**序列化结构代码生成器1.0**

**研发三部**

模块名： <weedong/core/seq/seq\_write.h>

依赖: <weedong/core/seq/seq\_log.h>

简介(详细请看<weedong/core/ seq/seq\_write.h > )：

提供对序列化结构信息进行解释

相关函数

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description: 将序列化记录生成C++类文件

Input: log 需要生成的类记录文件

Return: 类记录字符串

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static std::string transformCPP( CSeqLog &log );

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description: 将序列化记录生成AS类文件

Input: log 需要生成的类记录文件

Return: 类记录字符串

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static std::string transformAS( CSeqLog &log );

基本信息示例：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

定义CSampleBase

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CSampleBase

{

id : uint16 = 65535; //Id号

name : string = "示例基类描述"; //名称

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

定义CSample, 继承CSampleBase

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CSample : CSampleBase

{

var1 : int8 = -1; //int8变量

var2 : uint8 = 1; //uint8变量

var3 : int16 = -2; //int16变量

var4 : uint16 = 2; //uint16变量

var5 : int32 = -3; //int32变量

var6 : uint32 = 3; //uint32变量

var7 : float = -3.1415; //float变量

var8 : string = "研发三部"; //字符串变量

var9 : CSampleBase; //对象变量

var10 : array<string>; //对象数组变量

var11 : map<CSampleBase>; //对象词典变量

}

C++生成示例：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

定义CSampleBase

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class CSampleBase : public wd::CSeq

{

public:

uint16 id; //Id号

std::string name; //名称

CSampleBase()

{

id = 65535; //Id号

name = "示例基类描述"; //名称

}

void write( wd::CStream &stream ){ loop( stream, CSeq::eWrite ); }

void read( wd::CStream &stream ){ loop( stream, CSeq::eRead ); }

void loop( wd::CStream &stream, CSeq::ELoopType type )

{

wd::CSeq::loop( stream, type );

TFVarTypeProcess( id, type, stream );

TFVarTypeProcess( name, type, stream );

}

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

定义CSample, 继承CSampleBase

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class CSample : public CSampleBase

{

public:

int8 var1; //int8变量

uint8 var2; //uint8变量

int16 var3; //int16变量

uint16 var4; //uint16变量

int32 var5; //int32变量

uint32 var6; //uint32变量

float var7; //float变量

std::string var8; //字符串变量

CSampleBase var9; //对象变量

std::vector< std::string > var10; //对象数组变量

std::map< std::string, CSampleBase > var11; //对象词典变量

CSample()

{

var1 = -1;

var2 = 1;

var3 = -2;

var4 = 2;

var5 = -3;

var6 = 3;

var7 = -3.1415f;

var8 = "研发三部";

}

void write( wd::CStream &stream ){ loop( stream, CSeq::eWrite ); }

void read( wd::CStream &stream ){ loop( stream, CSeq::eRead ); }

void loop( wd::CStream &stream, CSeq::ELoopType type )

{

CSampleBase::loop( stream, type );

TFVarTypeProcess( var1, type, stream );

TFVarTypeProcess( var2, type, stream );

TFVarTypeProcess( var3, type, stream );

TFVarTypeProcess( var4, type, stream );

TFVarTypeProcess( var5, type, stream );

TFVarTypeProcess( var6, type, stream );

TFVarTypeProcess( var7, type, stream );

TFVarTypeProcess( var8, type, stream );

TFVarTypeProcess( var9, type, stream );

TFVarTypeProcess( var10, type, stream );

TFVarTypeProcess( var11, type, stream );

}

};

AS生成示例:

package

{

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

定义CSampleBase

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

public class CSampleBase

{

public var id:Number = 65535; //Id号

public var name:String = "示例基类描述"; //名称

public function CSampleBase()

{

elementInfo = static\_element\_info;

}

static protected var static\_element\_info:Vector.<CSeqEleInfo> = CSeq.static\_element\_info

.concat

(

new <CSeqEleInfo>

[

new CSeqEleInfo( 'uint16', 'id' ),

new CSeqEleInfo( 'string', 'name' )

]

);

}

}

package

{

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

定义CSample, 继承CSampleBase

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

public class CSample extends CSampleBase

{

public var var1:Number = -1; //int8变量

public var var2:Number = 1; //uint8变量

public var var3:Number = -2; //int16变量

public var var4:Number = 2; //uint16变量

public var var5:Number = -3; //int32变量

public var var6:Number = 3; //uint32变量

public var var7:Number = -3.1415; //float变量

public var var8:String = "研发三部"; //字符串变量

public var var9:CSampleBase = new CSampleBase; //对象变量

public var var10:Vector.<String> = new Vector.<String>; //对象数组变量

public var var11:Object = new Object; //对象词典变量

public function CSample()

{

elementInfo = static\_element\_info;

}

static protected var static\_element\_info:Vector.<CSeqEleInfo> = CSampleBase.static\_element\_info

.concat

(

new <CSeqEleInfo>

[

new CSeqEleInfo( 'int8', 'var1' ),

new CSeqEleInfo( 'uint8', 'var2' ),

new CSeqEleInfo( 'int16', 'var3' ),

new CSeqEleInfo( 'uint16', 'var4' ),

new CSeqEleInfo( 'int32', 'var5' ),

new CSeqEleInfo( 'uint32', 'var6' ),

new CSeqEleInfo( 'float', 'var7' ),

new CSeqEleInfo( 'string', 'var8' ),

new CSeqEleInfo( 'CSampleBase', 'var9' ),

new CSeqEleInfo( 'array', 'var10', 'string' ),

new CSeqEleInfo( 'map', 'var11', 'CSampleBase' )

]

);

}

}

使用示例

#include <weedong/core/seq/seq\_writer.h>

int main( int argc, char\* argv[] )

{

wd::CSeqLog log;

log.className = “CTest”;

//解释结构化信息

std::list< wd::CSeqLog > logs = wd::CSeqParser::parse( seq, seq + sizeof( seq ) );

//输出C++类声明

std::string cppString = wd::CSeqWriter::transformCPP( \*logs.begin() );

//输出AS类声明

std::string asString = wd::CSeqWriter::transformAS( \*logs.begin() );

return 0;

}