

Arduino Json 库使用教程

By 小橘子豆

今天为了完善手里 ESP8266 开发板和乐联网的数据通信，折腾了好久 POST 和 GET。相比来讲，对于任何物联网 IOT 来说，都免不了云平台的 json 字符解析。好在 Arduino 资源多，在 github 上找到了关于 Json 解析的库。今天就对这个库进行下介绍。

Json 库下载地址

<https://github.com/bblanchon/ArduinoJson>

下载之后按照安装要求解压到安装目录下的 **libraries** 目录，

| | | | |
|-------------------|------------------|-------|--------|
| dist | 2015/12/17 16:23 | 文件夹 | |
| drivers | 2015/12/17 16:23 | 文件夹 | |
| examples | 2015/12/17 16:23 | 文件夹 | |
| hardware | 2016/1/21 20:04 | 文件夹 | |
| java | 2015/12/17 16:24 | 文件夹 | |
| lib | 2016/1/21 22:32 | 文件夹 | |
| libraries | 2016/2/3 11:34 | 文件夹 | |
| portable | 2016/2/3 11:36 | 文件夹 | |
| reference | 2015/12/17 16:23 | 文件夹 | |
| tools | 2015/12/17 16:23 | 文件夹 | |
| tools-builder | 2015/12/17 16:23 | 文件夹 | |
| 8266_SDK_1.5.1.md | 2016/1/21 20:33 | MD 文件 | 0 KB |
| arduino.exe | 2015/12/17 16:23 | 应用程序 | 853 KB |
| arduino.exe | 2016/1/21 22:30 | 快捷方式 | 1 KB |
| arduino.l4j.ini | 2015/12/17 16:23 | 配置设置 | 1 KB |
| arduino debug.exe | 2015/12/17 16:23 | 应用程序 | 390 KB |

该目录下有如下的文件

| 名称 | 修改日期 | 类型 | 大小 |
|---------------|------------------|-----|----|
| ArduinoJson | 2016/2/3 11:34 | 文件夹 | |
| Bridge | 2015/11/19 16:33 | 文件夹 | |
| Esplora | 2015/5/20 17:10 | 文件夹 | |
| Ethernet | 2015/12/17 16:23 | 文件夹 | |
| Firmata | 2015/11/8 20:46 | 文件夹 | |
| GSM | 2015/12/17 16:23 | 文件夹 | |
| Keyboard | 2015/10/12 12:24 | 文件夹 | |
| LiquidCrystal | 2015/12/17 16:23 | 文件夹 | |
| Mouse | 2015/10/12 12:24 | 文件夹 | |
| Robot_Control | 2015/6/10 15:00 | 文件夹 | |
| Robot_Motor | 2015/6/10 15:00 | 文件夹 | |
| RobotIRremote | 2015/6/10 15:00 | 文件夹 | |
| SD | 2015/12/17 16:23 | 文件夹 | |
| Servo | 2015/12/17 16:23 | 文件夹 | |
| SpacebrewYun | 2015/3/27 15:01 | 文件夹 | |
| Stepper | 2015/12/17 16:23 | 文件夹 | |
| Temboo | 2015/10/14 21:55 | 文件夹 | |
| TFT | 2015/12/17 16:23 | 文件夹 | |
| WiFi | 2015/12/17 16:23 | 文件夹 | |

把 Arduino Json 解压到这里。

之后重启 **arduino IDE**，就能在示例中看到了。

下面介绍示例：

打开乐联网的 API 手册，我们能发现更新数据点的 POST 格式是这样的。

```
[ { "Name": "T1", "Value": "1" }, { "Name": "01H1", "Value": "96.2" } ]
```

这里要介绍下，这种格式的生成方法。

首先，方括号[] 在库中被称作 array，也就是数组，而大括号{}被称为 Object，也就是对象。

在如上的 json 中，我们分析的是，就要先创建一个 array，这个 array 中又包含了两个 object，那么代码如下：

```
#include <ArduinoJson.h>    包含头文件
```

```
StaticJsonBuffer<200> jsonBuffer;  设置静态 buffer
```

```
JsonObject& root = jsonBuffer.createArray();  创建根，也就是最外边的 array。
```

如果你的数据最外边是{}，那么你就要用

```
JsonObject& root = jsonBuffer.createObject();
```

```
JsonObject& son1= root.createNestedObject();  设置下级的 array 或者 object。这里可以通过 root.createNestedObject()创建 object 或者 root.createNestedArray()创建 array；
```

```
JsonObject& son2= root.createNestedObject();  创建两个 object
```

接下来为两个 object 设置内容

```
son1["name"]="H1";
```

```
son1["value"]="566";
```

```
son2["name"]="H2";
```

```
son2["value"]="466";
```

显示整个 JSON

```
root.printTo(Serial); //此处为打印到一行中
```

```
root.prettyPrintTo(Serial);此处为分行打印
```

运行结果（忽略开头的乱码）

```
裕毒vAM禧 4C8 [{"name":"H1","value":"566"}, {"name":"H2","value":"466"}]
[
  {
    "name": "H1",
    "value": "566"
  },
  {
    "name": "H2",
    "value": "466"
  }
]
```

接下来说下 Json 的解码

既然编码理解了，大家就知道如何解码了。打开示例

初始化的部分就不在提示了，直接进入正题

```
{
  "Data": [
    {
      "updateTime": "2016/2/3 10:33:24",
      "value": 5
    }
  ],
  "Successful": true,
  "Message": null
}
```

解析的字符串用这个即可。使用方法很简单，

```
JsonObject& root = jsonBuffer.parseObject(json);
```

```
JsonObject& root = jsonBuffer.parseArray(json);
```

这两个方法当然对应了开头是{}和[]的情况。

先说第一种，解析 Object，

```
if (!root.success()) { //判断是否解析成功
    Serial.println("parseObject() failed");
    return;
}
```

成功之后执行

```
const char* sensor = root["Successful"];
```

```
const char* updateTime = root["Data"][0]["value"];
```

```
const char* updateTime1 = root["Data"][0]["updateTime"];
```

我觉得第一个大家都能理解，但是第二个和第三个怎么解释呢？我们回到要解析的字符串，发现一个名叫 data 的 object 下面包含一个 array，这个 array 里有两个 object。所以索引时，第二个方括号的内容要写 0 1 2 3 等 array 的索引编号。第三个参数写最里层的名称。

对于解析 array 也是相同的道理。在此不再解释了。

最后提醒大家注意，这个 `JsonObject& root = jsonBuffer.parseObject(json);` 函数中，`json` 要求输入的是 `char *类型`。而如果你的是 `string` 类型，那么需要通过 `toCharArray()` 的函数进行转换，而不能使用 `c_str()`，因为 `c_str ()` 返回的是 `const` 类型