<https://blog.csdn.net/yufang5780642/article/details/46536879>

无论是使用别人的工具软件，还是自己开发工具，都会出现使用配置文件的需求。对于配置层次不多的参数配置，使用.ini文件就足够了。当配置项涉及比较复杂的描述结构时，就需要使用到.xml文件进行描述了。

ini开源库的使用见这里https://launchpad.net/miniini/+download

常见XML开源库：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 实现语言 | 源码地址 | 基于模型 |
| Tinyxml | C++ | http://sourceforge.net/projects/tinyxml/files/tinyxml/ | DOM |
| libxml2 | C | http://xmlsoft.org/sources/win32/libxml2-2.7.8.win32.zip | DOM |
| CMarkup | C++ | <http://www.firstobject.com/Markup115.zip> | DOM |
| Mini-XML | C | <http://www.msweet.org/files/project3/mxml-2.9.tar.gz> | DOM/SAX |
| Expat-XML | C | <http://sourceforge.net/projects/expat/files/expat/> | SAX |
| Xerces | C++/Java/Perl | <http://xerces.apache.org/mirrors.cgi> | DOM/SAX |

上表中的模型含义如下：

SAX（Simple API for XML）是基于事件的，其基本工作流程是分析XML文档，当发现了一个新的元素时，产生一个对应事件，并调用相应的用户处理函数。这种方式占用内存少，速度快，但用户程序相应地会比较复杂。

DOM（Document Object Model）是在分析时，一次性地将整个XML文档进行分析，并在内存中形成对应的树结构，同时，向用户提供一系列的接口来访问和编辑该树结构。这种方式占用内存大，速度往往慢于SAX，但可以给用户提供一个面向对象的访问接口，对用户更为友好。

注：tinyxml增强版tinyxml2，相比于tinyxml，内存性能有一定提升，并且适用于移动平台，获取地址http://www.grinninglizard.com/tinyxml2/index.html

以在工作中会接触到的tinyxml使用为例，简要介绍一下其使用方法

1. 下载某一个版本的tinyxml，例如tinyxml v2.6.1(是VS2008工程类型，v2.6.2是VS2010工程类型)

其中的主要文件如下：

tinyxml.cpp：      tinyxml主要类实现

tinystr.cpp：      tinyxml在不使用STL时自定义的string类实现

tinyxmlerror.cpp： tinyxml错误信息定义

tinyxmlparser.cpp：tinyxml主要类实现中涉及数据转换的函数实现

tinyxml.h：        tinyxml主要类声明

tinystr.h：        tinyxml在不使用STL时自定义的string类声明

使用时将这些文件都包含到自己的工程项目中便可以直接使用，也可以将源码编译出lib库来使用

2. tinyxml主要类的继承层次

tinyxml中的常用类型是TiXmlNode和TiXmlAttribute，即认为在xml文件中，除了节点的属性外，其他一切元素都被认为是节点，各个类对应于xml文件如下：

1 <?xml version="1.0" encoding="UTF-8"?>

2 <!-- comment -->

3 <root attr1="1">

4     <element attr2="2" attr3="3">

5         text

6     </element>

7 </root>

8 <//unknow//>

（1）以上8行构成的xml文档称为TiXmlDocument

（2）第1行节点是声明，称为TiXmlDeclaration，声明通过调用TiXmlDocument::FirstChild()->ToDeclaration()获取

（3）第2行节点是注释，称为TiXmlComment，注释节点通过上一个节点调用NextSibling()->ToComment()获取

（4）第3行节点root是根节点，称为TiXmlElement，通过调用TiXmlDocument::RootElement()获取，或者通过上一个节点调用NextSibling()->ToElement()获取

（5）第4行节点element是根节点里面的子节点，也称为TiXmlElement，通过此处根节点调用FirstChild()->ToElement()获取

（6）第5行节点text是当前节点里的内容，称为TiXmlText，通过上一个节点调用FirstChild()->ToText()获取

（7）第4行元素attr2是节点的属性，称为TiXmlAttribute，通过当前节点调用ToElement()->FirstAttribute()获取，同一个节点的属性之间可以直接通过上一个属性调用Next()获取

（8）第8行节点是未知节点，称为TiXmlUnknown，通过上一个节点调用NextSibling()->ToUnknown()获取

3. 读取过程示例：

#include "tinyxml.h"

#include <string>

using namespace std;

int main()

{

    string strXmlFile("examplexmlfile.xml");

    TiXmlDocument         xmlDoc;

    TiXmlDeclaration\*   pObjXmlDeclaration = NULL;

    TiXmlComment\*        pObjXmlComment = NULL;

    TiXmlElement\*          pObjXmlRootElement = NULL;

    TiXmlElement\*          pObjXmlElement = NULL;

    TiXmlText\*               pObjXmlText = NULL;

    TiXmlUnknown\*       pObjXmlUnknown = NULL;

    TiXmlAttribute\*      pObjXmlAttribute = NULL;

    xmlDoc.LoadFile(strXmlFile.c\_str()); //加载整个xml文档

    //使用之前判空，以下略

    if (NULL == xmlDoc.FirstChild())

    {

        return 0;

    }

    pObjXmlDeclaration = xmlDoc.FirstChild()->ToDeclaration(); //获取第1行声明节点

    pObjXmlComment = pObjXmlDeclaration->NextSibling()->ToComment(); //获取第2行注释节点

    pObjXmlRootElement = xmlDoc.RootElement(); //获取xml文档的根节点

    pObjXmlUnknown = pObjXmlRootElement->NextSibling()->ToUnknown(); //获取xml文档中不识别的未知节点

    pObjXmlElement = pObjXmlRootElement->FirstChild()->ToElement(); //获取当前节点的子节点

    pObjXmlAttribute = pObjXmlElement->FirstAttribute(); //获取当前节点的第一个属性

    pObjXmlAttribute = pObjXmlAttribute->Next(); //获取当前属性的下一个属性

    pObjXmlText = pObjXmlElement->FirstChild()->ToText(); //获取当前节点的子节点

    return 0;

}