# Android 解析 JSON，你真的需要三方库？

Android 解析 JSON，你真的需要三方库？。一般情况下，如果服务器返回 JSON 数据，而且你又是做 Android 的，那么你首先想到的可能是 GSON，或是 fastJson 这样的框架。这些框架能够很方便和快速的让我们将 JSON 转换成本地对象，是开发的首选。但是引用三方库也是有代价的，显而易见的就是包体积增大，库的升级等。这个时候，就需要想一想我们是不是必须要使用三方库了。

其实在 Android 上处理 JSON ，Google 已经给我们提供了一些类，而这些类满足了大多数简单的需求。如果我们只是需要对 JSON 进行简单的解析处理，那么完全可以避免引入第三方库。

1、org.json

在 Android 的参考文档中，我们能发现一个包，名为：org.json。这个包下主要有四个类：

| 类名 | 描述 |
| --- | --- |
| JSONArray | A dense indexed sequence of values. |
| JSONObject | A modifiable set of name/value mappings. |
| JSONStringer | ImplementstoString()andtoString() |
| JSONTokener | Parses a JSON (RFC 4627) encoded string into the corresponding object. |

关于 org.json 的代码，还可以在 https://github.com/stleary/JSON-java 上找到，而且里面的类远不止这 4 个，里面很多内容都值得去探索一下。但是由于本文仅专注于在 Android 上利用此工具处理 JSON，所以就仅仅讨论着四个类，下面将分别对着四个类逐一讲解。

2、JSONObject

这个类是这四个类中最关键的一个，它表示了一个可更改且无序的键值对集合，更简单一点，可以直接认为这个类表示了一个 JSON 的信息。在 JSON 字符串中，其表示了一个包裹在花括号的字符串，键和值之间使用冒号隔开，键值和键值之间使用逗号隔开的。

在这个类中，其键名是唯一且不为 null 的字符串。而值则可以为 JSONObject、JSONArray、Strings、Booleans、Integers、Longs、Double 或者 JSONObject.NULL。特别注意，这里的 NULL 可不是 null，而是 JSONObject 的一个内部类。

对于这个类，在使用时要要注意的就是在调用时，其会按照调用的方法进行类型转换。下面就介绍一下三类函数：

getXXX() 获取一个值，此方法如果发生失败，例如没有找到对应的键值或类型转换失败，就会抛出一个 JSONException 异常； optXXX() 此类方法也是用于获取一个值，但是如果发生失败，其不会抛出异常，而是返回一个默认值；

put()此类方法就是向对象中插入一个键值对。特别注意其插入 NULL 和 null 是不同的；

刚说到此类中的 NULL，它与 JAVA 中的 null 是不一样的，它仅仅是 JSONObject 中用于标识 null 的对象。举个例子：

put(name, null) 这个方法调用将会移除该对象中对应的键值；

put(name, JSONObject.NULL) 将会往对象中添加一个键值，而其值为 JSONObject.NULL；

下面通过一个实际使用的代码来演示该类：

```
JSONObject jsonObject = new JSONObject("{\"first_name\":\"Taylor\",
\"last_name\":\"swifter\"}");
String firstName = jsonObject.getString("first_name");
String lastName = jsonObject.getString("last_name");
Log.i("swifter", firstName + " " + lastName); //输出 Taylor swifter
jsonObject.put("first_name", "Avril");
Log.i("swifter", jsonObject.toString()); //输出 {"first_name":"Avril",
"last_name":"swifter"}
```

通过代码也可以看到 JSONObject 的简单用法，可以看到这个类也可以用来进行构建 JSON 字符串，只要不断向类中使用 put()方法插入或是修改，在最后使用 toString()就可以得到最终的 JSON 了。

关于这个 toString()这个函数后面还会提到，下面介绍的是在这四个类中第二重要的类：表示数组的 JSONArray。

3、JSONArray

该类表示了 JSON 中的值的数组，可以简单的理解其为一个普通数组，也有 getXXX()和 optXXX()方法，但是基本都需要传入索引值。这个类代表了 JSON 中的一个包裹在方括号的字符串，值和值之间使用逗号隔开的信息。

除此之外，这个类有很多性质都与 JSONObject 一样，比如说类型转换，对于 NULL 和 null 的处理，有 get，put，opt 方法等，所以，只要你熟悉了 JSONObject，那么使用这个类，也会非常容易：

```
JSONArray jsonArray = new JSONArray("[10,11,12,13,14,15,16]");
for(int index = 0; index < jsonArray.length(); index++) {
  Log.i("swifter", index+" : "+jsonArray.getInt(index));
}
```

此段代码的输出是将 10 到 16 这几个数组中的数打印出来。在更多情况下，该类一般都是与 JSONObject 一起使用的，因为普遍的 JSON 都是其中的某一个字段是数组，因此需要先使用 JSONObject 进行解析，然后使用 getJSONArray()再来获取这个数组的信息并进行处理：

```
JSONObject jsonObject = new JSONObject("{\"first_name\":\"Taylor\",
\"last_name\":\"swifter\",
                \"array\":[first, second, third, fourth]}");
JSONArray jsonArray = jsonObject.getJSONArray("array");
for(int index = 0; index < jsonArray.length(); index++) {
  Log.i("swifter", index+" : "+jsonArray.getString(index));
}
```

此段待会会将 JSON 中 array 的四个值以字符串的形式依次打印出来。

4、JSONStringer

此类可以用于快速构建 JSON 文本，它实现了 JSONObject 中的两个 toString()方法，对于大多数程序员来说，应该直接调用 JSONObject 的 toString()方法而忽略这个类的：

JSONObject object = …

String json = object.toString();

它使用了一种类似于 XML 事件解析的方式来实现 JSON 构建，例如它有 key()方法用于插入键，有 value()方法用于插入值，而且还有 array()和 endArray()用于开始和结束插入数组，有 object()和 endObject()方法用于开始和结束插入 JSON 内容。

仅仅说其有什么方法比较涩会难懂，那就上代码：

```
JSONStringer jsonStringer = new JSONStringer();
jsonStringer.object().key("first_name").value("Taylor").key("last_name").
value("swifter");
jsonStringer.key("array").array().value(12).value(13).value(14).endArray()
.endObject();
Log.i("swifter", jsonStringer.toString());   //输出 {"first_name":"Taylor"
,"last_name":"swifter","array":[12,13,14]}
```

可以看到对于 JSON 的构建，该类就是通过这几个方法的调用来实现的。

刚才说到 JSONObject 类的 toString()方法就是通过这个类来实现的，那它是怎么实现的呢？现在就来看看代码吧。它有两个方法，为了简单起见，我们就看其中一个：

```
/**
 * Encodes this object as a compact JSON string, such as:
 *
 {"query":"Pizza","locations":[94043,90210]}
*/ @Override public String toString() { try { JSONStringer stringer =
new JSONStringer(); writeTo(stringer); return stringer.toString(); }
catch (JSONException e) { return null; } } void writeTo(JSONStringer
stringer) throws JSONException { stringer.object(); for
(Map.Entry entry : nameValuePairs.entrySet())
{ stringer.key(entry.getKey()).value(entry.getValue()); }
stringer.endObject(); }
```

这段代码显示了 JSONObject 是如何处理 toString()的，其主要的工作在 writeTo()这个函数中，而又可以看到，这个函数也是调用的 JSONStringer 的 key()和 value()等方法来实现 JSON 构造的。另一个 toString()也类似，仅仅是是用了 JSONStringer 的另一个构造函数而已。

JSONTokener

这个类是四个类中最不常用的一个类，它能够将一个 JSON 字符串（RFC 4627）解析到相关的对象，对于大多数客户端仅仅需要使用它的构造函数和 nextValue()函数：

```
String json = "{"
    + "  \"query\": \"Pizza\", "
    + "  \"locations\": [ 94043, 90210 ] "
    + "}";
JSONObject object = (JSONObject) new JSONTokener(json).nextValue();
String query = object.getString("query");
JSONArray locations = object.getJSONArray("locations");
```

可见，这个类就是和 JSONObject 的构造函数一起使用的，用于解析 JSON 源字符串，还例如：

```
JSONObject jsonobj = new JSONObject(new JSONTokener(new FileReader
(new File("json.txt"))));
```

上面的四个类的内容都如此吧。相比于其他第三方库，内置的 JSON 解析功能上确实非常简单，但在很多场景下也是完全够用的。当熟悉了这几个类之后，我们也就可以权衡一下是否真的需要引用第三方库才能完成需求了。如果能满足的话，为什么还需要引用其他库呢。