /**
   * 指定された緯度経度を中心に、一辺指定メートルのエリアの位置を返す
   * @param pt
   * @param marginMeter
   * @return
   */
  public static GeoPt[] getRect(GeoPt pt, double meter) {
    double halfSide = meter / 2d;
    double fromLat = getLatitudeAddedMeter(pt, -halfSide);
    double toLat   = getLatitudeAddedMeter(pt,  halfSide);
    double fromLon = getLongtudeAddedMeter(pt, -halfSide);
    double toLon   = getLongtudeAddedMeter(pt,  halfSide);

    GeoPt[] ret = new GeoPt[2];
    ret[0] = new GeoPt(fromLat, fromLon);
    ret[1] = new GeoPt(toLat,   toLon);

    return ret;
  }
```

```
  /**
   * 経度、緯度の書式
   * @param pt
   * @return
   */
  public static String formatGeoPt(double pt) {
    return String.format("%9.6f",pt);
  }

  /**
   * 指定された緯度に対して、指定メートル垂直移動した緯度を返す
   * @param lat
   * @param meter 正の値は北向き、負の値は南向き
   * @return
   */
  public static double getLatitudeAddedMeter(GeoPt pt, double meter) {
    double lat = pt.lat;
    double ret = lat +  ((meter / WGS84_MAJOR_AXIS) * 180d) / Math.PI;
    if (ret < -90d) { ret = -90d; }
    if (ret >  90d) { ret =  90d; }
    return ret;
  }

  /**
   * 指定された経度に対して、指定メートル水平移動した経度を返す
   * @param lat
   * @param lon 正の値は東向き、負の値は西向き
   * @param meter
   * @return
   */
  public static double getLongtudeAddedMeter(GeoPt pt, double meter) {
    double lat = pt.lat;
    double lon = pt.lon;
    double radius = (lat * Math.PI / 180d) * WGS84_MAJOR_AXIS; // 緯度の半径
    double ret = lon + ((meter / radius) * 180d / Math.PI);
    if (ret < -180d) { ret = -180d; }
    if (ret >  180d) { ret =  180d; }
    return ret;
  }
}
```

これは何？という場合、http://typea.info/blg/glob/2011/03/android-gps-1.html を見て

いただけると、参考になるかもです。

3.2.2 サービスにロジックを実装する

では、サービスにロジックを実装してみましょう。たぶん時間もないですし、ざっくりと目を通して、

サンプルからコピーしてみてください。

```
package com.a_yan_android.lbmsg;

import static com.a_yan_android.lib.location.LocationUtil.formatGeoPt;
```

```
public class SeekMessageService extends Service implements LocationListener {

  private final RemoteCallbackList<ISeekMessageServiceCallback> callbacks
    = new RemoteCallbackList<ISeekMessageServiceCallback>();

  private NotificationManager notifManager;
  private LocationManager locationManager;

  private GoogleServiceUtil.Executer exec;

  @Override
  public void onCreate() {
    super.onCreate();

    locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
    locationManager.requestLocationUpdates(
        LocationManager.GPS_PROVIDER,
        60000,   // 位置取得間隔の最小値(ミリ秒) 電源を節約するためのヒント
        50,     // 通知位置間隔の最小値(m)
        this);

    notifManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
    exec = new GoogleServiceUtil.Executer(this);
  }

  @Override
  public IBinder onBind(Intent intent) {
    return binder;
  }

  private ISeekMessageService.Stub binder = new ISeekMessageService.Stub() {
    @Override
    public void registerCallback(ISeekMessageServiceCallback callback)
        throws RemoteException {
      callbacks.register(callback);
    }

    @Override
    public void unregisterCallback(ISeekMessageServiceCallback callback)
         throws RemoteException {
      callbacks.unregister(callback);
    }

    @Override
    public void refleshMessages() throws RemoteException {
      Location location
          = SeekMessageService.this.locationManager
            .getLastKnownLocation(LocationManager.GPS_PROVIDER);
      SeekMessageService.this.getMessages(location);
    }
  };

  @Override
  public void onLocationChanged(Location location) {
    getMessages(location);
  }

  /**
   * @param location
   */
  public void getMessages(Location location) {
```

```
final Location loc = location;

for (Account a : exec.getGoogleAccounts()) {

  exec.execute(a, GOOGLE_ACCOUNT_TYPE.GAE,
      new GoogleServiceUtil.AbstractGoogleServiceCallback(this) {

    @Override
    public String getAuthenticateUri(String authToken) {
      return GoogleServiceUtil.defaultGAELoginUrl("typea-android-apps.appspot.com",
          "/lbmsg",
          authToken);
    }

    @Override
    public HttpPost request() {
      HttpPost post
        = new HttpPost("http://typea-android-apps.appspot.com/lbmsg/list");

      try {
        List<NameValuePair> parms = new ArrayList<NameValuePair>();

        // 現在の位置を中心に一片指定メートルの位置を得る
        GeoPt currentPos = new GeoPt(loc.getLatitude(),loc.getLongitude());
        GeoPt[] rect = LocationUtil.getRect(currentPos, 3000); // 3km

        parms.add(new BasicNameValuePair("from_lat", formatGeoPt(rect[0].getLat())));
        parms.add(new BasicNameValuePair("from_lon", formatGeoPt(rect[0].getLon())));
        parms.add(new BasicNameValuePair("to_lat",   formatGeoPt(rect[1].getLat())));
        parms.add(new BasicNameValuePair("to_lon",   formatGeoPt(rect[1].getLon())));
        post.setEntity(new UrlEncodedFormEntity(parms, HTTP.UTF_8));

      } catch (UnsupportedEncodingException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
      }
      // return createRequest();
      return post;
    }

    @Override
    public void callback(HttpResponse response) {
      if (response != null) {
        int status = response.getStatusLine().getStatusCode();

        StringBuilder buf = new StringBuilder();
        try {
          InputStream in = response.getEntity().getContent();
          BufferedReader reader = new BufferedReader(new InputStreamReader(in));
          String l = null;
          while((l = reader.readLine()) != null) {
            buf.append(l + "¥r¥n");
          }
          if (status != HttpStatus.SC_OK) {
            Log.e("MyApp", buf.toString());
          }

          String title = getContext().getString(R.string.msg_receive_msg);
          String text   = getContext().getString(R.string.msg_location_msg,
                          loc.getLatitude(),
                          loc.getLongitude());
```

```
            Notification notification
              = new Notification(
                  android.R.drawable.ic_dialog_email,
                  title,
                  System.currentTimeMillis());

            Intent intent = new Intent(getContext(), MessageViewerActivity.class);
            Log.i("MyApp","Service:" + buf.toString());

            String extraname = INTENT_KEY.MESSAGES.toString();
            Log.i("MyApp", "SERVICE:" + extraname);
            intent.putExtra(extraname, buf.toString());

            PendingIntent contentIntent
              = PendingIntent.getActivity(
                  getContext(),
                  0,
                  intent,
                  PendingIntent.FLAG_UPDATE_CURRENT);

            notification.setLatestEventInfo(getContext(), title, text, contentIntent);

            notifManager.notify(R.string.app_name, notification);

          } catch(Exception e) {
            e.printStackTrace();
          }
        }
      }
    });
    break;
  }
}

@Override
public void onProviderDisabled(String provider) {
  // TODO Auto-generated method stub
}

@Override
public void onProviderEnabled(String provider) {
  // TODO Auto-generated method stub
}

@Override
public void onStatusChanged(String provider, int status, Bundle extras) {
  // TODO Auto-generated method stub
}
}
```

このソースを実装しても、2 点ほどコンパイルエラーとなる箇所(赤のマーカー)があります。1 点は、

Enum の利用、もう一点は、存在しない Activity を Intent で呼び出している箇所です。

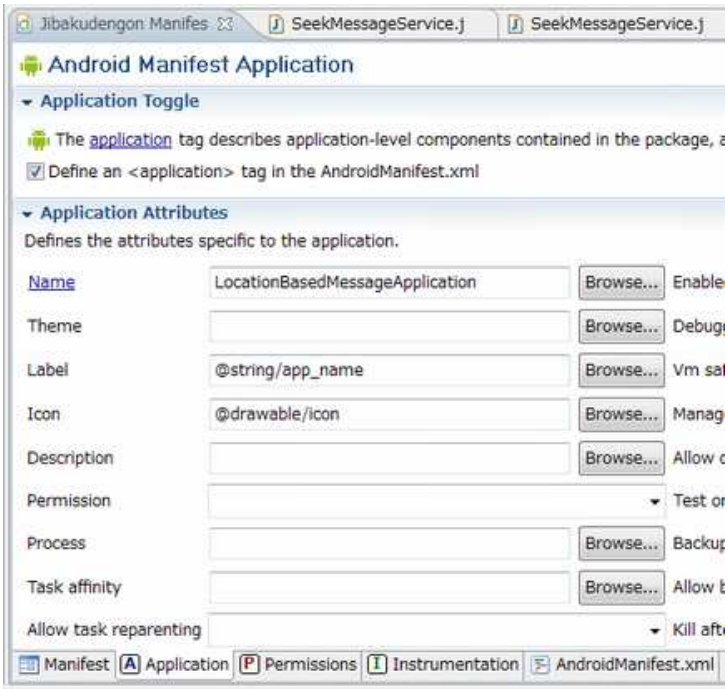
3.2.3 Application をサブクラス化する

Enum がコンパイルエラーかというと、Application クラスをに宣言しているからです。Activity

のライフサイクル上、バックグラウンドになると、メンバー変数とか解放されてしまうのですが、Application は、継続して存在しているので、Application クラスをサブクラス化すると Activity をまたがって設定を保存するまでもないような一時的な値とかを保持しておくのに便利だったりもします。

```
package com.a_yan_android.lbmsg;

import android.app.Application;

public class LocationBasedMessageApplication extends Application{
  public static final String TAG = "LBMSG";

  public enum INTENT_KEY {
    MESSAGES { public String toString() {return "com.a_yan_android.lbmsg.MESSAGES"; } },
  }
}
```

Application クラスを継承して上記のようにアプリ用の Application クラスを作成しますが、これだけでは、実際にうごかしたときの Application のインスタンスは入れ替わりません。AndroidManifest、Application タブの Application Attributes にて指定する必要があります。



3.2.4 Notification と Intent と Activity

サービスが、GAE アプリケーションからメッセージを取得し、メッセージが取得できると、ステータスバーに通知します。そしてステータスバーに通知されたメッセージをタップすると、Intent が飛んで、メッセージを表示する Activity が起動する。

という流れになります。

3.2.4.1 Notification

上記サービスで、NotificationManager が担う部分です。実際には、以下の様に表示されます。

3.2.4.1 Intent

Intent は、画面間でやりとりするメッセージみたいなもので、ある Activity から別の Activity を起動したりすることができます。また、このとき情報を付加することができます。

今回の例では、

```
Intent intent = new Intent(getContext(), MessageViewerActivity.class);
intent.putExtra(extraname, buf.toString());
```

のように、クラス名を指定して Intent を生成し、付随データーとして、文字列を設定しています。

このような Intent を明示的 Intent といいます。

明示的とあえて言うからには、暗黙的 Intent というものがあるのですが、実はこの暗黙的 Intent は、Android アプリのかなり特徴的な機能で、利用者にとっても開発者にとっても、画期的に便利なのではないかと思います。かいつまんで言うと、「メッセージの宛先を明示せずにメッセージを投げて、受けられるアプリがあればそのメッセージを受ける」という非常に緩い、疎結合の極みではないかというようにアプリケーションが連携できてしまうのです。

説明が長くなりそう、且つ、今回は利用しないので、以下を確認いただければ幸いです。

Android 暗黙的 Intent で Twitter や Evernote と連携する

http://typea.info/blg/glob/2011/02/android-intent-twitter-evernote.html

3.2.4.2 Activity

そんな Intent ですが、今回は即時発行ではなく、時間差で発行されるように、PendingIntent を利用します。Activity から別の Activity を Intent で即時呼び出しする場合、

```
startActivity(intent);
```

とすれば良いのですが、時間差で Activity を以下の様にして起動させています。

```
PendingIntent contentIntent
          = PendingIntent.getActivity(
             getContext(),
             0,
             intent,
             PendingIntent.FLAG_UPDATE_CURRENT);
```

起動される Activity については、章を変えて。

6.　その他の機能

最後に、呼び出される側の Activity で、JSON の利用、カメラの利用、アニメーションの利用等を押し込んでいきたいと思います。

6.1 呼び出される Activity

呼び出される側の Activity も、使える概念を結構含んでいるため、import 文以外のソースをまず掲げ、おって説明していきます。

```
package com.a_yan_android.lbmsg;

public class MessageViewerActivity extends Activity implements SurfaceHolder.Callback {
  private Camera camera;
  private SurfaceView preview;
  private SurfaceHolder holder;

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

    preview = (SurfaceView) findViewById(R.id.surview_preview);
    holder = preview.getHolder();
    holder.addCallback(this);
    holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);

    try {

      String extraname = INTENT_KEY.MESSAGES.toString();
      Bundle bundle = getIntent().getExtras();
      String json = bundle.getString(extraname);

      int index = 0;
      if (json != null) {
        JSONArray messages = new JSONArray(json);
        for (int i=0; i<messages.length(); i++) {
          Message msg = new Message();
          msg.fromJSONObject(messages.getJSONObject(i));
          if (!TextUtils.isEmpty(msg.getMsg())) {

            MessageView mv = new MessageView(this, index++, msg.getOwner(), msg.getMsg());
            Animation anim = AnimationUtils.loadAnimation(this,R.anim.push_left_in);
            anim.setDuration(index * 1000);
            mv.startAnimation(anim);

            addContentView(
                mv,new LayoutParams(LayoutParams.FILL_PARENT, LayoutParams.FILL_PARENT));
          }
        }
      }
      bundle.remove(extraname);

    } catch (Exception e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
  }

  @Override
  public void surfaceCreated(SurfaceHolder holder) {
    try {
```

```
      camera = Camera.open();
      camera.setPreviewDisplay(holder);
    } catch (IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
  }

  @Override
  public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3) {

    Camera.Parameters p = null;

    // 擬似的にポートレートモードに
    try {
      camera.stopPreview();
      p = camera.getParameters();
      p.setRotation(90);
      camera.setParameters(p);

    } catch (Exception e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
      return;
    }

    camera.startPreview();
  }

  @Override
  public void surfaceDestroyed(SurfaceHolder holder) {
    camera.stopPreview();
    camera.release();
  }
}
```

4.1.1 SurfaceView と カメラ

「SurfaceView なら Android で高速描画ゲームが作れる」

http://www.atmarkit.co.jp/fsmart/articles/android12/android12_1.html

あたりを読むとわかりやすく説明されています。

『「ウィジェット」も、同様に View のサブクラスですが、それらのウィジェットと SurfaceView には決定的な違いがあります。

それは、描画の方式が違うのです。パッケージ「android.widget」に属するウィジェットは、アプリケーションのスレッド内で描画が行われるため、定期的に再描画を繰り返すゲームなどには向いていません。一方、SurfaceView は、アプリケーションのスレッドと描画処理のスレッドが独立している

ため、定期的な再描画に向いています。』

ということで、カメラのプレビューにて、SurfaceView を利用しています。

Java ってマルチスレッドじゃないの？何で、SurfaceView だけ、描画処理のスレッドが独立していることを特別扱いするの？なんて思われるかもしれませんが、Java に限らず、基本的にほとんどの GUI はシングルスレッドで動作しています。

といっても、アプリケーション全体がシングルスレッドなのではなくて、イベントディスパッチスレッドという、GUI 処理を行うため、イベントをキューイングして、イベントハンドラに送信するための専用のスレッドが作られます。

なので、通常 GUI のコンポーネントを操作するときに、複数のスレッドを考慮して同期したりする必要はないですが、上記のように非同期で処理を行う方が都合がよい場合、そういう用途のクラスを利用する必要があります。例えば、android.os.AcyncTask を利用すると、非同期で処理を行うことができますし、android.os.Handler を利用すると、GUI を非同期で操作することができます。

ちなみに、カメラを利用するには、CAMERA パーミッションが必要です。

4.1.2 Intent を受ける

先ほど、サービス側から、PendingIntent で文字列情報を付加して、Intent を飛ばしましたが、この Activity で、情報を取り出します。

```
Bundle bundle = getIntent().getExtras();
String json = bundle.getString(extraname);
```

4.1.3 JSON を解析

Intent から取り出した文字列は、GAE アプリから受け取ったレスポンスで、JSON そのものなので、これを解析して、Java のオブジェクトを作ります。

そもそも JSON とは、JavaScript Object Notation http://ja.wikipedia.org/wiki/JSON

で、オブジェクトの表記法で、配列は、

```
["milk", "bread", "eggs"]
```

のように角括弧で、マップは、

```
{"name": "John Smith", "age": 33}
```

キーと値をコロンでつないで、中括弧で表現します。

この書き方は、まんま Python なので、ジェイソンのソンはパイソンのソンなのでは？と個人的に密かに思っていますが、裏はとってません。単なる想像です。

で、この表記方法は JavaScript と非常に親和性が高く、JavaScript では、eval() 関数一発で、JavaScript のオブジェクトを生成します。

しかしながら、Android では、org.json.JSONObject や　org.json.JSONArray を利用して、解析する必要があります。

```
JSONArray messages = new JSONArray(json);
for (int i=0; i<messages.length(); i++) {
    JSONObject jsonObj = messages.getJSONObject(i);
    Message msg = new Message();
    msg.setMessage(jsonObj.get("message”));
}
```

JSONObject や、JSONArray を利用すると、Java のクラスと JSON との相互変換が非常に容易になるため、WEB アプリケーションとの連携を非常に簡単に行うことができます。

4.1.4 AR(拡張現実)

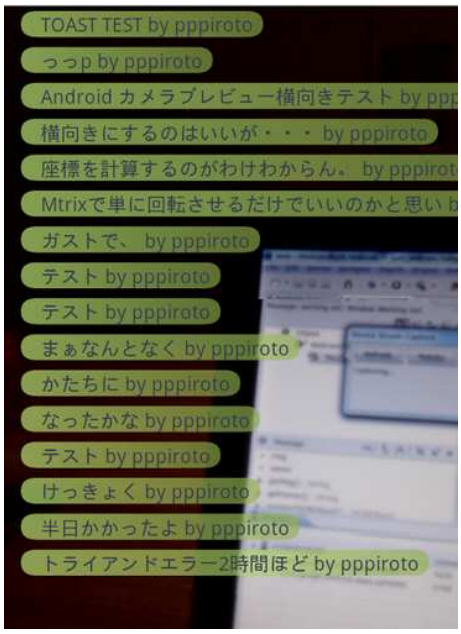
で、この Activity では何をやりたいかというと、AR への取っ掛かりみたいな感じでしょうか。

「バーチャルリアリティと対を成す概念。強化現実とも呼ばれ、現実の環境（の一部）に付加情報としてバーチャルな物体を電子情報として合成提示することを特徴とする。

合成提示される電子情報はアノテーションと呼ばれる。アノテーションは現実環境中の特定の物体に関

する説明や関連情報を含み、説明対象となる実物体近くに提示されることが多い。このため、拡張現実を実現するための技術として使用者が対象を観察する位置など現実環境の情報を取得する技術が基礎技術として重要視されている。」

http://ja.wikipedia.org/wiki/%E6%8B%A1%E5%BC%B5%E7%8F%BE%E5%AE%9F



こんな感じで、カメラプレビューの上に、現在地点付近のメッセージを付箋のように表示させます。

で、この付箋一つ一つが View を継承して作成した MessageView というクラスで、

```
MessageView mv = new MessageView(this);
addContentView(mv,new LayoutParams(LayoutParams.FILL_PARENT, LayoutParams.FILL_PARENT));
```

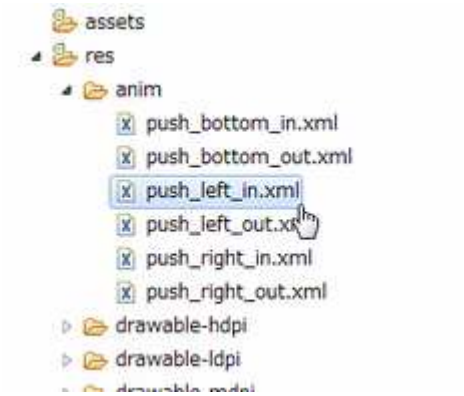
addContentView() を行うことで、カメラプレビューの上にどんどんレイヤーとしてかぶせていくことで、見た目的には、簡単に「アノテーション」を付与することができるようになります。

4.1.5 アニメーション

で、その付箋というかアノテーションというかを、画面に表示するときに、アニメーションを仕掛けているのが、以下の箇所になります。

```
MessageView mv = new MessageView(this, index++, msg.getOwner(), msg.getMsg());
Animation anim = AnimationUtils.loadAnimation(this,R.anim.push_left_in);
anim.setDuration(index * 1000);
mv.startAnimation(anim);
```

```
addContentView(mv,new LayoutParams(LayoutParams.FILL_PARENT, LayoutParams.FILL_PARENT));
```



アニメーションも、リソースとして XML を作成し、/res/anim 等においておけば、上記のように必要

な箇所で利用することができます。XML の内容は、

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
   <translate android:fromXDelta="100%p"
            android:toXDelta="0"
            android:duration="300"/>
   <alpha android:fromAlpha="1.0"
         android:toAlpha="1.0"
         android:duration="300" />
</set>
```

YDelta ではなく、XDelta となっているのは、カメラプレビューを横向きに表示させているためです。

4.1.6 図形描画

View に対して、図形を描画し、透明度を設定したり、グラデーションをつけたりということも割合と

簡単にできます。・・・がねらってやるのは結構大変だったりします。上記付箋部分のビューのソース

を一応貼っておきます。ニュアンスがつかめるでしょうか？

```
package com.a_yan_android.lbmsg;

public class MessageView extends View {
      private int index;
      private String msg;
      private String owner;

      public MessageView(Context context, int index, String owner, String msg) {
             super(context);
             this.index = index;
             this.msg = msg;
             this.owner = owner;
```

```
}

/**
 * カメラプレビューのために、縦向きにビューを設定(90度回転)した場合、
 * 元に戻して(-90度)描画を行うが、縦向を想定した位置を、元に戻した場合に
 * 同じ位置に来るように事前に調整する
 * @param rect
 * @param degrees
 * @return
 */
private RectF prepareRestorePortrait(RectF rect) {
        float left   = rect.left;
        float top    = rect.top;
        float right  = rect.right;
        float bottom = rect.bottom;

        float[] src = {left, top, right, top, right, bottom, left, bottom};
        float[] dst = new float[8];
        Matrix matrix = new Matrix();
        matrix.preTranslate(-getHeight(),0);
        matrix.mapPoints(dst, src);
        //             left    top    right   bottom
        return new RectF(dst[0], dst[1], dst[4],dst[5]);
}

@Override
protected void onDraw(Canvas canvas) {
        final int   FONT_SIZE   = 20;
        final float FONT_MARGIN = 5f;
        final float TEXT_LEFT   = 40f;
        final float MEMO_LEFT   = 20f;
        int   memoColor1 = Color.parseColor("#D3FF82");
        int   memoColor2 = Color.parseColor("#94B24D");
        int   textColor  = Color.parseColor("#363D52");

        canvas.rotate(-90);

        String message = this.msg + " by " + this.owner;

        Paint paintMsg = new Paint();
        paintMsg.setTextAlign(Paint.Align.LEFT);
        paintMsg.setAntiAlias(true);
        paintMsg.setTypeface(Typeface.DEFAULT);
        paintMsg.setColor(textColor);
        paintMsg.setTextSize(FONT_SIZE);

        FontMetrics fontMetrics = paintMsg.getFontMetrics();

        float txtHeight = (fontMetrics.top * -1) + fontMetrics.bottom;
        float txtMargin = txtHeight + (FONT_MARGIN * 2);
        float txtWidth  = paintMsg.measureText(message);

        RectF memoRect = prepareRestorePortrait(
           new RectF(MEMO_LEFT,
            this.index * txtMargin + FONT_MARGIN,
            TEXT_LEFT  + txtWidth  + (FONT_MARGIN * 2),
            this.index * txtMargin + txtHeight + FONT_MARGIN
                                                ));
        Paint paintMemo = new Paint();
        paintMemo.setPathEffect(new CornerPathEffect(16f));
        paintMemo.setAntiAlias(true);
```

```
            paintMemo.setShader(new LinearGradient(
                           memoRect.left, memoRect.top,
                           memoRect.right,memoRect.bottom,
                           memoColor1,memoColor2,
                           Shader.TileMode.CLAMP));
            paintMemo.setAlpha(160);
            paintMemo.setStyle(Style.FILL);

            canvas.drawRect(memoRect, paintMemo);

            RectF txtRect = prepareRestorePortrait(
               new RectF(TEXT_LEFT, this.index * txtMargin,0,0));
            canvas.drawText(message, txtRect.left, txtRect.top + txtHeight ,paintMsg);
      }
}
```

5　最後に

以上で、地縛伝言® の作成は終わりです。

これだけのことが、たったこれだけのコーディングでできる！と思うのか、たったこれだけのことをやるのに、こんなにコーディングしなければいけないのか！と思うのかどちらでしょうか。

アイディアを膨らます一助になればこれ幸いです。