项目阅读报告

(TinyXML)

计33 侯林之 2023010779

https://github.com/leethomason/tinyxml2

项目整体介绍

TinyXML2是一个轻量级、高效的C++ XML解析库，具有简单直接的API和易于使用的接口，支持XML文档的解析、创建、读取和写入，提供丰富的功能和错误处理机制，可跨平台运行，是开发者处理XML数据的理想工具。

- 解析和读取：TinyXML2可以轻松地解析和读取XML文档，将XML文件中的数据转换为程序中的数据结构，方便进行后续的操作和处理。
- 创建和写入：TinyXML2可以创建新的XML文档，添加节点、属性和内容，并将数据写入到XML文件中，实现对XML文档的生成和编辑。
- 操作和修改：TinyXML2提供了丰富的接口和方法，用于操作和修改XML文档中的节点、元素、属性和内容，可以添加、删除、修改节点的信息。
- 错误处理和异常处理：TinyXML2提供了错误码和异常处理机制，在解析和操作XML文档过程中，可以捕获和处理可能发生的错误，保证程序稳定运行。

使用方法：将tinyxml2.cpp和tinyxml2.h拷贝至项目目录,使用时包含

```
#include "tinyxml2.h"
using namespace tinyxml2;
```

项目整体介绍

TinyXML2是一个轻量级、高效的C++ XML解析库，具有简单直接的API和易于使用的接口，支持XML文档的解析、创建、读取和写入，提供丰富的功能和错误处理机制，可跨平台运行，是开发者处理XML数据的理想工具。

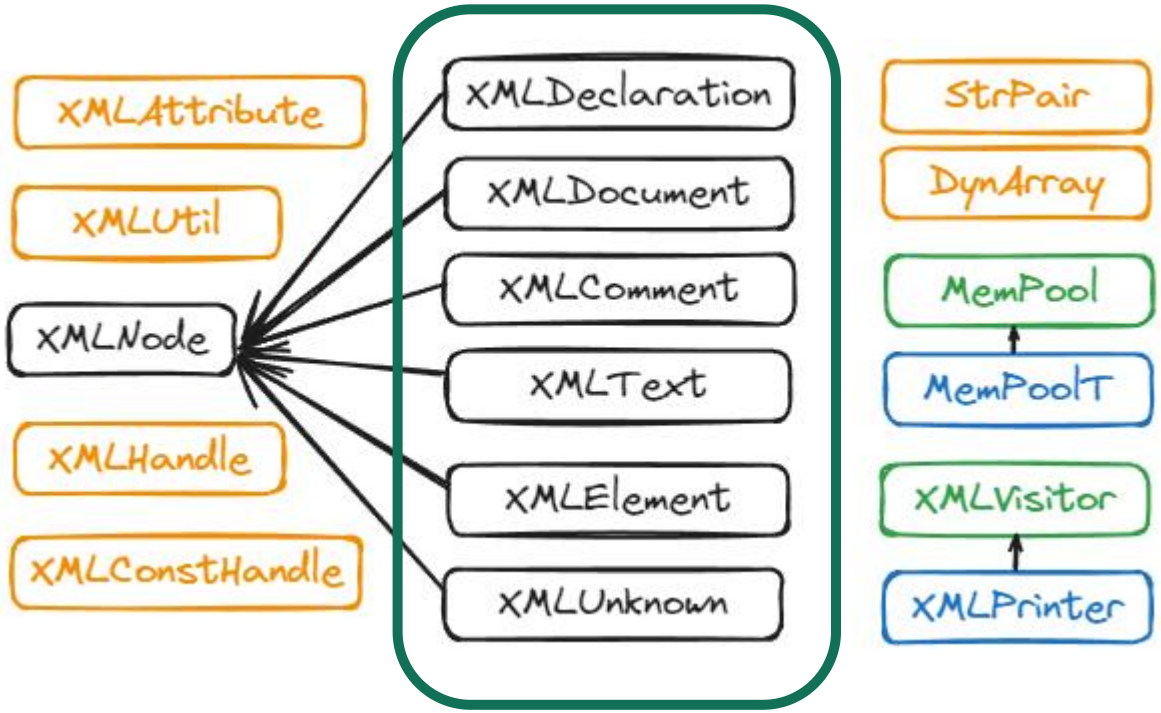
- 解析和读取：TinyXML2可以轻松地解析和读取XML文档，将XML文件中的数据转换为程序中的数据结构，方便进行后续的操作和处理。
- 创建和写入：TinyXML2可以创建新的XML文档，添加节点、属性和内容，并将数据写入到XML文件中，实现对XML文档的生成和编辑。
- 操作和修改：TinyXML2提供了丰富的接口和方法，用于操作和修改XML文档中的节点、元素、属性和内容，可以添加、删除、修改节点的信息。
- 错误处理和异常处理：TinyXML2提供了错误码和异常处理机制，在解析和操作XML文档过程中，可以捕获和处理可能发生的错误，保证程序稳定运行。

使用方法：将tinyxml2.cpp和tinyxml2.h拷贝至项目目录,使用时包含

```
#include "tinyxml2.h"
using namespace tinyxml2;
```

代码框架分析

项目包含了tinyxml2.cpp和tinyxml2.h两个文件，其中tinyxml2.h中声明了TinyXML2库中主要使用的类和函数，而tinyxml2.cpp则包含了对这些类和函数的具体实现。



一些核心类的说明

- XMLNode：该类提供了处理XML文档中节点的基本功能，是除了XML- Attributes之外所有对象的基类。
- XMLDocument等（左图中列）：TinyXML2 的核心对象，用于表示XML声明、文档、注释、文本、元素、未知节点，继承自XMLNode。
- XMLAttribute：用于表示XML元素的属性，属性是元素的名称-值对。
- XMLUtil：提供了许多实用的功能如处理字符串操作、类型转换等。
- XMLHandle和XMLConstHandle：用于封装节点指针和空指针检查，使代码更加简洁、安全和易于理解。
- XMLPrinter：灵活地控制XML文档的打印输出方式，包括打印到内存、打印到文件以及直接打印XML内容。

具体功能测试——从C++对象到XML文档

```cpp
void create()
{
    XMLDocument doc;
    XMLDeclaration *declaration = doc.NewDeclaration();
    doc.InsertEndChild(declaration);
    XMLElement *root = doc.NewElement("Root");
    doc.InsertEndChild(root);
    XMLElement *child = doc.NewElement("Child");
    child->SetText("Hello, World!");
    root->InsertEndChild(child);
    XMLText *textNode = doc.NewText("TextContent");
    child->InsertEndChild(textNode);
    XMLElement *AAA = doc.NewElement("AAA");
    root->InsertFirstChild(AAA);
    XMLElement *BBB = doc.NewElement("BBB");
    root->InsertAfterChild(AAA, BBB);
    XMLElement *CCC = doc.NewElement("CCC");
    CCC->SetText("This is text of CCC.");
    AAA->InsertEndChild(CCC);
    XMLComment *comment = doc.NewComment("This is a comment");
    doc.InsertEndChild(comment);
    doc.SaveFile("example.xml");
}
```

XML文档的创建和修改

create()函数用于创建一个XML文档，
modify()函数则可修改已存在的XML文档。

```cpp
void modify()
{
    XMLDocument doc;
    if (doc.LoadFile("example.xml") == XML_SUCCESS)
    {
        XMLElement *root = doc.RootElement();
        XMLElement *nodeToModify1 = root->FirstChildElement("AAA");
        XMLElement *nodeToModify2 = root->FirstChildElement("BBB");
        XMLElement *nodeToModify3 = nodeToModify1->FirstChildElement("CCC");
        nodeToModify1->SetName("NewName");
        nodeToModify1->SetText("NewText");
        nodeToModify1->SetAttribute("sing", "dance");
        nodeToModify1->SetAttribute("cxk", "ikun");
        nodeToModify2->SetAttribute("rap", "basketball");
        nodeToModify2->SetAttribute("giegie", 2.5);
        nodeToModify1->DeleteChild(nodeToModify3);
        doc.SaveFile("example_modified.xml");
    }
}
```

工作流程——create()

- 创建一个XMLDocument对象doc，用于表示一个完整的XML文档。
- 创建一个XMLDeclaration对象declaration，通过doc.NewDeclaration()方法创建一个XML声明节点。
- 将XML声明节点插入到XML文档的末尾，即作为文档的第一个子节点。
- 创建一个名为"Root"的根元素节点，并将其插入到XML文档的末尾。
- 在根元素节点下创建一个名为"Child"的子元素节点，并设置其文本内容为"Hello, World!"。
- 在"Child"节点中插入一个文本节点，文本内容为"TextContent"。
- 创建一个名为"AAA"的元素节点，并将其插入到根元素节点的首部。
- 创建一个名为"BBB"的元素节点，并将其插入到"AAA"节点之后。
- 创建一个名为"CCC"的元素节点，并设置其文本内容为"This is text of CCC."，再将其插入到"AAA"节点的末尾。
- 创建一个注释节点，内容为"This is a comment"，并将其插入到XML文档的末尾。
- 最后将整个XML文档保存到名为"example.xml"的文件中。

工作流程——modify()

- 创建一个XMLDocument对象doc。
- 加载名为"example.xml"的XML文件到doc中，如果加载成功即XML_SUCCESS，则继续执行下面的操作。
- 获取XML文档的根元素节点。
- 通过根元素节点获取名为"AAA"和"BBB"的子元素节点。
- 从"AAA"节点中获取名为"CCC"的子元素节点。
- 修改"AAA"节点的名称为"NewName"，设置其文本内容为"NewText"。
- 给"AAA"节点设置两个属性："sing"为"cxk"，"dance"为"ikun"。
- 给"BBB"节点设置两个属性："rap"为"giegie"，"basketball"为2.5。
- 从"AAA"节点中删除"CCC"节点。
- 最后将修改后的XML文档保存到名为"example_modified.xml"的文件中。

通过这两个函数的调用流程，可以实现对一个XML文档的创建、修改和保存，展示了TinyXML2库在处理和操作XML文档时的基本用法。

具体功能测试——从XML文档到C++对象

XML文档的读取和查询

```
void read()
{
    tinyxml2::XMLDocument doc;
    doc.LoadFile("example_modified.xml");
    tinyxml2::XMLElement *root = doc.RootElement();
    tinyxml2::XMLElement *firstNode = root->FirstChildElement();
    if (firstNode)
    {
        std::string nodeName = firstNode->Name();
        printf("node name is %s\n", nodeName.c_str());
        std::string nodeConrtext = firstNode->GetText();
        printf("node text is %s\n", nodeConrtext.c_str());
        const char *attrNameValue = firstNode->Attribute("sing");
        std::cout << "Parameter name value: " << attrNameValue << std::endl;
        const tinyxml2::XMLAttribute *attr = firstNode->FirstAttribute();
        while (attr)
        {
            std::string attrName = attr->Name();
            std::string attrValue = attr->Value();
            printf("param name is: %s, param value is: %s\n", attrName.c_str(), attrValue.c_str());
            attr = attr->Next();
        }
    }
}
```

工作流程——read()

- 创建一个tinyxml2::XMLDocument对象doc。
- 加载名为"example_modified.xml"的XML文件到doc中。
- 获取XML文档的根元素节点。
- 从根元素节点中获取第一个子元素节点。
- 判断第一个子元素节点是否存在，如果存在，则执行以下操作。
- 获取第一个子元素节点的名称，将其存储在nodeName中，并通过printf输出节点名称。
- 获取第一个子元素节点的文本内容，将其存储在nodeConrtext中，并通过printf输出节点文本内容。
- 获取第一个子元素节点的名为"sing"的属性的值，存储在attrNameValue中，并通过std::cout输出参数名和值。
- 获取第一个子元素节点的第一个属性，存储在attr中，进入循环处理属性信息。
- 循环遍历节点的所有属性，获取每个属性的名称和值，并通过printf输出参数名和值。

通过以上流程，这个函数实现了读取修改后的XML文件"example_modified.xml"中第一个子元素节点的名称、文本内容和属性信息，并将结果输出到控制台，方便开发者查看和验证XML文件的修改是否成功。

结果展示

create()结果:

example.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Root>
    <AAA>
        <CCC>This is text of CCC.</CCC>
    </AAA>
    <BBB/>
    <Child>Hello, World!TextContent</Child>
</Root>

<!--This is a comment-->
```

modify()结果:

example_modified.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Root>
    <NewName sing="dance" cxk="ikun">NewText</NewName>
    <BBB rap="basketball" giegie="2.5"/>
    <Child>Hello, World!TextContent</Child>
</Root>

<!--This is a comment-->
```

read()输出结果:

```
zenith@Zenith:~/workspace/test$ ./main
node name is NewName
node text is NewText
Parameter name value: dance
param name is: sing, param value is: dance
param name is: cxk, param value is: ikun
```

优越性分析

代码设计模式采用模版方法

- 通过将处理XML节点的共同流程抽象为模板方法，实现了基类节点的基本功能和子类特定节点的差异化处理
- XMLNode：该类提供了处理XML文档中节点的基本功能，是除了XML- Attributes之外所有对象的基类。
- 而XMLDocument、XMLComment、XMLElement、XMLUnknown、XMLText、XMLDeclaration等核心对象则继承自XMLNode，代表着XML文档中的不同节点类型，分别表示XML声明、文档、注释、文本、元素、未知节点。
- 这种继承关系使得TinyXML2库能够更加灵活地表示和操作XML文档中各种节点，可实现不同节点类型的定制化处理逻辑，提高了代码的可复用性和可扩展性，使整个XML文档处理框架更为强大和可靠。

```
class TINYXML2_LIB XMLNode;
class TINYXML2_LIB XMLText
    : public XMLNode;
class TINYXML2_LIB XMLComment
    : public XMLNode;
class TINYXML2_LIB XMLDeclaration
    : public XMLNode;
class TINYXML2_LIB XMLUnknown
    : public XMLNode;
class TINYXML2_LIB XMLElement
    : public XMLNode;
class TINYXML2_LIB XMLDocument
    : public XMLNode;
```

优越性分析

代码错误处理详细、鲁棒性强

- 使用枚举类型XMLError定义了各种可能的错误类型，提高了代码的可读性和可维护性，使得对不同错误类型的处理更加清晰。
- 通过使用不同的Query函数来获取不同类型的数值，避免了类型转换错误，提高了代码的健壮性和安全性，减少了潜在的运行时错误发生的可能性。
- 进行了类型转换的错误处理，当无法成功转换时会返回相应的错误类型，使得在数据类型转换失败的情况下，能够明确告知调用方问题出现在哪里，更容易进行问题定位和修复。
- 此种错误处理方式使得代码具有较好的鲁棒性和健壮性，能够有效处理各种可能出现的错误情况，提高了程序的可靠性和稳定性。

```
enum XMLError
{
    XML_SUCCESS = 0,
    XML_NO_ATTRIBUTE,
    XML_WRONG_ATTRIBUTE_TYPE,
    XML_ERROR_FILE_NOT_FOUND,
    XML_ERROR_FILE_COULD_NOT_BE_OPENED,
    XML_ERROR_FILE_READ_ERROR,
    XML_ERROR_PARSING_ELEMENT,
    XML_ERROR_PARSING_ATTRIBUTE,
    XML_ERROR_PARSING_TEXT,
    XML_ERROR_PARSING_CDATA,
    XML_ERROR_PARSING_COMMENT,
    XML_ERROR_PARSING_DECLARATION,
    XML_ERROR_PARSING_UNKNOWN,
    XML_ERROR_EMPTY_DOCUMENT,
    XML_ERROR_MISMATCHED_ELEMENT,
    XML_ERROR_PARSING,
    XML_CAN_NOT_CONVERT_TEXT,
    XML_NO_TEXT_NODE,
    XML_ELEMENT_DEPTH_EXCEEDED,

    XML_ERROR_COUNT
};
```

优越性分析

代码可扩展性强

- 继承体系：TinyXML2中XMLNode是所有节点对象的基类，其他核心对象如XMLDocument、XMLElement等都是继承自XMLNode，这种继承体系使得新功能的添加变得简单，只需创建新的子类并实现特定的功能即可扩展现有功能。
- 接口设计：TinyXML2提供了一系列方便的接口方法来操作XML文档节点，如AddElement、DeleteElement、QueryValue等，这些接口的设计和命名清晰简洁，使得添加新功能和操作节点变得直观和方便。
- 面向对象设计：TinyXML2采用面向对象的设计模式，将不同类型的节点封装成对象并提供统一的接口操作，这种设计使得系统具有较好的灵活性和可扩展性，便于增添新的节点类型或功能。

```
class XMLDocument;
class XMLElement;
class XMLAttribute;
class XMLComment;
class XMLText;
class XMLDeclaration;
class XMLUnknown;
class XMLPrinter;
```

可改进空间

tinyxml2虽然轻巧简洁，但相比于其他XML解析库，功能还有一定的提升空间，可以考虑增加一些高级功能或者特性。

- API设计方面，可以考虑为tinyxml2增加更多的便捷、高级的功能接口，例如提供更灵活的节点遍历和搜索方法、更多属性操作的便捷方式，以及更丰富的节点操作方法等，这样能够使得开发者能够更快速、更方便地对XML文档进行操作，提高开发效率。还可以考虑优化一些API的命名和参数设计，使得接口更加直观清晰，易于理解和使用。
- 在功能丰富性方面，可以进一步提升tinyxml2的功能，例如增加对DTD（文档类型定义）的支持、实现XPath查询功能、提供更为灵活的错误处理机制等功能。此外，可以考虑增加对XML Schema等高级XML标准的支持，以满足更复杂、更多样化的XML文档处理需求。通过扩展这些高级功能或特性，将使得tinyxml2在功能上更加全面，提高库的实用性和适用性。

结束