Json 的用法

目录

1. 什么是json

- JSON 指的是 JavaScript 对象表示法（JavaScript Object Notation）
- JSON 是轻量级的文本数据交换格式
- JSON 独立于语言 *
- JSON 具有自我描述性，更易理解
- JSON 使用 JavaScript 语法来描述数据对象，但是 JSON 仍然独立于语言和平台。JSON 解析器和 JSON 库支持许多不同的编程语言。

2. Json 数据的格式

JSON 对象在花括号中书写：对象可以包含多个名称/值对：

{ "firstName":"John" , "lastName":"Doe" }

Json 数组：

```
{
"employees": [
{ "firstName":"John" , "lastName":"Doe" },
{ "firstName":"Anna" , "lastName":"Smith" },
{ "firstName":"Peter" , "lastName":"Jones" }
]
}
```

3. JSON 的类

- Json::Value 可以表示里所有的类型，比如 int,string,object,array 等，具体应用将会在后边示例中介绍。
- Json::Reader 将 json 文件流或字符串解析到 Json::Value，主要函数有 Parse。
- Json::Writer 与 Json::Reader 相反，将 Json::Value 转化成字符串流，

注意它的两个子类：Json::FastWriter 和 Json::StyleWriter，分别输出不带格式的 json 和带格式的 json。

4. 解析 json

4.1 从字符串解析

```
int ParseJsonFromString()
{
  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
  }
  return 0;
}
```

4.2 从文件解析 json

json 文件内容：

```
{
    "uploadid": "UP000000",
    "code": "0",
    "msg": "",
    "files":
    [
        {
            "code": "0",
            "msg": "",
            "filename": "1D_16-35_1.jpg",
            "filesize": "196690",
            "width": "1024",
            "height": "682",
            "images":
            [
                {
```

```
                "url": "fmn061/20111118",
                "type": "large",
                "width": "720",
                "height": "479"
            },
            {
                "url": "fmn061/20111118",
                "type": "main",
                "width": "200",
                "height": "133"
            }
        ]
    }
]
}
```

解析代码

```
int ParseJsonFromFile(const char* filename)
{
  // 解析json用Json::Reader
  Json::Reader reader;
  // Json::Value是一种很重要的类型，可以代表任意类型。如int, string, object, array...
  Json::Value root;

  std::ifstream is;
  is.open (filename, std::ios::binary );
  if (reader.parse(is, root))
  {
    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();

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

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

    // 遍历数组
    for(int i = 0; i < file_size; ++i)
```

```
    {
      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();
      }
    }
  }
  is.close();
  return 0;
}
```

4.3 在 json 结构中插入 json

```
Json::Value arrayObj;   // 构建对象
Json::Value new_item, new_item1;
new_item["date"] = "2011-12-28";
new_item1["time"] = "22:30:36";
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
```

4.4 输出 json

```
// 转换为字符串（带格式）
std::string out = root.toStyledString();
// 输出无格式json字符串
Json::FastWriter writer;
std::string out2 = writer.write(root);
```