SOM-ITSOLUTIONS

ANDROID

Developing a nearby POI REST Client

SOMENATH MUKHOPADHYAY

19-Dec-15

som-itsolutions
#A2 1/13 South Purbachal Hospital Road Kolkata 700078 Mob: +91 9748185282
Email: som@som-itsolutions.com / som.mukhopadhyay@gmail.com
Website: https://sites.google.com/a/som-itsolutions.com/som-itsolutions/
Blog: www.som-itsolutions.blogspot.com

Today i will show you how to develop an Android Point of Interest REST client and how to integrate it with Google Map.

To develop a REST client, the first thing we need to do is to create a background task which will talk to the web server and thus help the Android app consume the REST Api. Here I have used a simple Asynctask to achieve this. However, there are other libraries like Retrofit, Volley etc which will do the heavy lifting of connecting to the server and downloading the data and you can use these free of cost.
So let us develop the Asynctask. I have developed a somewhat generic Asynctask the code of which is as follows:

```
package com.somitsolutions.training.android.restclient;

import android.content.Context;
import android.os.AsyncTask;
import android.util.Log;

import org.json.JSONObject;

import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashMap;

public class HTTPAsyncTask extends AsyncTask<String, Void, String> {
  private CallBack mCb;
  LinkedHashMap<Object, Object> mData = null;
  //List mParams= new ArrayList();
  LinkedHashMap<Object, Object> mParams = new LinkedHashMap<>();
  String mTypeOfRequest;
  String mStrToBeAppended = "?";
  boolean isPostDataInJSONFormat = false;
  JSONObject mJSONPostData = null;
  Context mContext = null;

  public HTTPAsyncTask(Context context, CallBack c, HashMap<Object, Object> data, JSONObject jsonObj, String request) {
    mContext = context;
    mCb = c;
    mTypeOfRequest = request;
    mJSONPostData = jsonObj;
    //Log.i("JSONDATA", mJSONPostData.toString());
    if((data != null) && (jsonObj == null)){
    mData = (LinkedHashMap)data;

//GET
    if(mTypeOfRequest.equalsIgnoreCase("GET")){
```

```
        Object key = null;
        Iterator<Object> it = mData.keySet().iterator();
        while(it.hasNext()){
          key = it.next();
          mParams.put(key, mData.get(key));
          Log.d("Data", key.toString() + " " + mData.get(key).toString());
          }
        Iterator<Object>itParams = mParams.keySet().iterator();
        int sizeOfParams = mParams.size();
        int index = 0;
        while(itParams.hasNext()){
          Object keyParams = itParams.next();
          index++;
          if (index == sizeOfParams){
            mStrToBeAppended+=  keyParams + "=" + mParams.get(keyParams);
            break;
          }
          mStrToBeAppended+=  keyParams + "=" + mParams.get(keyParams)+ "&";

        }
      }

//POST
    if(mTypeOfRequest.equalsIgnoreCase("POST")){
        Object key = null;
        isPostDataInJSONFormat = false;
        Iterator<Object> it = mData.keySet().iterator();
        while(it.hasNext()){
          key = it.next();
          mParams.put(key, mData.get(key));
          }
        }
      }

//POST with Data in jSON format
    if ((mData == null) && (mJSONPostData != null) && (mTypeOfRequest.equalsIgnoreCase("POST") == true)){
        isPostDataInJSONFormat = true;
        }
  }

  @Override
  protected String doInBackground(String... baseUrls) {
  //android.os.Debug.waitForDebugger();
    publishProgress(null);
    if(mTypeOfRequest.equalsIgnoreCase("GET")){
    String finalURL = baseUrls[0]+ mStrToBeAppended;
    return HttpUtility.GET(finalURL);
```

```
    }

    if (mTypeOfRequest.equalsIgnoreCase("POST")){
      if(isPostDataInJSONFormat == false){
        return HttpUtility.POST(baseUrls[0],mParams );
      }
      if(isPostDataInJSONFormat == true){
        Log.i("JSONDATAPOSTMETHOd","JSON POST method to be called...");
        return HttpUtility.POST(baseUrls[0], mJSONPostData);
      }
    }
    return null;
  }

  // onPostExecute displays the results of the AsyncTask.
  @Override
  protected void onPostExecute(String result) {
    mCb.onResult(result);
  }

  @Override
  protected void onProgressUpdate(Void...voids ) {
  mCb.onProgress();
  }
}
```

Have a look at the doInBackground function of the Asynctask which is actually responsible for communicating with the server in a background thread. In the doInBackground there are specific cases each for “GET”, “POST” & “POST with jSON postdata”. All of these cases are taking the help of an HttpUtility class, the code for which is given below:

```
package com.somitsolutions.training.android.restclient;

import android.util.Log;

import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicHeader;
import org.apache.http.protocol.HTTP;
import org.json.JSONObject;
```

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.HashMap;
import java.util.List;

public class HttpUtility {

  private final static HttpClient mHhttpclient = new DefaultHttpClient();

   public static String GET(String url){
      Log.i("URL", url);
     InputStream inputStream = null;
     String result = "";
     try {

        // make GET request to the given URL
        HttpResponse httpResponse = mHhttpclient.execute(new HttpGet(url));

        // receive response as inputStream
        inputStream = httpResponse.getEntity().getContent();

        // convert inputstream to string
        if(inputStream != null){
            result = convertInputStreamToString(inputStream);
        }

        else
            result = "Did not work!";

     } catch (Exception e) {
        Log.d("InputStream", e.getLocalizedMessage());
     }

     return result;
  }



public static String POST(String url, HashMap<Object, Object> mParams){
  InputStream inputStream = null;
  String result = "";
  try{
     HttpPost post = new HttpPost(url);
     post.setEntity(new UrlEncodedFormEntity((List<? extends NameValuePair>) mParams, "UTF-8"));
```

```
    HttpResponse httpResponse = mHhttpclient.execute(post);
    // receive response as inputStream

    inputStream = httpResponse.getEntity().getContent();

    // convert inputstream to string
    if(inputStream != null){
    result = convertInputStreamToString(inputStream);
    //inputStream.close();
    }
    else
    result = "Did not work!";
    }
  catch (Exception e) {
    Log.d("InputStream", e.getLocalizedMessage());
    }

  return result;

}

public static String POST(String url, JSONObject obj){

     InputStream inputStream = null;
     String result = "";

     try{
        HttpPost post = new HttpPost(url);
        post.setHeader("Content-type", "application/json");

        StringEntity se = new StringEntity(obj.toString());
        se.setContentEncoding(new BasicHeader(HTTP.CONTENT_TYPE, "application/json"));
        post.setEntity(se);

        HttpResponse httpResponse = mHhttpclient.execute(post);

        // receive response as inputStream
        inputStream = httpResponse.getEntity().getContent();

        // convert inputstream to string
        if(inputStream != null){
           result = convertInputStreamToString(inputStream);
        }

        else
           result = "Did not work!";
```

```
    } catch (Exception e) {
        Log.d("InputStream", e.getLocalizedMessage());
    }
    Log.i("JSONPOSTEND", "End of JSON data post methos...");

    return result;
}

public static String convertInputStreamToString(InputStream inputStream) throws IOException{
    BufferedReader bufferedReader = new BufferedReader( new InputStreamReader(inputStream));
    String line = "";
    String result = "";
    while((line = bufferedReader.readLine()) != null)
        result += line;

    inputStream.close();
    return result;

}
}
```

The different static methods of this class are self-explanatory and are responsible for the actual communication with the server.

So once the basic Asynctask infrastructure is ready, we need to concentrate on the UI of the Android app. Even before that we must be sure which REST api we will consume. Here i have used the REST apis from geoNames and it is in the form of

http://api.geonames.org/findNearbyPOIsOSMJSON?lat=22.5735314&lng=88.4331189&username=demo

So its a GET REST Api and we need to pass three parameters to it. One is the latitude, another is the longitude and the last is the username. We can test this REST Api easily with the help of Advanced REST client app for chrome and available at

https://chrome.google.com/webstore/detail/advanced-rest-client/hgmloofddffdnphfgcellkdfbfbjeloo

So if we see the return result of this REST Api, we will see that it consists of an array of Point of interests or POIs near the place whose latitude and longitude we have passed while calling the REST api. Each element of this array are like the followings:

```
{
    "typeName": "bank"
```

```
    "distance": "0.06"
    "name": "United Bank Of India"
    "lng": "88.4337242"
    "typeClass": "amenity"
    "lat": "22.5735671"
}
```


From this Array elements we will extract name, typename, latitude and longitude.

Now to represent these array elements, we have created a Data Model class by the name POIDataModel and which looks as the follows:

```
package com.somitsolutions.training.android.restclient;

/**
* Created by som on 26/12/15.
*/
public class POIDataModel {
  private String name;
  private String typeName;
  private double lat;
  private double lng;

  public POIDataModel(){
     this.name = "";
     this.typeName="";
  }

  public POIDataModel(String name, String typeName){
     this.name = name;
     this.typeName = typeName;
  }

  public String getName(){
     return name;
  }

  public String getTypeName(){
     return typeName;
  }

  public double getLat(){return lat;}
  public double getLng(){return lng;}
```

```
  public  void setName(String name){
     this.name = name;
  }
  public  void setTypeName(String typeName){
     this.typeName = typeName;
  }
  public void setLat(double lat){
     this.lat = lat;
  }
  public void setlng(double lng){
     this.lng = lng;
  }
}
```

This class is simple and self explanatory.

Now once the REST api returns the data in the JSON format and after we extract the necessary data from that response, we need to display them in a list.
So now creates another Activity called DisplayActivity extended from ListActivity.

```
package com.somitsolutions.training.android.restclient;

import android.app.ListActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ListView;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;

public class DisplayActivity extends ListActivity implements View.OnClickListener{
  ArrayList<POIDataModel> mDisplayArray = new ArrayList<POIDataModel>();
  ListView mListView;
  Button mBtnShowOnGoogleMap;
  String displayData;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_display);

     mListView = getListView();
```

```
    mBtnShowOnGoogleMap = (Button)findViewById(R.id.showOnMap);
    mBtnShowOnGoogleMap.setOnClickListener(this);
    displayData = getIntent().getStringExtra("jsonArray");

    try {
       JSONArray jsonDisplayData = new JSONArray(displayData);

       for (int i = 0; i <jsonDisplayData.length(); i++){
          JSONObject jsonData = jsonDisplayData.getJSONObject(i);
          POIDataModel dataModelTemp = new POIDataModel();
          dataModelTemp.setName(jsonData.getString("name"));
          dataModelTemp.setTypeName(jsonData.getString("typeName"));
          mDisplayArray.add(dataModelTemp);
       }
    } catch (JSONException e) {
       // TODO Auto-generated catch block
       e.printStackTrace();
    }

    final POICustomAdapter customArrayAdapter = new POICustomAdapter(getApplicationContext(),
R.layout.singleitemlayout,mDisplayArray);
    setListAdapter(customArrayAdapter);
  }


  @Override
  public void onClick(View view) {
    if(view.equals(mBtnShowOnGoogleMap)){
       Intent displayMapIntent = new Intent(getApplicationContext(), MapsActivity.class);
       displayMapIntent.putExtra("MapData",displayData);
       startActivity(displayMapIntent);

    }
  }
}
```

The XML layout of this Activity is called activity_display and it is as follows

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
```

```
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.somitsolutions.training.android.restclient.DisplayActivity">


    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@android:id/list"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_marginTop="50dp" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Show on Google Map"
        android:id="@+id/showOnMap"
        android:layout_alignParentLeft="true"
        android:layout_alignParentBottom="true" />
</RelativeLayout>
```

Now as you are aware of that for a custom listview we need to create a custom adapter. Our adapter class is called POICustomAdapter and it looks as follows:

```
package com.somitsolutions.training.android.restclient;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import java.util.List;

/**
* Created by som on 26/12/15.
*/
public class POICustomAdapter extends ArrayAdapter<POIDataModel> {

    private Context mContext;
    List<POIDataModel> mDataList;

    public POICustomAdapter(Context context, int resource,
                            List<POIDataModel> objects) {
```

```
    super(context, R.layout.singleitemlayout, objects);
    // TODO Auto-generated constructor stub
    mContext = context;
    mDataList = objects;
  }

  @Override
  public View getView(int position, View convertView, ViewGroup parent) {

    if (convertView == null) {
      // This a new view we inflate the new layout
      LayoutInflater inflater = (LayoutInflater) mContext.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
      convertView = inflater.inflate(R.layout.singleitemlayout, parent, false);
    }

    TextView name = (TextView)convertView.findViewById(R.id.name);
    TextView typeName = (TextView)convertView.findViewById(R.id.typeName);

    POIDataModel temp = mDataList.get(position);

    name.setText(temp.getName());
    typeName.setText(temp.getTypeName());

    return convertView;
  }
}
```

In the getView method, we inflate an XML layout which is responsible to display a single row in the List is called singleitemlayout.xml and it is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="horizontal" android:layout_width="match_parent"
  android:layout_height="match_parent"
  >

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Name"
    android:id="@+id/name"
    android:gravity="left"
    android:layout_weight="2"
    android:singleLine="false"
    android:textColor="#FF33B5E5"/>
```

```
  <TextView
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="Type"
     android:id="@+id/typeName"
     android:gravity="right"
     android:layout_weight="1"
     android:singleLine="false"
     android:textColor="#FF33B5E5"
     />
</LinearLayout>
```

Once all of this infrastructure is ready, we can write the MainActivity class. The code of the MainActivity class is as follows:

```
package com.somitsolutions.training.android.restclient;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.LinkedHashMap;

public class MainActivity extends Activity implements View.OnClickListener{

  Button mButtonGetWeather;
  Button mButtonGetPlace;
  public static double lat;
  public static double lng;

  ProgressDialog mProgressDialog;
  static Context mContext;
  LocationManager lm;
```

```
private static MainActivity mainActivity;

@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  mContext = this;
  mainActivity = this;
  mButtonGetPlace = (Button)findViewById(R.id.buttonPlace);

  mProgressDialog = new ProgressDialog(this);
  mProgressDialog.setTitle("Getting Data...Please wait...");

  mButtonGetPlace.setOnClickListener(this);


  lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
  ////
  class myLocationlistener implements LocationListener {
    @Override
    public void onLocationChanged(Location location) {
      //int counter = 0;
      if (location != null) {
        lng = location.getLongitude();
        lat = location.getLatitude();
      }

    }

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

    }

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

    }

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

    }
  }

  Location location = null;
  LocationListener ll = new myLocationlistener();
```

```
  try {

    // getting GPS status
    boolean isGPSEnabled = lm.isProviderEnabled(LocationManager.GPS_PROVIDER);

    // getting network status
    boolean isNetworkEnabled = lm.isProviderEnabled(LocationManager.NETWORK_PROVIDER);

    if (!isGPSEnabled && !isNetworkEnabled) {
      // no network provider is enabled
    } else {
      // First get location from Network Provider
      if (isNetworkEnabled) {
        lm.requestLocationUpdates( LocationManager.NETWORK_PROVIDER,  35000,  0, ll);
        Log.d("Network", "Network");
        if (lm != null) {
          location = lm.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
          if (location != null) {
            lat = location.getLatitude();
            lng = location.getLongitude();
            Log.i("Network Latitude :", Double.toString(lat));
          }
        }
      }
      //get the location by gps
      if (isGPSEnabled) {
        if (location == null) {
          lm.requestLocationUpdates(LocationManager.GPS_PROVIDER,35000, 0, ll);
          Log.d("GPS Enabled", "GPS Enabled");
          if (lm != null) {location = lm.getLastKnownLocation(LocationManager.GPS_PROVIDER);
            if (location != null) {
              lat = location.getLatitude();
              lng = location.getLongitude();
              Log.i("GPS Location: ", Double.toString(lat));
            }
          }
        }
      }
    }

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

}
 public static MainActivity getMainActivity() {
   return mainActivity;
```

```
}

@Override
public void onClick(View v) {
  // TODO Auto-generated method stub

  if(v.equals(mButtonGetPlace)){
    LinkedHashMap<Object, Object> data = new LinkedHashMap<Object,Object>();
    //String pinStr = mPin.getText().toString();
    //Log.i("Pin", pinStr);
    data.put("lat", lat/*22.5735314*/);
    data.put("lng", lng/*88.4331189*/);
    //Toast.makeText(getApplicationContext(), "Lat:" + Double.toString(lat) + "lng:" + Double.toString(lng),
Toast.LENGTH_LONG).show();
    data.put("username", "demo");

    HTTPAsyncTask httpAsyncTaskGetPlace = new HTTPAsyncTask(mContext,new CallBack(){

      @Override
      public void onProgress() {
      // TODO Auto-generated method stub
      mProgressDialog.show();
      }
      @Override
      public void onResult(String result) {
      mProgressDialog.dismiss();

      // TODO Auto-generated method stub
      //
      if (!result.equals("")) {
          try {
          JSONObject jsonObject = new JSONObject(result);
          Log.i("Data", jsonObject.toString());
          if (jsonObject.length() != 0){
            JSONArray poi = new JSONArray(jsonObject.getString("poi"));
            if(poi.length() != 0){

              Intent intentDisplay = new Intent(getApplicationContext(), DisplayActivity.class);
              intentDisplay.putExtra("jsonArray", poi.toString());
              startActivity(intentDisplay);
            }
            else{
              Toast.makeText(getApplicationContext(), "No Data Available...", Toast.LENGTH_LONG).show();
            }
          }
          else{
            Toast.makeText(getApplicationContext(), "No Data Available...", Toast.LENGTH_LONG).show();
```

```
            }

           }catch(JSONException e){
             // TODO Auto-generated catch block
             e.printStackTrace();
           }
         }
          //no data available
        }

        @Override
        public void onCancel() {
        // TODO Auto-generated method stub

        }
      },data , null, "GET");
      httpAsyncTaskGetPlace.execute("http://api.geonames.org/findNearbyPOIsOSMJSON");
    }
  }
}
```

What it actually does, it calculates the current latitude and the longitude and then calls the Asynctask from the onClick event of its button. Once the response comes from the server, we pass the data to the DisplayActivity. The DisplayActivity then parses the data and displays it in a ListView.

In the ListView there is an option to show the POIs on Google Map as well. So let us discuss how we can integrate this app with Google Map. To integrate with the Map, first of all we have to get the correct API keys from Google Console. There are enough documentation for the same available in the internet. Once we get the API key, is ready do as follows:

- Right click on the package of the App
- Go to New->Google->Google Maps Activity

and create the new Activity.

Now change the Manifest file. Add the permission, meta-data section etc. The final Manifest file will look like the following:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.somitsolutions.training.android.restclient"
  android:versionCode="1"
  android:versionName="1.0">

  <uses-sdk
```

```
    android:minSdkVersion="8"
    android:targetSdkVersion="21" />

<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

<!-- Required OpenGL ES 2.0. for Maps V2 -->
<uses-feature
    android:glEsVersion="0x00020000"
    android:required="true" />



<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme">
    <activity
        android:name=".MainActivity"
        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=".DisplayActivity"
        android:label="@string/title_activity_display" />

    <!--
        The API key for Google Maps-based APIs is defined as a string resource.
        (See the file "res/values/google_maps_api.xml").
        Note that the API key is linked to the encryption key used to sign the APK.
        You need a different API key for each encryption key, including the release key that is used to
        sign the APK for publishing.
        You can define the keys for the debug and release targets in src/debug/ and src/release/.
    -->
    <meta-data
            android:name="com.google.android.geo.API_KEY"
```

```
        android:value="@string/google_maps_key" />

    <activity
      android:name=".MapsActivity"
      android:label="@string/title_activity_maps"></activity>
  </application>

</manifest>
```

In the second line of the meta-data section put the API Key that you have got.

Now this infrastructure is ready, change the MapsActivity class. Parse the data passed from the DisplayActivity and extract the latitude and longitude and then display it by creating new Markers for each of the POIs data. The MapsActivity class looks like the following:

```
package com.somitsolutions.training.android.restclient;

import android.content.Intent;
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.util.Log;

import com.google.android.gms.maps.CameraUpdate;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.List;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {

  private GoogleMap mMap;
  String mDisplayData;
  List<POIDataModel> mapData = new ArrayList<POIDataModel>();

  @Override
  protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_maps);
    // Obtain the SupportMapFragment and get notified when the map is ready to be used.
    SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.map);
    mapFragment.getMapAsync(this);
    Intent temp = getIntent();
    mDisplayData = temp.getStringExtra("MapData");
    try {
        JSONArray jsonDisplayData = new JSONArray(mDisplayData);

        for (int i = 0; i <jsonDisplayData.length(); i++){
            JSONObject jsonData = jsonDisplayData.getJSONObject(i);
            POIDataModel dataModelTemp = new POIDataModel();
            dataModelTemp.setLat(jsonData.getDouble("lat"));
            dataModelTemp.setIng(jsonData.getDouble("lng"));
            dataModelTemp.setName(jsonData.getString("name"));
            mapData.add(dataModelTemp);
        }
    } catch (JSONException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

}


/**
 * Manipulates the map once available.
 * This callback is triggered when the map is ready to be used.
 * This is where we can add markers or lines, add listeners or move the camera. In this case,
 * we just add a marker near Sydney, Australia.
 * If Google Play services is not installed on the device, the user will be prompted to install
 * it inside the SupportMapFragment. This method will only be triggered once the user has
 * installed Google Play services and returned to the app.
 */
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;

    // Add a marker in Sydney and move the camera
    /*LatLng sydney = new LatLng(-34, 151);
    mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney"));
    mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));*/

    for (int i = 0; i < mapData.size(); i++){
        LatLng temp = new LatLng(mapData.get(i).getLat(),mapData.get(i).getLng());
```

```
        String tempName = mapData.get(i).getName();
        Log.i("Marker Title", tempName);
        mMap.addMarker(new MarkerOptions().position(temp).title(tempName));

      }

      CameraUpdate center= CameraUpdateFactory.newLatLng(new LatLng(MainActivity.lat, MainActivity.lng));
      CameraUpdate zoom=CameraUpdateFactory.zoomTo(15);
      mMap.moveCamera(center);
      mMap.animateCamera(zoom);
    }
}
```

Thats it…. Enjoy the App you have just created. For the full source code, get it from

https://github.com/sommukhopadhyay/AndroidRESTClient