# 一、源代码说明

本文介绍的是采用开发框架的解析xml格式字符串函数。

开发框架函数和类的声明文件是/project/public/\_public.h。

开发框架函数和类的定义文件是/project/public/\_public.h.cpp。

示例程序位于/project/public/demo目录中。

编译规则文件是/project/public/demo/makefile。

# 二、xml格式字符串介绍

xml格式字符串是应用开发中被广泛采用的一种数据格式，简单易懂，容错性和可扩展性非常好，是数据处理、数据通讯和数据交换等应用场景的首选数据格式。

完整的xml格式比较复杂，但是，在实际开发中，对我们C/C++程序员来说，绝大部分场景下用到的xml数据格式比较简单，例如表示文件列表信息的xml数据集或文件内容如下：

<data>

<filename>\_public.h</filename><mtime>2020-01-01 12:20:35</mtime><size>1834</size><endl/>

<filename>\_public.cpp</filename><mtime>2020-01-01 10:10:15</mtime><size>5094</size><endl/>

</data>

数据集说明：

<data>：数据集的开始。

</data>：数据集的结束。

<endl/>：每行数据的结束。

filename标签：文件名。

mtime标签：文件最后一次被修改的时间。

size标签：文件的大小。

# 三、xml格式字符串的解析

在开发框架中，提供了解析以下xml格式字符串的一系函数。

函数声明：

bool GetXMLBuffer(const char \*xmlbuffer,const char \*fieldname,bool \*value);

bool GetXMLBuffer(const char \*xmlbuffer,const char \*fieldname,int \*value);

bool GetXMLBuffer(const char \*xmlbuffer,const char \*fieldname,unsigned int \*value);

bool GetXMLBuffer(const char \*xmlbuffer,const char \*fieldname,long \*value);

bool GetXMLBuffer(const char \*xmlbuffer,const char \*fieldname,unsigned long \*value);

bool GetXMLBuffer(const char \*xmlbuffer,const char \*fieldname,double \*value);

bool GetXMLBuffer(const char \*xmlbuffer,const char \*fieldname,char \*value,const int ilen=0);

参数说明：

xmlbuffer：待解析的xml格式字符串的内容。

fieldname：字段的标签名。

value：传入变量的地址，用于存放字段内容，支持bool、int、insigned int、long、unsigned long、double和char[]。

注意，当value参数的数据类型为char []时，必须保证value数组的内存足够，否则可能发生内存溢出的问题，也可以用ilen参数限定获取字段内容的长度，ilen的缺省值为0，表示不限定获取字段内容的长度。

返回值：true-获取成功；false-获取失败。

**示例（demo22.cpp）**

/\*

\* 程序名：demo22.cpp，此程序演示调用开发框架的GetXMLBuffer函数解析xml字符串。

\*/

#include "../\_public.h"

// 用于存放足球运动员资料的结构体。

struct st\_player

{

char name[51]; // 姓名

char no[6]; // 球衣号码

bool striker; // 场上位置是否是前锋，true-是；false-不是。

int age; // 年龄

double weight; // 体重，kg。

long sal; // 年薪，欧元。

char club[51]; // 效力的俱乐部

}stplayer;

int main()

{

memset(&stplayer,0,sizeof(struct st\_player));

char buffer[301];

STRCPY(buffer,sizeof(buffer),"<name>messi</name><no>10</no><striker>true</striker><age>30</age><weight>68.5</weight><sal>21000000</sal><club>Barcelona</club>");

GetXMLBuffer(buffer,"name",stplayer.name,50);

GetXMLBuffer(buffer,"no",stplayer.no,5);

GetXMLBuffer(buffer,"striker",&stplayer.striker);

GetXMLBuffer(buffer,"age",&stplayer.age);

GetXMLBuffer(buffer,"weight",&stplayer.weight);

GetXMLBuffer(buffer,"sal",&stplayer.sal);

GetXMLBuffer(buffer,"club",stplayer.club,50);

printf("name=%s,no=%s,striker=%d,age=%d,weight=%.1f,sal=%ld,club=%s\n",\

stplayer.name,stplayer.no,stplayer.striker,stplayer.age,\

stplayer.weight,stplayer.sal,stplayer.club);

// 输出结果:name=messi,no=10,striker=1,age=30,weight=68.5,sal=21000000,club=Barcelona

}

# 四、应用经验

对C/C++程序员来说，采用简单的xml字符串表达数据可以提高开发效率，我不建议采用复杂的xml格式，会让程序代码很烦锁。

如果在实际开发中需要解析更复杂的xml，可以寻找网上的开源库，例如libxml++。