Weather Forecast using JSON Parser

This module will help you to devour and employ the Json Parsing effectively to exchange data for your Application.

About The Author

Megha Das is an IT graduate from National Institute of Science & Technology. Besides being a technical writer, she is an avid blogger, voracious reader and a creative persona. When she isn't hooked to her writing cell, she hones up her culinary skills and taps her way to dancing trance.

Overview

JSON (JavaScript Object Notation) is a lightweight data-interchange format. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

JSON object

A JSON object contains key/value pairs like map. The keys are strings and the values are the JSON types. Keys and values are separated by comma. The { (curly brace) represents the json object.

```
1. {
2.    "employee": {
3.       "name":      "Megha",
4.       "salary":     40000,
5.       "Status":    “Single”
6.    }
7. }
```

Advantage of JSON over XML

1) JSON is faster and easier than xml for AJAX applications.

2) Unlike XML, it is shorter and quicker to read and write.

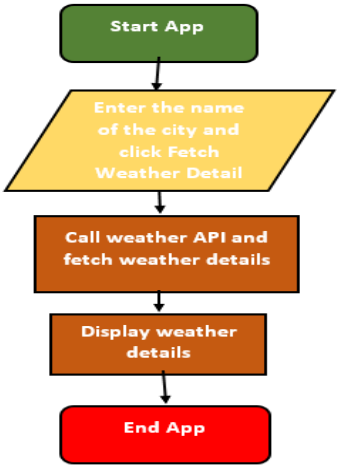
3) It uses array.

It is used primarily to transmit data between a server and web application, as an alternative to XML.Although originally derived from the JavaScript scripting language, JSON is a language-independent data format. Code for parsing and generating JSON data is readily available in many programming languages.

Overview of the App

This is just a simple example of JSON parsing app. The app will take the details of the weather from an open source web service called **Openweathermap**. The API exposed by this web service can be used to get the weather details of any location. The app will ask the user for the location name and fetch the details in the JSON format, which then will be parsed to show particular details on the app.

Basic Flowchart



Design Layout For The Application

activity_main.xml

```
<TextView
    android:id="@+id/textView1"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_marginTop="25dp"
    android:text="@string/location"/>
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
      android:layout_toRightOf="@+id/textView1"
    android:layout_marginLeft="20dp"
    android:layout_height="wrap_content"
    android:layout_alignBottom="@+id/textView1"
    android:layout_alignParentRight="true"
    android:ems="10"/>
<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/textView1"
    android:layout_below="@+id/textView1"
    android:layout_marginTop="68dp"
    android:text="@string/country"
```

```
android:textAppearance="?android:attr/textAppearanceSmall" />
   <TextView
      android:id="@+id/textView3"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_below="@+id/textView2"
      android:layout_marginTop="19dp"
      android:text="@string/temperature"

android:textAppearance="?android:attr/textAppearanceSmall" />
<TextView
      android:id="@+id/textView4"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignLeft="@+id/textView3"
      android:layout_below="@+id/textView3"
      android:layout_marginTop="32dp"
      android:text="@string/humidity"

android:textAppearance="?android:attr/textAppearanceSmall" />
 <TextView
      android:id="@+id/textView5"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
```

```
      android:layout_alignRight="@+id/textView4"
      android:layout_below="@+id/textView4"
      android:layout_marginTop="21dp"
      android:text="@string/pressure"

android:textAppearance="?android:attr/textAppearanceSmall" />
   <EditText
      android:id="@+id/editText2"
      android:clickable="false"
      android:layout_width="wrap_content"
      android:layout_toRightOf="@+id/textView2"
      android:layout_marginLeft="20dp"
      android:layout_height="wrap_content"
      android:layout_alignParentRight="true"
      android:layout_alignBottom="@+id/textView2"
      android:layout_above="@+id/textView3"
      android:ems="10" >
 <requestFocus />
   </EditText>
<EditText
      android:id="@+id/editText3"
      android:clickable="false"
      android:layout_width="wrap_content"
      android:layout_toRightOf="@+id/textView3"
```

```
    android:layout_marginLeft="20dp"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/textView3"
    android:layout_alignBottom="@+id/textView3"
    android:layout_alignLeft="@+id/editText2"
    android:ems="10" />

  <EditText
    android:id="@+id/editText4"
    android:clickable="false"
    android:layout_width="wrap_content"
    android:layout_toRightOf="@+id/textView4"
    android:layout_marginLeft="20dp"
    android:layout_height="wrap_content"
    android:layout_above="@+id/textView5"
    android:layout_alignLeft="@+id/editText1"
    android:ems="10" />
<EditText
    android:id="@+id/editText5"
    android:clickable="false"
    android:layout_width="wrap_content"
    android:layout_toRightOf="@+id/textView5"
    android:layout_marginLeft="20dp"
    android:layout_height="wrap_content"
```

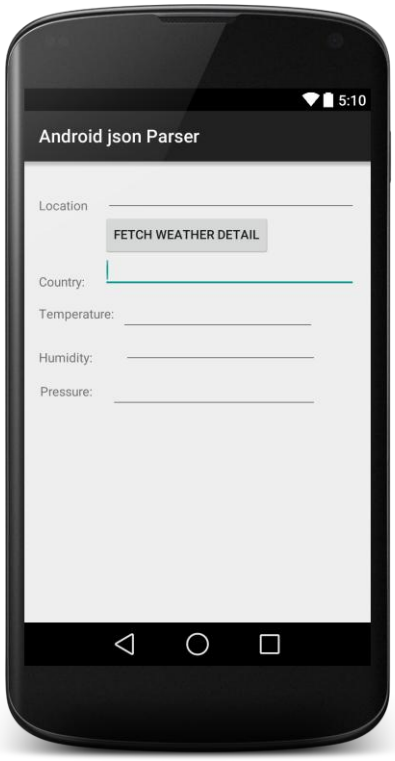
```
    android:layout_alignBaseline="@+id/textView5"
    android:layout_alignBottom="@+id/textView5"
    android:layout_alignRight="@+id/editText4"
    android:ems="10" />

  <Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/editText2"
    android:layout_below="@+id/editText1"
    android:onClick="open"
    android:text="@string/weather" />
```

This is a simple layout, which has a **RelativeLayout** as the base layout. The layout height and width is set to match parent. The padding left, right, top and bottom is set to 16dp. The **button, TextView** and **EditText** as its child views. The base layout of core activity is the **MainActivity**. The details of **button, Textview** and **EditText** are as follows:

Button – The height and width of the button layout is set to **wrap content**. The text of the button is set to **Fetch weather Detail**. The Button is aligned left and below **editText1**. The button on click is set to **Open** and id is set to **button1**.

TextView – There are five TextViews. The height and width of the TextViews is set to wrap content. The alignParentLeft for the TextViews is set to true. All the TextViews are aligned left. The id's assigned to these TextViews are as follows: textView1, textView2, textView3, textView4 and textView5. The text for these TextViews are as follows: Location, Country, Temperature, Humidity and Pressure.

EditText – There are five EditTexts. The height and width of the EditText is set to wrap content. The layout alignParentRight is set to true for all the EditTexts. The layout marginLeft of EditTexts are set to True. The EditText id's are set as follows: editText1, editText2, editText3, editText4 and editText5 The clickable for editText5 is set to true and for all the rest of the EditTexts is set to false.



strings.xml

```
<string name="app_name">Android Xml Parser</string>
<string name="action_settings">Settings</string>
<string name="location">Location</string>
<string name="country">Country:</string>
<string name="temperature">Temperature:</string>
<string name="humidity">Humidity:</string>
<string name="pressure">Pressure:</string>
<string name="weather">Fetch Weather Detail</string>
```

AndroidMenifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.jsonparser" >
   <uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="21" />
   <uses-permission
android:name="android.permission.INTERNET"/>
   <application
      android:allowBackup="true"
      android:icon="@mipmap/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>
```

```
    </application>


</manifest>
```

The AndroidManifest.xml file contains information of your package including components of the application such as activities, services, broadcast receivers, content providers etc. This app will be fetching the weather from a web service, therefore the permission is requested to access the internet using **android.permission.INTERNET**. The application details such as description, icon, label, theme, allowBackup is provided. The Android Manifest file contains only single activities. This activity provides the screen with which the user interacts in order to give the location name. Once the location details are given the weather details are displayed.

MainActivity.java

```
public class MainActivity extends ActionBarActivity {

    private Context context;

    private String url1 = "http://api.openweathermap.org/data/2.5/weather?q=";

    private String url2 = "&mode=json";

    EditText location, country, temperature, humidity, pressure;

    TextView countryLabel, temperatureLabel, humidityLabel, pressureLabel;

    private Weatherbean obj;


    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
        context = this;
        location = (EditText) findViewById(R.id.editText1);
        country = (EditText) findViewById(R.id.editText2);
        temperature = (EditText) findViewById(R.id.editText3);
        humidity = (EditText) findViewById(R.id.editText4);
        pressure = (EditText) findViewById(R.id.editText5);
        countryLabel = (TextView) findViewById(R.id.textView2);
        temperatureLabel = (TextView) findViewById(R.id.textView3);
        humidityLabel = (TextView) findViewById(R.id.textView4);
        pressureLabel = (TextView) findViewById(R.id.textView5);



        country.setKeyListener(null);
        temperature.setKeyListener(null);
        humidity.setKeyListener(null);
        pressure.setKeyListener(null);

        toggleFieldsVisibility(false);

    }

    public void toggleFieldsVisibility(Boolean visible) {
```

```
    int state = (visible) ? View.VISIBLE : View.INVISIBLE;


    country.setVisibility(state);
    countryLabel.setVisibility(state);
    temperature.setVisibility(state);
    temperatureLabel.setVisibility(state);
    humidity.setVisibility(state);
    humidityLabel.setVisibility(state);
    pressureLabel.setVisibility(state);
    pressure.setVisibility(state);


}


public void open(View view) {
    String url = location.getText().toString();
    if (!url.equals("")) {
        String finalUrl = url1 + url + url2;
        Log.d("Final url", finalUrl);
        country.setText(finalUrl);
        obj = new Weatherbean(finalUrl);


        new FetchWeather(obj.getUrlString(), context).execute();


    } else {
```

```
        Toast.makeText(this, "Please enter some place name", Toast.LENGTH_LONG).show();
    }


  }


  private class FetchWeather extends AsyncTask<String, String, String> {
    private String urlStr;
    private Context context;
    private ProgressDialog progressDialog;


    FetchWeather(String url, Context context) {
      urlStr = url;
      this.context = context;
    }



    @Override
    protected void onPreExecute() {
      super.onPreExecute();


      progressDialog = new ProgressDialog(context);
      progressDialog.setMessage("Fetching weather details... please wait");
```

```
        progressDialog.show();
    }

    @Override
    protected String doInBackground(String... params) {
        String result = null;
        try{
            HttpURLConnection con = null ;
            InputStream is = null;

            con = (HttpURLConnection) ( new URL(urlStr).openConnection());
            con.setRequestMethod("GET");
            con.setDoInput(true);
            con.setDoOutput(true);
            con.connect();

            // Let's read the response
            StringBuffer buffer = new StringBuffer();
            is = con.getInputStream();
            BufferedReader br = new BufferedReader(new InputStreamReader(is));
            String line = null;
            while (  (line = br.readLine()) != null )
              buffer.append(line + "\r\n");
```

```
            is.close();
            con.disconnect();
            result =  buffer.toString();
        Log.d("result",result);


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

@Override
protected void onPostExecute(String result)
{
    progressDialog.dismiss();
    Weatherbean weatherObj = new Weatherbean();
    if(result!=null)
    {
        try {
            Log.d("Result",result);
```

```
            JSONObject weatherResponseJson = new
JSONObject(result);



weatherObj.setCountry(weatherResponseJson.getString("name"))
;


            String                  mainStrng                  =
weatherResponseJson.getString("main");


            JSONObject mainObj = new JSONObject(mainStrng);



weatherObj.setTemperature(mainObj.getString("temp"));

weatherObj.setPressure(mainObj.getString("pressure"));

weatherObj.setHumidity(mainObj.getString("humidity"));



            String                  sysString                  =
weatherResponseJson.getString("sys");
            JSONObject sysJson = new JSONObject(sysString);


            weatherObj.setCountry(sysJson.getString("country"));
            weatherObj.setGotResponse(true);
        }
```

```
            catch (JSONException e)
            {
               Log.d("Exception","True");
               weatherObj.setGotResponse(false);
               e.printStackTrace();
            }
         }
         if (weatherObj.isGotResponse()) {

            country.setText(weatherObj.getCountry());
            temperature.setText(weatherObj.getTemperature());
            humidity.setText(weatherObj.getHumidity());
            pressure.setText(weatherObj.getPressure());
            toggleFieldsVisibility(true);
         } else {
            Toast toast = Toast.makeText(context, "Could not fetch weathe for the city. please enter proper city name", Toast.LENGTH_LONG);
            toast.setGravity(Gravity.CENTER, 0, 0);
            toast.show();
            location.setText("");
            toggleFieldsVisibility(false);
         }
 }
   }
```

Declare the variables **'url1'** and **'url2'** of string type and then assign it with the weather data API and Json API. Now declare variables **'location**', **'country'**, **'temperature'**, **'humidity'** and **'pressure'** of '**EditText'** type to display the results. The variables **'countryLabel'** , **'temperatureLabel'**, **'humidityLabel'** and '**pressureLabel'** is declared with the **TextView** type to label the results. The context variable is assign to the current activity context.

Fetch the reference for the above-mentioned variables using **'findViewById'** , then typecast it before assigning it to the concerned variables . Since we have implemented the EditText to display the results, setKeyListner is passed with **'Null'** to seal the fields from edit option.

The **'toggleFieldsVisibility'** is used to set the visibility for the fields. If the location provided by the user exists the fields will be visible else the field will not be visible.

The **'url'** string fetches the location name entered by the user. If the the suer provides the location, it is appended to the **'url1'** and **'url2'** to form the complete URL for the weather API. The URL is then passed to the **'weatherbean'** object else, the Toast notifies the user to enter the location. By passing the context and Url to the **'FetchWeather()'** constructor, we are getting the data from the weather API. To run the request in the background, the http request is performed in the **' AsyncTask'**. **'progressDialog'** is declared globally for the '**onPreExecute()'** and **'onPostExecute()'**.

'onPreExecute()' displays messages with the help of **'progressDialog'** while requesting the data from the API. **' doInBackground()'** sets up on open connection for **HttpURLConnection**. The buffer is fetching the data with Json object from the API. Now disconnect the connection and assign the data from the buffer to the result after converting the values in the buffer to string type.

'onPostExecute()' dismisses the **'progressDialog'.** The **'if(result!=null)'** condition checks whether the data is successful fetched or not, if yes then we will create an Json object, '**weatherResponseJson'** for the retrieved data in the result which is string type. The result string now contains the Json object mentioned below; this is fetched from the API.

{"coord":{"lon":77.6,"lat":12.98},"sys":{"type":1,"id":7823,"messag
e":0.173,"country":"IN","sunrise":1433895765,"sunset":1433942
118},"weather":[{"id":803,"main":"Clouds","description":"broken
clouds","icon":"04d"}],"base":"stations","main":{"temp":298.67,"p
ressure":1015,"humidity":78,"temp_min":298.15,"temp_max":299.
15},"visibility":8000,"wind":{"speed":6.2,"deg":250},"clouds":{"all
":75},"dt":1433917104,"id":1277333,"name":"Bangalore","cod":2
00}

The above mentioned Json object contained in the flower braces is the format of a Json object and the key elements is coord with values as lon and lat, sys with values type id etc. Now access the data via the '**weatherResponseJson'** object's key, since we cannot access the values directly in Json. **'mainObj'**, JSon object is used to access values from the concerned keys.

Now the fetched values are set to the fields in the UI to display the result. In case the if condition fails the else block is executed to show a Toast message, **"Could not fetch weather for the city."** It also sets the **toggleFieldsVisibility()** as False. All this is done in try{} and catch() block to handle the exception while fetching the value sing the Json object.

Weatherbean.Java

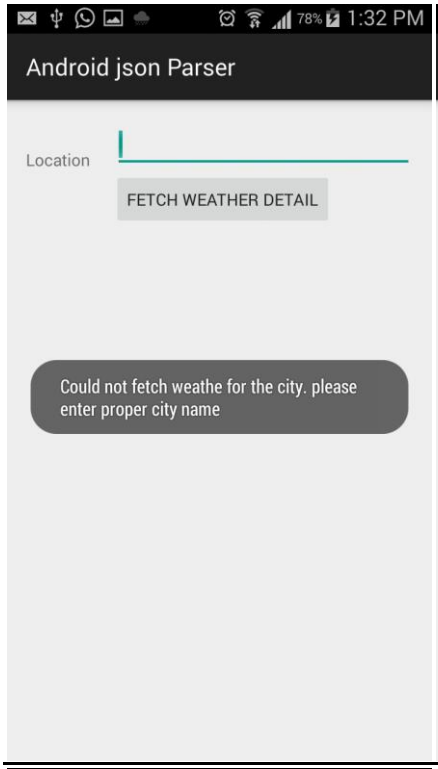
This is a getter and setter file to fetch the result and set it to the concerned EditText to display the result in the UI. The **'weatherObj'** gets the values from the MainActivity.Java and the retrieved values are assigned to the EditText against the concerned Labels.

```
public class Weatherbean {

private String country = "county";
```

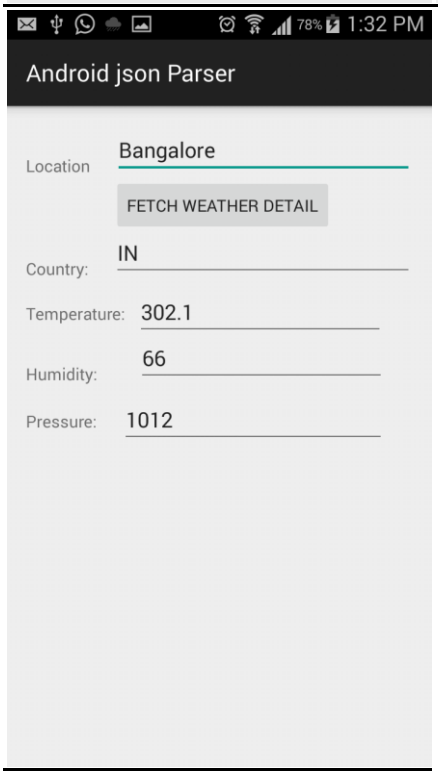
```
   private String temperature = "temperature";
   private String humidity = "humidity";
   private String pressure = "pressure";
   private String urlString = null;

   private boolean gotResponse ;
         public vaWeatherbean(String url){
          this.urlString = url;
   }
   public Weatherbean(){
}
 public String getCountry() {
      return country;
   }

   public boolean isGotResponse() {
      return gotResponse;
   }
 public void setGotResponse(boolean gotResponse) {
      this.gotResponse = gotResponse;
   }
 public String getTemperature() {
return temperature;
   }
```

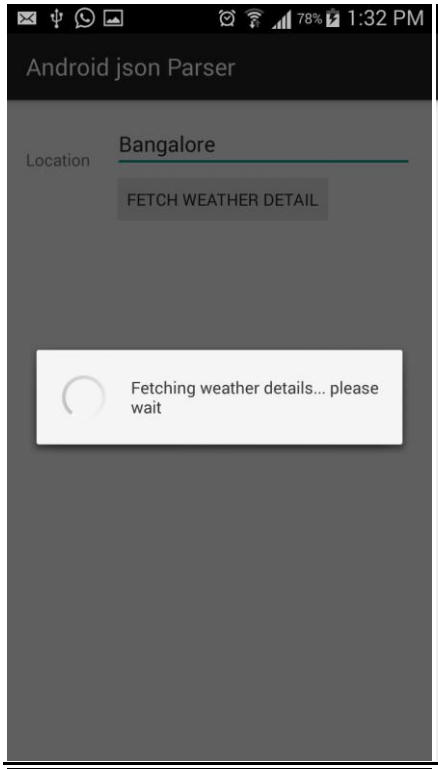
```
    public String getHumidity() {
        return humidity;
    }
public String getUrlString() {
        return urlString;
    }
 public String getPressure() {
        return pressure;
    }
    public void setCountry(String country) {
        this.country = country;
    }
    public void setTemperature(String temperature) {
        this.temperature = temperature;
    }


    public void setHumidity(String humidity) {
        this.humidity = humidity;
    }
public void setPressure(String pressure) {
        this.pressure = pressure;
    }
}
```

Snapshot for the Application

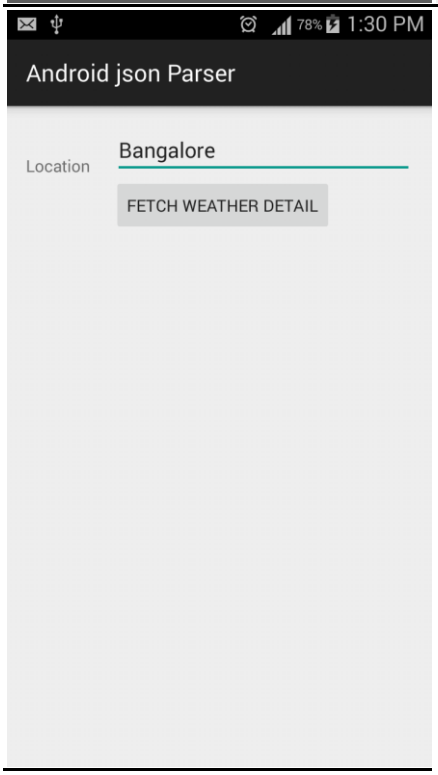

78% 1:32 PM
Android json Parser
Location
FETCH WEATHER DETAIL
Could not fetch weathe for the city. please enter proper city name


78% 1:32 PM
Android json Parser
Location
FETCH WEATHER DETAIL
Could not fetch weathe for the city. please enter proper city name


78% 1:32 PM
Android json Parser
Location
Bangalore
FETCH WEATHER DETAIL
Country:
IN
Temperature:
302.1
Humidity:
66
Pressure:
1012


78% 1:32 PM
Android json Parser
Location
Bangalore
FETCH WEATHER DETAIL
Fetching weather details... please wait


Saving screenshot...
Android json Parser
Location
FETCH WEATHER DETAIL
Could not fetch weathe for the city. please enter proper city name


78% 1:30 PM
Android json Parser
Location
Bangalore
FETCH WEATHER DETAIL