<https://www.cnblogs.com/likui360/p/6133005.html>

# [C++处理Json串——jsoncpp库](https://www.cnblogs.com/likui360/p/6133005.html)

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式，和xml类似，本文主要对VS2008中使用Jsoncpp解析json的方法做一下记录。  
Jsoncpp是个跨平台的开源库，下载地址：<http://sourceforge.net/projects/jsoncpp/>。

**方法一：使用Jsoncpp生成的lib文件**

      解压上面下载的Jsoncpp文件，在jsoncpp-src-0.5.0/makefiles/vs71目录里找到jsoncpp.sln，用VS2008版本编译，默认生成静态链接库。 在工程中引用，只需要包含include/json下的头文件及生成的.lib文件即可。  
      如何包含lib文件：在.cpp文件中#pragma comment(lib."json\_vc71\_libmt.lib")，在工程属性中Linker下Input中Additional Dependencies写入lib文件名字（Release下为json\_vc71\_libmt.lib，Debug为json\_vc71\_libmtd.lib）

注意：Jsoncpp的lib工程编译选项要和VS工程中的编译选项保持一致。如lib文件工程编译选项为MT（或MTd），VS工程中也要选择MT（或MTd），否则会出现编译错误问题，debug和release下生成的lib文件名字不同，注意不要看错了，当成一个文件来使用（我就犯了这个错误）。  
  
**方法二：使用Jsoncpp包中的.cpp和.h文件**  
      解压上面下载的Jsoncpp文件，把jsoncpp-src-0.5.0文件拷贝到工程目录下，将jsoncpp-src-0.5.0\jsoncpp-src-0.5.0\include\json和jsoncpp-src-0.5.0\jsoncpp-src-0.5.0\src\lib\_json目录里的文件包含到VS工程中，在VS工程的属性C/C++下General中Additional Include Directories包含头文件目录.\jsoncpp-src-0.5.0\include。在使用的cpp文件中包含json头文件即可，如：#include "json/json.h"。将json\_reader.cpp、json\_value.cpp和json\_writer.cpp三个文件的Precompiled Header属性设置为**Not Using Precompiled Headers**，否则编译会出现错误。

**jsoncpp 使用详解**

jsoncpp 主要包含三种类型的 class：Value、Reader、Writer。jsoncpp 中所有对象、类名都在 namespace Json 中，包含 json.h 即可。

Json::Value 只能处理 ANSI 类型的字符串，如果 C++ 程序是用 Unicode 编码的，最好加一个 Adapt 类来适配。

下面是从网上找的代码示例：  
1. 从字符串解析json

[复制代码](javascript:void(0);)

复制代码

const char\* str = "{\"uploadid\": \"UP000000\",\"code\": 100,\"msg\": \"\",\"files\": \"\"}";

Json::Reader reader;

Json::Value root;

if (reader.parse(str, root)) // reader将Json字符串解析到root，root将包含Json里所有子元素

{

std::string upload\_id = root["uploadid"].asString(); // 访问节点，upload\_id = "UP000000"

int code = root["code"].asInt(); // 访问节点，code = 100

}

复制代码

[复制代码](javascript:void(0);)

2. 从文件解析json

[复制代码](javascript:void(0);)

复制代码

int ReadJsonFromFile(const char\* filename)

{

Json::Reader reader;// 解析json用Json::Reader

Json::Value root; // Json::Value是一种很重要的类型，可以代表任意类型。如int, string, object, array

std::ifstream is;

is.open (filename, std::ios::binary );

if (reader.parse(is, root, FALSE))

{

std::string code;

if (!root["files"].isNull()) // 访问节点，Access an object value by name, create a null member if it does not exist.

code = root["uploadid"].asString();

code = root.get("uploadid", "null").asString();// 访问节点，Return the member named key if it exist, defaultValue otherwise.

int file\_size = root["files"].size(); // 得到"files"的数组个数

for(int i = 0; i < file\_size; ++i) // 遍历数组

{

Json::Value val\_image = root["files"][i]["images"];

int image\_size = val\_image.size();

for(int j = 0; j < image\_size; ++j)

{

std::string type = val\_image[j]["type"].asString();

std::string url = val\_image[j]["url"].asString();

printf("type : %s, url : %s \n", type.c\_str(), url.c\_str());

}

}

}

is.close();

return 0;

}

复制代码

[复制代码](javascript:void(0);)

3. 向文件中插入json

[复制代码](javascript:void(0);)

复制代码

void WriteJsonData(const char\* filename)

{

Json::Reader reader;

Json::Value root; // Json::Value是一种很重要的类型，可以代表任意类型。如int, string, object, array

std::ifstream is;

is.open (filename, std::ios::binary );

if (reader.parse(is, root))

{

Json::Value arrayObj; // 构建对象

Json::Value new\_item, new\_item1;

new\_item["date"] = "2011-11-11";

new\_item1["time"] = "11:11:11";

arrayObj.append(new\_item); // 插入数组成员

arrayObj.append(new\_item1); // 插入数组成员

int file\_size = root["files"].size();

for(int i = 0; i < file\_size; ++i)

root["files"][i]["exifs"] = arrayObj; // 插入原json中

std::string out = root.toStyledString();

// 输出无格式json字符串

Json::FastWriter writer;

std::string strWrite = writer.write(root);

std::ofstream ofs;

ofs.open("test\_write.json");

ofs << strWrite;

ofs.close();

}

is.close();

}

复制代码

[复制代码](javascript:void(0);)

 4.序列化json字符串

[复制代码](javascript:void(0);)

复制代码

先构建一个Json对象，此Json对象中含有数组，然后把Json对象序列化成字符串，代码如下：

Json::Value root;

Json::Value arrayObj;

Json::Value item;

for (int i=0; i<10; i++)

{

　　item["key"] = i;

　　arrayObj.append(item);

}

root["key1"] = “value1″;

root["key2"] = “value2″;

root["array"] = arrayObj;

root.toStyledString();

std::string out = root.toStyledString();

std::cout << out << std::endl;

复制代码

[复制代码](javascript:void(0);)

5.反序列化json

[复制代码](javascript:void(0);)

复制代码

比如一个Json对象的字符串序列如下,其中”array”:[...]表示Json对象中的数组：

{“key1″:”value1″,”array”:[{"key2":"value2"},{"key2":"value3"},{"key2":"value4"}]}，那怎么分别取到key1和key2的值呢，代码如下所示:

std::string strValue = “{\”key1\”:\”value1\”,\”array\”:[{\"key2\":\"value2\"},{\"key2\":\"value3\"},{\"key2\":\"value4\"}]}”;

Json::Reader reader;

Json::Value value;

if (reader.parse(strValue, value))

{

　　std::string out = value["key1"].asString();

　　std::cout << out << std::endl;

　　const Json::Value arrayObj = value["array"];

　　for (int i=0; i<arrayObj.size(); i++)

　　{

　　　　out = arrayObj[i]["key2"].asString();

　　　　std::cout << out;

　　　　if (i != arrayObj.size() – 1 )

　　　　　　std::cout << std::endl;

　　}

}

复制代码

[复制代码](javascript:void(0);)

6.删除json子对象

[复制代码](javascript:void(0);)

复制代码

std::string strContent = "{\"key\":\"1\",\"name\":\"test\"}";

Json::Reader reader;

Json::Value value;

if (reader.parse(strContent, value))

{

Json::Value root=value;

root.removeMember("key");

printf("%s \n",root.toStyledString().c\_str());  
　　}

复制代码

[复制代码](javascript:void(0);)

 7. 利用jsoncpp将json字符串转换为Vector

在API测试过程中经常会遇到传入参数为复杂类型，一般情况下在python下，习惯用字典来表示复杂类型。但是c++对字符串的处理是比较弱智的，一般c++里边会用vector来存储复杂类型，那么就存在转换的问题，下面小段代码记录了将字符串转换为Vector的过程

待转换的字符串如下：

[复制代码](javascript:void(0);)

复制代码

const char \* jsongroupinfo="[{/"groupId/" :946838524,/"groupname/" :/"bababa/", /"mask/":1,/"parentid/":946755072}]";

Json::Reader reader;

Json::Value json\_object;

if (!reader.parse(jsongroupinfo, json\_object))

　　return "parse jsonstr error";

SUserChggroup sucg;

VECTOR< SUserChggroup > m\_groupInfo;

for(int i = 0; i < json\_object.size(); i ++)

{

　　Json::Value &current = json\_object[i];

　　sucg.m\_groupId = current["groupId"].asInt();

　　sucg.m\_groupName = current["groupname"].asString();

　　sucg.m\_mask = current["mask"].asInt();

　　sucg.m\_parentId = current["parentid"].asInt();

　　m\_groupInfo.push\_back(sucg);

}

复制代码

[复制代码](javascript:void(0);)

 简而言之，就是把它变成解析成一个个对象，再将对象存储到vector中。