# 概述

json自动序列化与反序列化工具。

在京东的远程调用接口中，我们时常需要将结构体，序列化与反序列化成json数据。JsonCpp库就是做这样的操作的，这一步操作一般都是人手实现，繁琐，无聊，且容易出错，所以，我们能否实现一个工具让结构体序列化与反序列化成json数据的方式是自动化的呢？

# 源码

<https://github.com/fishedee/FishJsonCpp>

已经实现好了，源码已经放到github上

# 编译

直接make就可以了，会生成一个fishjson的可执行文件

# 例子

## 编写idl

首先，我们要写一个idl，表明需要自动化json转换的结构体。

#include <stdint.h>

#include <vector>

#include <list>

#include <map>

#include <set>

#include <string>

//京东库存服务的IDL

namespace jd{

namespace core{

namespace stock{

class CLockInventoryForPaiPaiReq{

//商品id

std::string productId;

};

class CLockInventoryForPaiPaiResp{

//结果

int32\_t retCode;

};

}

}

}

## 执行转换

参数比较简单，输入是idl文件，输出是含有自动化转换代码的C++头文件

./fishjson < in.h > out.h

## 生成结果

#ifndef \_\_JD\_CORE\_STOCK\_H\_\_

#define \_\_JD\_CORE\_STOCK\_H\_\_

#include <sstream>

#include "app\_platform/intf\_serializable.h"

#include "c2cplatform/library/the3/jsoncpp64/reader.h"

#include "c2cplatform/library/the3/jsoncpp64/value.h"

#include "c2cplatform/library/the3/jsoncpp64/writer.h"

#include <stdint.h>

#include <vector>

#include <list>

#include <map>

#include <set>

#include <string>

namespace jd{

namespace core{

namespace stock{

class CLockInventoryForPaiPaiReq{

private:

std::string m\_productId;

bool m\_productId\_u;

public:

CLockInventoryForPaiPaiReq():

m\_productId(),

m\_productId\_u(false){

}

~CLockInventoryForPaiPaiReq(){

}

public:

const std::string& GetProductId()const{

return m\_productId;

}

bool IsProductIdSet()const{

return m\_productId\_u;

}

void SetProductId( const std::string& productId){

m\_productId = productId;

m\_productId\_u = true;

}

public:

void DumpData( std::ostream& os , int32\_t indent = 0 )const{

DumpDataTree( os ,indent );

}

void DumpDataTree( std::ostream& os ,int32\_t indent = 0 )const{

g\_pprint(os,"productId",m\_productId,indent);

}

public:

Json::Value ToJson()const{

Json::Value root;

std::stringstream ss;

Json::Value root\_1;

if( m\_productId\_u == true ){

root\_1 = Json::Value(m\_productId);

}

ss.str("");ss<<"productId";

root[ss.str()] = root\_1;

return root;

}

void FromJson( const Json::Value& root ){

if( root.isObject() == false ){

return;

}

if( root["productId"].isString() == true ){

m\_productId = root["productId"].asString();

m\_productId\_u = true;

}

}

};

class CLockInventoryForPaiPaiResp{

private:

int32\_t m\_retCode;

bool m\_retCode\_u;

public:

CLockInventoryForPaiPaiResp():

m\_retCode(),

m\_retCode\_u(false){

}

~CLockInventoryForPaiPaiResp(){

}

public:

const int32\_t& GetRetCode()const{

return m\_retCode;

}

bool IsRetCodeSet()const{

return m\_retCode\_u;

}

void SetRetCode( const int32\_t& retCode){

m\_retCode = retCode;

m\_retCode\_u = true;

}

public:

void DumpData( std::ostream& os , int32\_t indent = 0 )const{

DumpDataTree( os ,indent );

}

void DumpDataTree( std::ostream& os ,int32\_t indent = 0 )const{

g\_pprint(os,"retCode",m\_retCode,indent);

}

public:

Json::Value ToJson()const{

Json::Value root;

std::stringstream ss;

Json::Value root\_1;

if( m\_retCode\_u == true ){

root\_1 = Json::Value(m\_retCode);

}

ss.str("");ss<<"retCode";

root[ss.str()] = root\_1;

return root;

}

void FromJson( const Json::Value& root ){

if( root.isObject() == false ){

return;

}

if( root["retCode"].isInt() == true ){

m\_retCode = root["retCode"].asInt();

m\_retCode\_u = true;

}

}

};

}

}

}

#endif

## 调用

int main(){

CLockInventoryForPaiPaiReq req;

Json::Value value1,value2;

value1 = req.toJson();

req.fromJson( value2 );

return 0;

}

示例代码，如上所述，简单直接。

# 特征

1. 支持常见的各种基础数据类型，如int,long,double
2. 支持各种容器类型，如vector,list,set,map
3. 支持嵌套自定义结构体