JSON Web Services for Android

Author: Bernard O'Leary

Contents

The basics

IDE: eclipse
UI designer: droiddraw
Android libraries: http://developer.android.com/resources/tutorials/hello-world.html

JSON and JSON web services for practice

(JavaScript Object Notation) is a lightweight data-interchange format.

http://www.json.org/

Picked JSON because I was originally doing the same thing with J2ME and it was a natural transition.
Some JSON web service examples:

http://www.geonames.org/export/JSON-webservices.html
http://ws.geonames.org/earthquakesJSON?north=44.1&south=-9.9&east=-22.4&west=55.2

Raw JSON:


```
{"earthquakes":[{"eqid":"2007hear","magnitude":8.4,"lng":101.3815,"src":"us","datet
ime":"2007-09-12 09:10:26","depth":30,"lat":-
4.5172},{"eqid":"2007aqbk","magnitude":8,"lng":156.9567,"src":"us","datetime":"2007
-04-01 18:39:56","depth":10,"lat":-
8.4528},{"eqid":"2007hec6","magnitude":7.8,"lng":100.9638,"src":"us","datetime":"20
07-09-12 21:49:01","depth":10,"lat":-
2.5265},{"eqid":"a00043nx","magnitude":7.7,"lng":100.1139,"src":"us","datetime":"20
10-10-25 12:42:22","depth":20.6,"lat":-
3.4841},{"eqid":"2010utc5","magnitude":7.7,"lng":97.1315,"src":"us","datetime":"201
0-04-06
20:15:02","depth":31,"lat":2.3602},{"eqid":"2009mebz","magnitude":7.6,"lng":99.9606
,"src":"us","datetime":"2009-09-30 08:16:09","depth":80,"lat":-
0.7889},{"eqid":"2009kdb2","magnitude":7.6,"lng":92.9226,"src":"us","datetime":"200
9-08-10
17:55:39","depth":33.1,"lat":14.0129},{"eqid":"2010zbca","magnitude":7.6,"lng":123.
533,"src":"us","datetime":"2010-07-23
20:51:11","depth":576.3,"lat":6.4939},{"eqid":"2010xkbv","magnitude":7.5,"lng":91.9
379,"src":"us","datetime":"2010-06-12
17:26:50","depth":35,"lat":7.7477},{"eqid":"2007fubd","magnitude":7.5,"lng":107.655
3,"src":"us","datetime":"2007-08-08 15:04:58","depth":289.2,"lat":-5.9682}]}
```


Indented:


```
{"earthquakes":
	[
	{
		"eqid":"2007hear"
		,"magnitude":8.4
		,"lng":101.3815
		,"src":"us"
		,"datetime":"2007-09-12 09:10:26"
		,"depth":30
		,"lat":-4.5172
	}
	,{

	…

	}
	]
}
```


Translated into Java:

```
package com.infostructure.experimental;

public class Earthquake {

	public String eqid;
	public String src;
	public String datetime;
	public double magnitude;
	public double lng;
	public double lat;
	public double depth;

	@Override
	public String toString()
	{
		return "eqid: "+alertid+ " src: "+alerttext+ " datetime: "+alertdate;
```

```
        }
}
```

Getting your app to talk HTTP and JSON

Shamelessly borrowed some code from “Jose C Gomez”:

http://www.josecgomez.com/2010/04/30/android-accessing-restfull-web-services-using-json/

WebService class

```
package com.infostructure.experimental;

import java.io.IOException;
import java.io.InputStream;
import java.io.UnsupportedEncodingException;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLConnection;
import java.net.URLEncoder;
import java.util.Map;

import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.methods.HttpPut;
import org.apache.http.client.methods.HttpDelete;
import org.apache.http.client.params.ClientPNames;
import org.apache.http.client.params.CookiePolicy;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpConnectionParams;
import org.apache.http.params.HttpParams;
import org.apache.http.protocol.BasicHttpContext;
import org.apache.http.protocol.HttpContext;
import org.apache.http.util.EntityUtils;
import org.json.JSONException;
import org.json.JSONObject;

import android.util.Log;

import com.google.gson.Gson;

public class WebService{

    DefaultHttpClient httpClient;
    HttpContext localContext;
    private String ret;

    HttpResponse response = null;
    HttpPost httpPost = null;
    HttpGet httpGet = null;
    HttpPut httpPut = null;
    HttpDelete httpDelete = null;
    String webServiceUrl;

    //The serviceName should be the name of the Service you are going to be using.
```

```
    public WebService(String serviceName){
        HttpParams myParams = new BasicHttpParams();

        HttpConnectionParams.setConnectionTimeout(myParams, 10000);
        HttpConnectionParams.setSoTimeout(myParams, 10000);
        httpClient = new DefaultHttpClient(myParams);
        localContext = new BasicHttpContext();
        webServiceUrl = serviceName;

    }

    //Use this method to do a HttpPost\WebInvoke on a Web Service
    public String webInvoke(String methodName, Map<String, Object> params) {

       JSONObject jsonObject = new JSONObject();

       for (Map.Entry<String, Object> param : params.entrySet()){
             try {
                    jsonObject.put(param.getKey(), param.getValue());
                    }
             catch (JSONException e) {
                    Log.e("Groshie", "JSONException : "+e);
                    }
       }
        return webInvoke(methodName,  jsonObject.toString(), "application/json");
    }

    // POST
    private String webInvoke(String methodName, String data, String contentType) {
        ret = null;

        httpClient.getParams().setParameter(ClientPNames.COOKIE_POLICY,
CookiePolicy.RFC_2109);

        httpPost = new HttpPost(webServiceUrl + methodName);
        response = null;

        StringEntity tmp = null;

        //httpPost.setHeader("User-Agent", "SET YOUR USER AGENT STRING HERE");
        httpPost.setHeader("Accept",
"text/html,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,i
mage/png,*/*;q=0.5");

        if (contentType != null) {
            httpPost.setHeader("Content-Type", contentType);
        } else {
            httpPost.setHeader("Content-Type", "application/x-www-form-
urlencoded");
        }

        try {
            tmp = new StringEntity(data,"UTF-8");
        } catch (UnsupportedEncodingException e) {
            Log.e("Groshie", "HttpUtils : UnsupportedEncodingException : "+e);
        }

        httpPost.setEntity(tmp);

        Log.d("Groshie", webServiceUrl + "?" + data);

        try {
            response = httpClient.execute(httpPost,localContext);

            if (response != null) {
                ret = EntityUtils.toString(response.getEntity());
            }
        } catch (Exception e) {
```

```
            Log.e("Groshie", "HttpUtils: " + e);
        }

        return ret;
    }

    // GET
    //Use this method to do a HttpGet/WebGet on the web service
    public String webGet(String methodName, Map<String, String> params) {
       String getUrl = webServiceUrl + methodName;

       int i = 0;
       for (Map.Entry<String, String> param : params.entrySet())
       {
              if(i == 0){
                     getUrl += "?";
              }
              else{
                     getUrl += "&";
              }

              try {
                            getUrl += param.getKey() + "=" +
URLEncoder.encode(param.getValue(),"UTF-8");
                     } catch (UnsupportedEncodingException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                     }

              i++;
       }

        httpGet = new HttpGet(getUrl);
        Log.e("WebGetURL: ",getUrl);

        try {
            response = httpClient.execute(httpGet);
        } catch (Exception e) {
            Log.e("Groshie:", e.getMessage());
        }

        // we assume that the response body contains the error message
        try {
            ret = EntityUtils.toString(response.getEntity());
        } catch (IOException e) {
            Log.e("Groshie:", e.getMessage());
        }

        return ret;
    }

    public static JSONObject Object(Object o){
       try {
                     return new JSONObject(new Gson().toJson(o));
              } catch (JSONException e) {
                     e.printStackTrace();
              }
              return null;
    }

    public InputStream getHttpStream(String urlString) throws IOException {
        InputStream in = null;
        int response = -1;

        URL url = new URL(urlString);
        URLConnection conn = url.openConnection();

        if (!(conn instanceof HttpURLConnection))
```

```
            throw new IOException("Not an HTTP connection");

        try{
            HttpURLConnection httpConn = (HttpURLConnection) conn;
            httpConn.setAllowUserInteraction(false);
            httpConn.setInstanceFollowRedirects(true);
            httpConn.setRequestMethod("GET");
            httpConn.connect();

            response = httpConn.getResponseCode();

            if (response == HttpURLConnection.HTTP_OK) {
                in = httpConn.getInputStream();
            }
        } catch (Exception e) {
            throw new IOException("Error connecting");
        } // end try-catch

        return in;
    }

    public void clearCookies() {
        httpClient.getCookieStore().clear();
    }

    public void abort() {
        try {
            if (httpClient != null) {
                System.out.println("Abort.");
                httpPost.abort();
            }
        } catch (Exception e) {
            System.out.println("Your App Name Here" + e);
        }
    }
}
```

GSON

In order to get this class to compile, need to get the GSON library:

http://code.google.com/p/google-gson/
"*A Java library to convert JSON to Java objects and vice-versa*"

…and add to your project's build path.

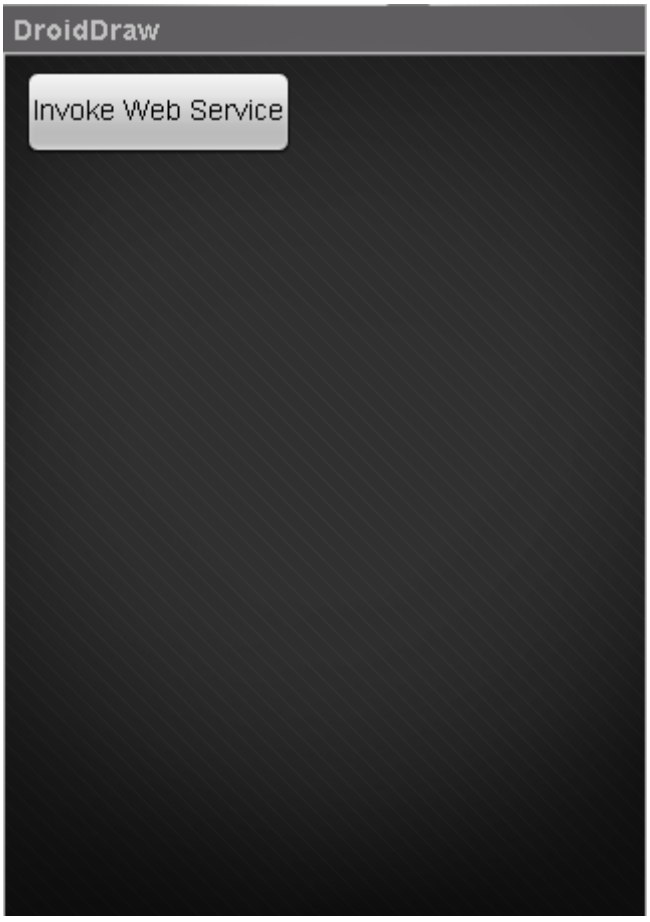
User interface

Use droiddraw to put a basic UI together:

```
<?xml version="1.0" encoding="utf-8"?>
       <AbsoluteLayout
              android:id="@+id/widget0"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
```

```
            xmlns:android="http://schemas.android.com/apk/res/android">
        <Button
            android:id="@+id/btninvokeWebService"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Invoke Web Service"
            android:layout_x="10px"
            android:layout_y="10px">
        </Button>
</AbsoluteLayout>
```

Looks like this:



Put the generated XML code into "<YOUR_PROJECT_NAME>\res\layout\main.xml".

Security

To get Android to allow internet access, edit the Android project's "AndroidMainfest.xml" file:

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

```
        </uses-permission>
</manifest>
```

Putting it all together

A simple front-end that uses the “WebService” and “Earthquake” classes, the droiddraw UI output and by implication, the GSON library:

```
package com.infostructure.experimental;

import java.lang.reflect.Type;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;

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

import android.widget.EditText;
import android.widget.TextView;
import android.widget.Button;
import android.util.Log;
import android.view.View;

public class MainActivity extends Activity
{
	private Button btninvokeWebService;

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

	private void initControls()
	{
		btninvokeWebService.setOnClickListener(new Button.OnClickListener() {
public void onClick (View v){ invokeWebService(); }});
	}

	private void invokeWebService()
	{
		// Instantiate the Web Service Class with he URL of the web service
not that you must pass
		WebService webService = new
WebService("http://ws.geonames.org/earthquakesJSON");

		//Pass the parameters if needed
		Map<String, String> params = new HashMap<String, String>();
		params.put("north", "44.1");
		params.put("south", "-9.9");
		params.put("east", "-22.4");
		params.put("west", "55.2");

		//Get JSON response from server the "" are where the method name would
normally go if needed example
```

```
                String response = webService.webGet("", params);

                try
                {
                        //Parse Response into our object
                        Type collectionType = new
TypeToken<List<Earthquake>>(){}.getType();
                        List<Earthquake> quakes = new Gson().fromJson(response,
collectionType);
                }
                catch(Exception e)
                {
                        Log.d("Error: ", e.getMessage());
                }
        }
}
```

Further details

http://bernard-on-technology.blogspot.com/2010/10/programming-droid.html