**[如何使用rapidjson库进行json格式的封装及解析](http://blog.csdn.net/wangqing_12345/article/details/52353593)**

2016-08-29 15:34 7339人阅读 [评论](http://blog.csdn.net/wangqing_12345/article/details/52353593#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/wangqing_12345/article/details/52353593#report)

版权声明：本文为博主原创文章，未经博主允许不得转载。

一，字符串json封装及及解析的实例

#include "rapidjson/document.h"

#include "rapidjson/prettywriter.h"    
#include "rapidjson/writer.h"  
#include "rapidjson/stringbuffer.h"  
#include <iostream>  
using namespace rapidjson;  
using namespace std;  
int main()   
{  
  
//一，转json格式  
    //1,获取Document对象  
    Document doc;    
    doc.SetObject();    //key-value 相当与map  
    //doc.Setvalue();        //数组型 相当与vector  
    Document::AllocatorType &allocator=doc.GetAllocator(); //获取分配器  
  
    //2，给doc对象赋值  
    doc.AddMember("name","张山",allocator);  
      
    //添加数组型数据  
    Value array1(kArrayType);  
    for(int i=0;i<3;i++)    
    {    
        Value int\_object(kObjectType);    
        int\_object.SetInt(i);    
        array1.PushBack(int\_object,allocator);    
    }  
      
    doc.AddMember("number",array1,allocator);  
  
    //3，将doc对象的值写入字符串  
    StringBuffer buffer;    
    //PrettyWriter<StringBuffer> writer(buffer);  //PrettyWriter是格式化的json，如果是Writer则是换行空格压缩后的json    
    Writer<StringBuffer> writer(buffer);  
    doc.Accept(writer);   
  
    cout<<buffer.GetString()<<endl;  
  
//二，解析json格式  
    //1，将json格式字符串转换  
    string readdate;  
    readdate = buffer.GetString();  
    Document document;    
    document.Parse<0>(readdate.c\_str());    
      
    //2,取出自己想要的值  
    Value &node1=document["name"];    
    cout<<"name:"<<node1.GetString()<<endl;   
  
    Value &node2=document["number"];    
    cout<<"number: "<<endl;    
    if(node2.IsArray())    
    {    
        for(int i=0;i<node2.Size();i++)    
            cout<<'\t'<<node2[i].GetInt()<<endl;    
    }    
  
    return 0;

}

二，文件json封装及解析的实例

**1 两个问题**

（1）标准json和非标准json：

标准json要求键必须都是双引号的字符串，而非标准json可以单引号。

例如：

{a : 'abc'}

{'a' : 'abc'}

{a : "abc"}

{"a" : "abc"}

只有第4个是标准json

（2）json中的[]与{}:

在 JSON 里 [] 是 Array {} 是Ojbect   
  
[] Array 的key 是 int  从0算起  
{} 的key 是 string   
  
var a= Array();   
a[a.length] = '3';   
a[a.length] = '4';   
a[a.length] = '5';   
  
a toJSON 后 ='["3", "4", "5"]'   
  
var a = new Object();   
  
a['test1'] = '3';   
a['test2'] = '4';   
a['test3'] = '5';   
  
a toJSON 后 = '{"test1":"3", "test2":"4", "test3":"5"}'

**2 rapidjson读写测试**

下载rapidjson库，解压后关联到工程。

代码：

**[cpp]** [view plain](http://blog.csdn.net/u012234115/article/details/42295665) [copy](http://blog.csdn.net/u012234115/article/details/42295665)

[print?](http://blog.csdn.net/u012234115/article/details/42295665)

1. #include <iostream>
2. #include <string>
3. #include <fstream>
4. //包含rapidjson必要头文件,rapidjson文件夹拷贝到工程目录，或者设置include路径，或者加入到工程树
5. #include "rapidjson/document.h"
6. #include "rapidjson/filestream.h"
7. #include "rapidjson/prettywriter.h"
8. #include "rapidjson/stringbuffer.h"
9. using namespace std;
10. using namespace rapidjson;  //引入rapidjson命名空间
12. //写json文件
13. void json\_write()
14. {
15. Document doc;
16. doc.SetObject();
17. Document::AllocatorType &allocator=doc.GetAllocator(); //获取分配器
18. //1.添加字符串对象
19. doc.AddMember("author","tashaxing",allocator);
20. //2.添加数组对象
21. Value array1(kArrayType);
22. for(**int** i=0;i<3;i++)
23. {
24. Value int\_object(kObjectType);
25. int\_object.SetInt(i);
26. array1.PushBack(int\_object,allocator);
27. }
28. doc.AddMember("number",array1,allocator);
29. //3.添加复合对象
30. Value object(kObjectType);
31. object.AddMember("language1","C++",allocator);
32. object.AddMember("language2","java",allocator);
33. doc.AddMember("language",object,allocator);
34. //4.添加对象数组和复合对象的组合
35. Value array2(kArrayType);
36. Value object1(kObjectType);
37. object1.AddMember("hobby","drawing",allocator);
38. array2.PushBack(object1,allocator);
39. Value object2(kObjectType);
40. object2.AddMember("height",1.71,allocator);
41. array2.PushBack(object2,allocator);
42. doc.AddMember("information",array2,allocator);
43. StringBuffer buffer;
44. PrettyWriter<StringBuffer> pretty\_writer(buffer);  //PrettyWriter是格式化的json，如果是Writer则是换行空格压缩后的json
45. doc.Accept(pretty\_writer);
46. //打印到屏幕
47. cout<<"the json output:"<<endl;
48. cout<<buffer.GetString()<<endl;
49. //输出到文件
50. ofstream fout;
51. fout.open("test");    //可以使绝对和相对路径，用\\隔开目录，test, test.json, test.txt 都行，不局限于文件格式后缀，只要是文本文档
52. fout<<buffer.GetString();
53. fout.close();
54. }
56. //读json文件
57. void json\_read()
58. {
59. cout<<"the json read:"<<endl;
60. ifstream fin;
61. fin.open("test");
62. string str;
63. string str\_in="";
64. while(getline(fin,str))    //一行一行地读到字符串str\_in中
65. {
66. str\_in=str\_in+str+'\n';
67. }
68. //解析并打印出来
69. Document document;
70. document.Parse<0>(str\_in.c\_str());
72. Value &node1=document["author"];
73. cout<<"author: "<<node1.GetString()<<endl;
75. Value &node2=document["number"];
76. cout<<"number: "<<endl;
77. if(node2.IsArray())
78. {
79. for(**int** i=0;i<node2.Size();i++)
80. cout<<'\t'<<node2[i].GetInt()<<endl;
81. }
83. Value &node3=document["language"];
84. cout<<"language: "<<endl;
85. Value &tmp=node3["language1"];
86. cout<<'\t'<<"language1: "<<tmp.GetString()<<endl;
87. tmp=node3["language2"];
88. cout<<'\t'<<"language2: "<<tmp.GetString()<<endl;
90. Value &node4=document["information"];
91. cout<<"information: "<<endl;
92. if(node4.IsArray())
93. {
94. **int** i=0;
95. Value &data=node4[i];   //注意，此处下表索引只能用变量，不能用常量，例如node[0]编译错误
96. cout<<'\t'<<"hobby: "<<data["hobby"].GetString()<<endl;
97. i=1;
98. data=node4[i];
99. cout<<'\t'<<"height: "<<data["height"].GetDouble()<<endl;
100. }
102. }
103. **int** main(**int** argc,**char** \*\*argv)
104. {
105. //写、读 测试
106. json\_write();
107. json\_read();
108. return 0;
109. }