Connectivity

Using Web Services

Contents

Using Web Services

In this first example we will see how to use connectivity with Web Services, using REST and parsing the response in JSON format. This example will use some Web Services of Google Maps.

We have to create a new Project called “Connectivity”.

The aim of this example is to search places using a Web Service of Google Maps. The user will enter the name of a place in a text field and it will be searched in Google Maps. Google Maps will return a list of possible results in JSON.

Firstly we have to edit the default layout file. In the *res/layout* folder, we have to open the “*activity_main.xml*” file.

Once the file is opened, we will add a new “*EditText*” and a new “*Button*”. In the “*EditText*” field the user could enter a place for being searched. We change the labels of the Button and TextView. Both labels have to be in the “*res/values/strings.xml*” file.

The layout file is as follows:

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

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/text" />

    <EditText
        android:id="@+id/editText1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/accept" />

</LinearLayout>
```

Then in the activity we create a class attribute of type EditText. This field will get the information that the user introduces:

```
EditText place;
```

In the "*onCreate*" method we have to link the EditText with the EditText of the layout.

```
place = (EditText)findViewById(R.id.editText1);
```

Now we have to create the "*OnClickListener*" method for the button:

```
Button button = (Button)findViewById(R.id.button1);
button.setOnClickListener(new OnClickListener() {

	@Override
	public void onClick(View arg0) {
    }
});
```

We will create a method called "*search()*", which will implement the search with the Web Service. This method returns the coordinates. The method has a string parameter, which is the content of the "*EditText*".

```
double[] coordinates=search(place.getText().toString());
```

The following lines of code implement the "*search()*" method, which is out of the "*onCreate()*" method.

```
public double[] search(String place)
{
	double[] coordinates = new double[2];
	String stringHTTP1 =
"http://maps.googleapis.com/maps/api/geocode/json?address=";
	String stringHTTP2 = "&sensor=false";
	String stringHTTP = stringHTTP1 + place + stringHTTP2;
}
```

In this first step, we are creating an array of type double with two positions (values), and then we create a String with the connection string (URL) to the Web Service.

Now we will filter the URL (it is composed of the content of *EditText*) to prevent the app to crash if the user enters blanks (' '). These blanks will be replaced by the "%20" string.

```
stringHTTP = deleteBlanks(stringHTTP);
```

Implementation of the "*deleteBlanks*" method is as follows:

```
public String deleteBlanks(String stringHTTP)
{
	//Replace the blanks in the  URL
	return stringHTTP.replace(" ", "%20");
}
```

Now we have to implement the consumer of the Web Service. We create a *try-catch* block for avoiding possible network exceptions. With this we send the request and receive the response, in JSON format:

```
try
{
	HttpClient client = new DefaultHttpClient();
	HttpGet request = new HttpGet();
	request.setURI(new URI(stringHTTP));
	HttpResponse response = client.execute(request);
	BufferedReader in = new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
	StringBuffer sb = new StringBuffer("");
	String line = "";
	while ((line = in.readLine()) != null)
	{
		sb.append(line);
	}
	in.close();

	System.out.println(sb.toString());
}
catch (Exception e)
{
	e.printStackTrace();
}
```

The "*System.out.println*" line will show the response of the server in the Eclipse console, in this case the response is in JSON format.

Now we have to parse the JSON response (in *try-catch* block, after the *in.close()* line) for getting the coordinates (latitude and longitude). Other data could be obtained, such as name, description, etc.

```
JSONObject jsonObject = new JSONObject(sb.toString());

String resp = jsonObject.getString("status");

if (resp.equals("OK"))
{
	JSONArray array = jsonObject.getJSONArray("results");
	JSONObject item = array.getJSONObject(0);

	JSONObject point =
item.getJSONObject("geometry").getJSONObject("location");

	coordinates[0] = point.getDouble("lat");
	coordinates[1] = point.getDouble("lng");

	System.out.println("Latitude: "+coordinates[0]+" - Longitude:
"+coordinates[1]);
}
```

Note: In this case we are getting only the first element of the results (*array.getJSONObject(0)*), but it would be interesting to get all the results.

The method for parsing JSON is similar to XML.

Once this is done, we have to return the coordinates in the "*search()*" method:

```
return coordinates;
```

Finally, we have to add permission to access to the Internet. To do this, we add the following line to the "*AndroidManifest.xml*" file (before *Application* node).

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

Summarizing, the whole "*search()*" method is shown below:

```
public double[] search(String place)
{
	double[] coordinates = new double[2];
	String stringHTTP1 =
"http://maps.googleapis.com/maps/api/geocode/json?address=";
	String stringHTTP2 = "&sensor=false";
	String stringHTTP = stringHTTP1 + place + stringHTTP2;

	stringHTTP = deleteBlanks(stringHTTP);

	try
	{
		HttpClient client = new DefaultHttpClient();
		HttpGet request = new HttpGet();
		request.setURI(new URI(peticionHTTP));
		HttpResponse response = client.execute(request);
		BufferedReader in = new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
		StringBuffer sb = new StringBuffer("");
		String line = "";
		while ((line = in.readLine()) != null)
		{
			sb.append(line);
		}
		in.close();

		System.out.println(sb.toString());

		JSONObject jsonObject = new JSONObject(sb.toString());

		String resp = jsonObject.getString("status");

		if (resp.equals("OK"))
		{
			JSONArray array =
jsonObject.getJSONArray("results");
			JSONObject item = array.getJSONObject(0);
```

```
                    JSONObject point =
item.getJSONObject("geometry").getJSONObject("location");

                    coordinates[0] = point.getDouble("lat");
                    coordinates[1] = point.getDouble("lng");

                    System.out.println("Latitude: "+coordinates[0]+" -
Longitude: "+coordinates[1]);
                }
      }
        catch (Exception e)
        {
                e.printStackTrace();
        }
        return coordinates;
}
```

Now we could test the application and check the results.

If we check the app we probably will get a network exception. This is because we have to use a thread or an asynctask for consuming the Web Service. Last versions of Android incorporated a mechanism called "strictmode", which does not allow blocking the device for a HTTP request.

To solve this, it is recommended to use threads or an asynctask. We will replace the "*search*" method for an asynctask:

```
public class Search extends AsyncTask<String, Void, Double[]> {
}
```

This async task will receive a string (the place for searching) and will return an array of Double with latitude and longitude.

Inside of this class there are two methods called "*doInBackground*" and "*onPostExecute*". The first one allows implementing the request to the Web Service and the second one is called when the task is finished. The second method will be used to display the second activity.

For calling this async task we have to write the following lines:

```
new Search().execute(place.getText().toString());
Toast.makeText(getBaseContext(),                    R.string.searching,
Toast.LENGTH_LONG).show();
```

Finally, the first activity will be as follows:

```
package com.example.sesion2gmaps;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.URI;
```

```
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONArray;
import org.json.JSONObject;

import android.app.Activity;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends Activity {

       EditText place;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
place = (EditText) findViewById(R.id.editText1);

        Button button = (Button) findViewById(R.id.button1);
        button.setOnClickListener(new OnClickListener() {

             @Override
             public void onClick(View v) {

       new Search().execute(place.getText().toString());
       Toast.makeText(getBaseContext(), R.string.searching,
Toast.LENGTH_LONG).show();
             }
       });
    }

    public class Search extends AsyncTask<String, Void, Double[]> {

       @Override
       protected Double[] doInBackground(String... params) {
             Double[] coordinates = new Double[2];

             String stringHTTP1 =
"http://maps.googleapis.com/maps/api/geocode/json?address=";
             String stringHTTP2 = "&sensor=false";
             String stringHTTP = stringHTTP1 + params[0] + stringHTTP2;

             stringHTTP = deleteBlanks(stringHTTP);

             System.out.println(stringHTTP);
```

```
            try
            {
                  HttpClient client = new DefaultHttpClient();
                  HttpGet request = new HttpGet();
                  request.setURI(new URI(stringHTTP));
                  HttpResponse response = client.execute(request);

                  BufferedReader in = new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));

                  StringBuffer sb = new StringBuffer("");
                  String line = "";

                  while ((line = in.readLine()) != null)
                  {
                        sb.append(line);
                  }

                  in.close();

                  System.out.println(sb.toString());
                  JSONObject jsonObject = new
JSONObject(sb.toString());
                  String resp = jsonObject.getString("status");

                  if (resp.equals("OK"))
                  {
                        JSONArray array =
jsonObject.getJSONArray("results");
                        JSONObject item = array.getJSONObject(0);

                        JSONObject point =
item.getJSONObject("geometry").getJSONObject("location");

                        coordinates[0] = point.getDouble("lat");
                        coordinates [1] = point.getDouble("lng");

                        System.out.println("Latitude: "+
coordinates[0]+" - Longitude: "+ coordinates[1]);
                  }
            }
            catch (Exception e)
            {
                  e.printStackTrace();
            }

            return coordinates;
            }

            protected void onPostExecute(Double[] coords)
            {
                  Toast.makeText(getBaseContext(), R.string.done,
Toast.LENGTH_LONG).show();
            }
    }
```

```
    public String deleteBlanks(String place)
    {
       return place.replace(" ", "%20");
    }
}
```

Practical exercise 1

Based on the above code, show (e.g., in the console) all the results (i.e., not just the first one) obtained after searching a place. Try searching “Paris” for getting more than one result and being able to test it correctly.

Practical exercise 2

In addition to the latitude and longitude, show the country of each result.