RapidJson 解析（1）

DionysosLai([906391500@qq.com](mailto:906391500@qq.com)) 2014/9/20

Json 是一种轻量级数据交换格式，具有易于人阅读和编写，同时也易于机器解析和生成。相较于XML，json更小、读写更快、更易解析。另一方面，Rapidjson作为json的升级版，在效率方面，具有更好的优势。

**Json语法规则**

1. 数据在名称/值对中

Json数据的书写格式: 名称/值对。

名称/值对包括字段名称（在双引号中），后面跟一个冒号，然后就是值：

例如：

“subject” : ”English”

Json值可以是：数字（整数或浮点数）、字符串（在双引号中）、逻辑值（true或false）、数组（在方括号中）、对象（在花括号中）、null

Json对象中可以包含多个名称/值对：

例如：

{“subject” : ”English”， “subject” : ”Math”}

Json数组可以包含对象：

例如：

{“student”:[

{“subject” : ”English”， “subject” : ”Math”},

{“subject” : ”Yuwen”， “subject” : ”Music”},

]

}

1. 数据有逗号隔开

这里要注意的是最后一个数据不要加逗号，

1. 花括号用来保存对象
2. 方括号用来保存数组

**Rapdijson解析**

Json本身是JavaScript对象表示法（JavaScript Ojbect Notation），在javascript层面很容易解析。对于c++，必须导入相关库文件。Cocos2dx从2.1版本后，引入了rapidjson库，因此这里是RapidJson解析。后面讲到的内容也是基于cocos2dx引擎（这里版本为：cocos2d-x-3.2rc0）

在json解析之前，要先了解json几个相关概念：

Value：value其实就是var，对于value可以理解为int，也是理解为string，或者是bool型变量等其他数据类型。对于定义Value value，只是一个定义，还没有决定其数据类型，如果明确value的值，则相应确定其数据类型了。

Json数据类型是一个map，表示为key-value形式，对于Value转换为基础数据类型有

一下几种方法：

  vall.SetArray() vall.SetArrayRaw() vall.SetBool() vall.SetDouble() vall.SetInt()

vall.SetNull() vall.SetObject() vall.SetString() vall.SetStringRaw() vall.SetUint();

vall.SetUint64()

同时，对于value的数据类型，是可以重复设置。

Write:将Value数据编码成json合适数据格式；

Reader：与Writer相反，是将json格式数据解析成一个Value值。

Json::Reader reader;

1. **数据解析**

首先，这里有一个待解析的json文件：“test.json”，这里注意，json文件格式一般以“josn

”为后缀。同时json文件的编码方式为：UTF-8无BOM格式。“test.json”文件，内容如下：

{

"hello": "world",

"t": true,

"f": false,

"n": null,

"i": 123,

"pi": 3.1416,

"a": [

1,

2,

3,

4

]

}

这里包含了常用的几种数据格式：string、bool、null、int等。

**数据读取、解析成json格式**

数据读取，不管对象时字符串还是文件形式，或者其他形式，最终表示成可读写的字符串格式即可。如果是文件形式，则是标准的读取文件内容。

读取文件数据：

SSIZE\_T size;

unsigned char\* ch = FileUtils::getInstance()->getFileData("test.json","r", &size);

std::string data = std::string((const char\* )ch, size);

这里要注意一点是，这样写不可：

std::string data = (const char\* )CCFileUtils::sharedFileUtils()->getFileData("DataTestQu.json", "r", &size); ///< 读取json文件

这是由于，格式转换发生错误。

下一步就是解析成json格式数据：

Document doc; ///< 创建一个Document对象 rapidJson的相关操作都在Document类中

doc.Parse<0>(data.c\_str()); ///< 通过Parse方法将Json数据解析出来

if (doc.HasParseError())

{

CCLOG("GetParseError %s\n",doc.GetParseError());

}

这里要注意一点就是一定要对解析出来的document（JSON解析出来以xml dom形式存在）进行判断，判断是否解析正确，否则后面一切处理均是无效的。

**Json数据读取和更改-----对值操作**

对于数据读取和值更改，基本思路：通过value读取键的值，判断键值类型，根据键值类型，采用对应方法进行输出和值更改操作。相关代码如下：

rapidjson::Value& valString = doc["hello"]; ///< 读取键“hello”的值，根据我们的json文档，是一个字符串

if (valString.IsString()) ///< 判断是否是字符串

{

const char\* ch = valString.GetString();

log(ch);

log(valString.GetString());

valString.SetString("newString");

log(valString.GetString());

}

rapidjson::Value& valArray = doc["a"]; ///< 读取键“a”值，根据我们的json文档，是一个数组

if (valArray.IsArray()) ///< 判断val的类型 是否为数组 我们的Tollgate键对应的value实际为数组

{

for (int i = 0; i < valArray.Capacity(); ++i)

{

rapidjson::Value& first = valArray[i]; ///< 获取到val中的第i个元素 根据我们这里的json文件 val中共有4个元素

CCLOG("%f", first.GetDouble()); ///< 将value转换成Double类型打印出来 结果为0.5

first.SetDouble(10.f);

CCLOG("%f", first.GetDouble()); ///< 将value转换成Double类型打印出来 结果为0.5S

}

}

**Json数据操作----对键操作**

1. **添加成员对象**

对于一个成员对象，必然是key-value格式。因此，要明确key和value的值。添加一个成员对象，原理是在最初json解析的dom数据下，分配成员变量的空间，然后将成员对象添加进来即可。

添加字符串对象、null对象和数组对象如下：

/// 添加一个String对象；

rapidjson::Document::AllocatorType& allocator = doc.GetAllocator(); ///< 获取最初数据的分配器

rapidjson::Value strObject(rapidjson::kStringType); ///< 添加字符串方法1

strObject.SetString("love");

doc.AddMember("hello1", strObject, allocator);

/\* doc.AddMember("hello1", "love you", allocator); ///< 添加字符串方法2:往分配器中添加一个对象\*/

/// 添加一个null对象

rapidjson::Value nullObject(rapidjson::kNullType);

doc.AddMember("null", nullObject, allocator); ///< 往分配器中添加一个对象

/// 添加一个数组对象

rapidjson::Value array(rapidjson::kArrayType); ///< 创建一个数组对象

rapidjson::Value object(rapidjson::kObjectType); ///< 创建数组里面对象。

object.AddMember("id", 1, allocator);

object.AddMember("name", "lai", allocator);

object.AddMember("age", "12", allocator);

object.AddMember("low", true, allocator);

array.PushBack(object, allocator);

doc.AddMember("player", array, allocator); ///< 将上述的数组内容添加到一个名为“player”的数组中

/// 在已有的数组中添加一个成员对象

rapidjson::Value& aArray1 = doc["a"];

aArray1.PushBack(2.0, allocator);

1. **更改键即key的名称**

**这个不会**

1. **删除成员对象**

成员对象删除方法有一个统一的方法：RemoveMember，对于一个数组对象，如果要删除最后一个元素，可以采用popBack方法;。示例如下：

/// 删除数组成员对象里面元素

rapidjson::Value& aArray2 = doc["a"]; ///< 读取键“a”值，根据我们的json文档，是一个数组

aArray2.PopBack(); ///< 删除数组最后一个成员对象

if (doc.RemoveMember("i")) ///< 删除键为“i”的成员变量

{

log("delet i member ok!");

}

**Json数据写入文件**

这里调用了前面所讲到的Write接口，将Value数据编码成json合适数据格式，在将数据写到文件中即可。示例如下：

/// 将json数据重新写入文件中---先将文件删除，再写入内容

rapidjson::StringBuffer buffer;

rapidjson::Writer<rapidjson::StringBuffer> writer(buffer);

doc.Accept(writer);

#if (CC\_TARGET\_PLATFORM == CC\_PLATFORM\_WIN32)

system("del E:\cocos2d-x-3.2rc0\tests\cpp-empty-test\Resources\test.josn"); ///< 先将文件删除掉---之前重这个文件读取数据，因此确保这个文件存在了

FILE\* file = fopen("test.json", "wb");

if (file)

{

fputs(buffer.GetString(), file);

fclose(file);

}

#else if(CC\_TARGET\_PLATFORM == CC\_PLATFORM\_ANDROID)

/// 原理差不多，就是先将文件清空，在写入。这里就不写了。

#endif

至此，“RapidJson 解析（1）”一文就此结束了。差不多涉及到基本json的所有操作，从数据解析、键值更改、数据重新写入等。最后一个问题是“更改键名称”，目前问题没有找到很好的方法，不知道谁有好的思路可以@我，不胜感激！