# tinyxml源码剖析（1）-源码中几个主要类的关系

原创 2015年09月23日 21:26:30

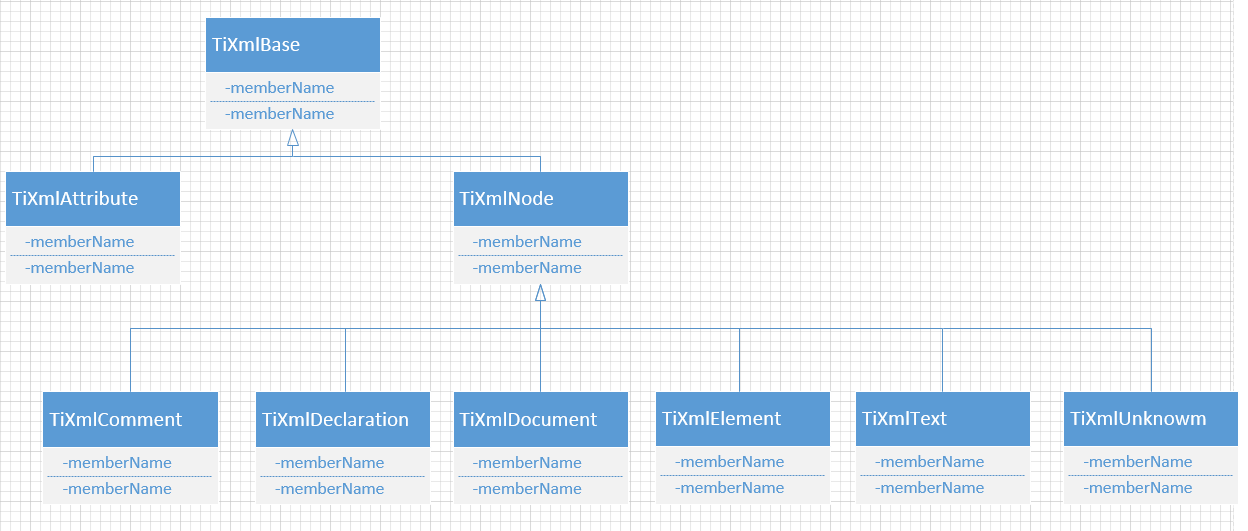
* 标签：
* [tinyxml源码](http://so.csdn.net/so/search/s.do?q=tinyxml源码&t=blog" \t "http://blog.csdn.net/u012796139/article/details/_blank) /
* [tinyxml剖析](http://so.csdn.net/so/search/s.do?q=tinyxml剖析&t=blog" \t "http://blog.csdn.net/u012796139/article/details/_blank) /
* 1029
* [编辑](http://write.blog.csdn.net/postedit/48687409)
* 删除

        tinyxml是一个小型的xml文件解析器，它使用文档对象模型(DOM)来进行解析，这种模型的处理方式为在解析时，一次性的将整个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，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，非常方便用户使用。整个源码主要有8个文件组成，6个头文件和2个cpp文件。tinyxml源码下载地址：[tinyxml源码下载](http://sourceforge.net/projects/tinyxml/" \t "http://blog.csdn.net/u012796139/article/details/_blank)

       关于XML基础知识在这里：[XML 新手入门基础知识](http://www.ibm.com/developerworks/cn/xml/x-newxml/" \l "list1" \t "http://blog.csdn.net/u012796139/article/details/_blank)（https://www.ibm.com/developerworks/cn/xml/x-newxml/#list1）

       关于tinyxml的简单使用请参考：[tinyxml之hello world](http://blog.csdn.net/u012796139/article/details/46706853" \t "http://blog.csdn.net/u012796139/article/details/_blank)

        tinyxml源码中主要由TiXmlBase、TiXmlAttribute、TiXmlNode、TiXmlComment、TiXmlDeclaration、TiXmlDocument、TiXmlElement、TiXmlText、TiXmlUnknown这几个类构成，它们之间的关系如下图所示：



        TiXmlBase是所有类的基类，TiXmlNode、TiXmlAttribute两个类都继承来自TiXmlBase类，其中TiXmlNode类指的是所有被<...>...<.../>包括的内容，而xml中的节点又具体分为以下几方面内容，分别是声明、注释、节点以及节点间的文本，因此在TiXmlNode的基础上又衍生出这几个类TiXmlComment、TiXmlDeclaration、TiXmlDocument、TiXmlElement、TiXmlText、TiXmlUnknown，分别用来指明具体是xml中的哪一部分。TiXmlAttribute类不同于TiXmlNode，它指的是在尖括号里面的内容，像<... \*\*\*=...>，其中\*\*\*就是一个属性。

**[cpp]** [view plain](http://blog.csdn.net/u012796139/article/details/48687409" \o "view plain) [copy](http://blog.csdn.net/u012796139/article/details/48687409" \o "copy) [print](http://blog.csdn.net/u012796139/article/details/48687409" \o "print)[?](http://blog.csdn.net/u012796139/article/details/48687409" \o "?)

1. //一个Document中可以包括：
2. Element(container/leaf)
3. Comment(leaf)
4. Unknow(leaf)
5. Declatation(leaf)
6. //一个Element可以包括：
7. Element(container/leaf)
8. Text(leaf)
9. Attributes(not on tree)
10. Comment(leaf)
11. Unknow(leaf)

//一个Document中可以包括：

Element(container/leaf)

Comment(leaf)

Unknow(leaf)

Declatation(leaf)

//一个Element可以包括：

Element(container/leaf)

Text(leaf)

Attributes(not on tree)

Comment(leaf)

Unknow(leaf)

源码中把整个xml文件的内容构造成了一个树形的结构，TiXmlNode类中有一下数据成员

**[cpp]** [view plain](http://blog.csdn.net/u012796139/article/details/48687409" \o "view plain) [copy](http://blog.csdn.net/u012796139/article/details/48687409" \o "copy) [print](http://blog.csdn.net/u012796139/article/details/48687409" \o "print)[?](http://blog.csdn.net/u012796139/article/details/48687409" \o "?)

1. TiXmlNode\*    parent;
2. TiXmlNode\*    prev;
3. TiXmlNode\*    next;
5. NodeType      type;
6. TIXML\_STRING  value;
8. TiXmlNode\*    firstChild;
9. TiXmlNode\*    lastChild;

TiXmlNode\* parent;

TiXmlNode\* prev;

TiXmlNode\* next;

NodeType type;

TIXML\_STRING value;

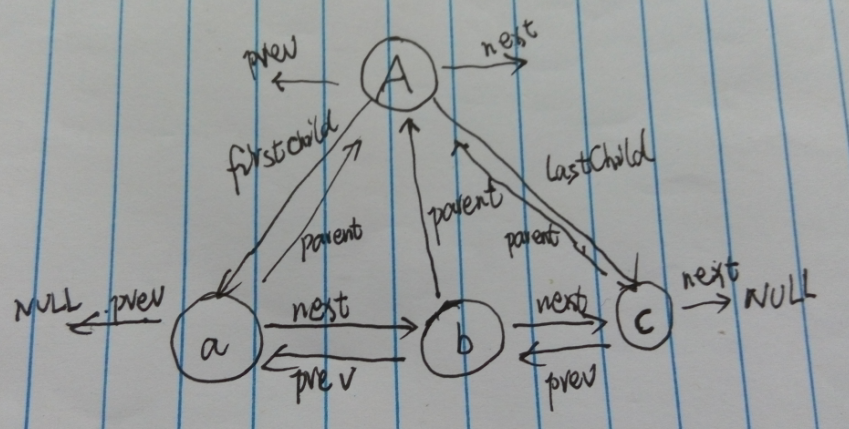
TiXmlNode\* firstChild;

TiXmlNode\* lastChild;

       NodeType为枚举类型，表示该TiXmlNode的类型，有document、element、comment、unknow、text、declaration

       TIXML\_STRING为tinyxml定义的一个类型，当使用STL时，就是string类型。

这些成员变量是指针，这样就组成了整个XML文件的解析树，XML文件的每部分都是树中的一些节点或者内容。如下图所示：



下面来看一个XML文件，具体看一下其中的节点关系：

**[html]** [view plain](http://blog.csdn.net/u012796139/article/details/48687409" \o "view plain) [copy](http://blog.csdn.net/u012796139/article/details/48687409" \o "copy) [print](http://blog.csdn.net/u012796139/article/details/48687409" \o "print)[?](http://blog.csdn.net/u012796139/article/details/48687409" \o "?)

1. <?xml version="1.0" ?>
2. <phonebook>
3. <item>
4. <name>hangzhou hdu</name>
5. <addr>hang zhou</addr>
6. <tel>110</tel>
7. <email>hdu1@hdu.com</email>
8. </item>
9. <item>
10. <name>hdu2</name>
11. <addr>hang zhou</addr>
12. <tel>111</tel>
13. <email>hdu2@hdu.com</email>
14. </item>
15. <item>
16. <name>hdu3</name>
17. <addr>hang zhou</addr>
18. <tel>112</tel>
19. <email>hdu3@hdu.com</email>
20. </item>
21. </phonebook>

<?xml version="1.0" ?>

<phonebook>

<item>

<name>hangzhou hdu</name>

<addr>hang zhou</addr>

<tel>110</tel>

<email>hdu1@hdu.com</email>

</item>

<item>

<name>hdu2</name>

<addr>hang zhou</addr>

<tel>111</tel>

<email>hdu2@hdu.com</email>

</item>

<item>

<name>hdu3</name>

<addr>hang zhou</addr>

<tel>112</tel>

<email>hdu3@hdu.com</email>

</item>

</phonebook>

       这个XML文件就是一个TiXmlDocument，它的第一个child就是TiXmlDeclaration，第二个child就是TiXmlElement，TiXmlElement本身又有3个child，分别是3TiXmlElement，就是3个item节点，每个item节点有4个child，也就是4个TiXmlElement，这4个TiXmlElement的Value分别是”name”、”addr”、”tel”、”email”。其中Value为”name”的TiXmlElement只有一个child，就是TiXmlText，TiXmlText的Value就是”hangzhou hdu”。至此，整个xml树就建立完成了。

参考：

1、[tinyxml源码学习-hello world](http://blog.csdn.net/u012796139/article/details/46706853" \t "http://blog.csdn.net/u012796139/article/details/_blank)

# [tinyxml源码剖析（1）-源码中几个主要类的关系 1](#_Toc26008)

原创 2015年07月01日 09:56:37

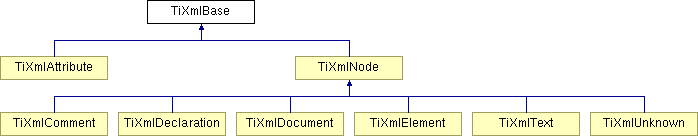
* 标签：
* [tinyxml](http://so.csdn.net/so/search/s.do?q=tinyxml&t=blog" \t "http://blog.csdn.net/u012796139/article/details/_blank) /
* [tinyxml简单使用](http://so.csdn.net/so/search/s.do?q=tinyxml简单使用&t=blog" \t "http://blog.csdn.net/u012796139/article/details/_blank) /
* [tinyxml基础](http://so.csdn.net/so/search/s.do?q=tinyxml基础&t=blog" \t "http://blog.csdn.net/u012796139/article/details/_blank) /
* 671
* [编辑](http://write.blog.csdn.net/postedit/46706853)
* 删除

        tinyXml使用文档对象模型（DOM）来解析xml文件，这种模型的处理方式为在分析时，一次性的将整个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，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，非常方便用户使用

        想了解关于XML介绍的的童鞋请移步到：[XML 新手入门基础知识](http://www.ibm.com/developerworks/cn/xml/x-newxml/" \l "list1" \t "http://blog.csdn.net/u012796139/article/details/_blank) （这里讲的XML基础很清楚的^\_^）

        想下载tinyxml源码的童鞋请移步到：[http://sourceforge.net/projects/tinyxml/](http://sourceforge.net/projects/tinyxml/" \t "http://blog.csdn.net/u012796139/article/details/_blank)

        刚使用timyxml进行xml文件的解析，不得不了解一下tinyxml中的几个类及其关系，下面是tinyxml中几个主要类的关系图(引用自tinyxml文档)



        想进一步了解tinyxml文档的童鞋可以查看tinyxml源码中的doc文件夹中的文档。

        TiXmlBase是所有类的基类，TiXmlNode、TiXmlAttribute两个类都继承来自TiXmlBase类，其中TiXmlNode类指的是所有被<...>...<.../>包括的内容，而xml中的节点又具体分为以下几方面内容，分别是声明、注释、节点以及节点间的文本，因此在TiXmlNode的基础上又衍生出这几个类TiXmlComment、TiXmlDeclaration、TiXmlDocument、TiXmlElement、TiXmlText、TiXmlUnknown，分别用来指明具体是xml中的哪一部分。TiXmlAttribute类不同于TiXmlNode，它指的是在尖括号里面的内容，像<... \*\*\*=...>，其中\*\*\*就是一个属性。

****以下是一个简单的xml文件示例****：

**[html]** [view plain](http://blog.csdn.net/u012796139/article/details/46706853" \o "view plain) [copy](http://blog.csdn.net/u012796139/article/details/46706853" \o "copy) [print](http://blog.csdn.net/u012796139/article/details/46706853" \o "print)[?](http://blog.csdn.net/u012796139/article/details/46706853" \o "?)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<phonebook>**
3. <!--one item behalfs one contacted person.-->
4. **<item>**
5. **<name>**miaomaio**</name>**
6. **<addr>**Shaanxi Xi'an**</addr>**
7. **<tel>**13759911917**</tel>**
8. **<email>**miaomiao@home.com**</email>**
9. **</item>**
10. **<item>**
11. **<name>**gougou**</name>**
12. **<addr>**Liaoning Shenyang**</addr>**
13. **<tel>**15840330481**</tel>**
14. **<email>**gougou@home.com**</email>**
15. **</item>**
16. <!--more contacted persons.-->
17. **</phonebook>**

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

<phonebook>

<!--one item behalfs one contacted person.-->

<item>

<name>miaomaio</name>

<addr>Shaanxi Xi'an</addr>

<tel>13759911917</tel>

<email>miaomiao@home.com</email>

</item>

<item>

<name>gougou</name>

<addr>Liaoning Shenyang</addr>

<tel>15840330481</tel>

<email>gougou@home.com</email>

</item>

<!--more contacted persons.-->

</phonebook>

l  像TiXmlDeclaration指的就是<?xml version="1.0"encoding="UTF-8"?>，

l  像TiXmlComment指的就是<!--one item behalfs onecontacted person.-->、 <!--more contacted persons.-->，

l  像TiXmlDocument指的就是整个xml文档，

l  像TiXmlElement指的就是<phonebook>、<item>、<name>、<addr>等等这些节点，

l  像TiXmlText指的就是‘gougou’、‘15840330481’这些夹在<item>与</item>、<name>与</name>、<addr>与</addr>之间的文本文字，

l  像TiXmlAttribute指的就是<?xml version="1.0"encoding="UTF-8"?>节点中version、encoding，

l  除此之外就是TiXmlUnknown。

****注意****：

**[html]** [view plain](http://blog.csdn.net/u012796139/article/details/46706853" \o "view plain) [copy](http://blog.csdn.net/u012796139/article/details/46706853" \o "copy) [print](http://blog.csdn.net/u012796139/article/details/46706853" \o "print)[?](http://blog.csdn.net/u012796139/article/details/46706853" \o "?)

1. **<phonebook>**
2. ……
3. **</phonebook>**

<phonebook>

……

</phonebook>

        这就是一个TiXmlNode，而……就是该TiXmlNode的TiXmlElement，TiXmlElement不包括TiXmlComment(注释类，它表示文件的注释部分)

下面是简单的****hello world程序****(不是输出hello world哟 IMG_257)

**[cpp]** [view plain](http://blog.csdn.net/u012796139/article/details/46706853" \o "view plain) [copy](http://blog.csdn.net/u012796139/article/details/46706853" \o "copy) [print](http://blog.csdn.net/u012796139/article/details/46706853" \o "print)[?](http://blog.csdn.net/u012796139/article/details/46706853" \o "?)

1. /\*
2. \* create a new xml file, it's name is phonebook.xml
3. \* timyxml version: 2.6.2
4. \* 程序新建一个空的doc文件，然后往该文件中添加一个phonebook节点，此时doc文件的根节点就是phonebook，
5. \* 向根节点中添加3项联系人信息(name、addr、tel、email)，然后打印出每项联系人的信息。
6. \* 接着删除其中一项联系人中的一条信息，比如第一项联系人的name信息，然后再删除一项联系人，接着打印出来
7. \*/
8. #include <iostream>
9. #include <sstream>
10. #include "tinyxml.h"
12. using namespace std;
14. /\*\*
15. \* 向root节点中插入name、addr、tel、email等节点
16. \*/
17. int PhoneBook\_InsertEndChild(TiXmlNode\* root, char\* \_name = "", char\* \_addr = "",
18. char\* \_tel = "", char\* \_email = "")
19. {
20. if (!root)
21. return -1;
23. TiXmlElement\* node = new TiXmlElement("item");
25. TiXmlNode\* name = new TiXmlElement("name");
26. node->InsertEndChild(\*name)->InsertEndChild(TiXmlText(\_name));
27. TiXmlNode\* addr = new TiXmlElement("addr");
28. node->InsertEndChild(\*addr)->InsertEndChild(TiXmlText(\_addr));
29. TiXmlNode\* tel = new TiXmlElement("tel");
30. node->InsertEndChild(\*tel)->InsertEndChild(TiXmlText(\_tel));
31. TiXmlNode\* email = new TiXmlElement("email");
32. node->InsertEndChild(\*email)->InsertEndChild(TiXmlText(\_email));
34. /\* 向root中添加该element，element中包含4个节点 \*/
35. root->InsertEndChild(\*node);
37. return 0;
38. }
40. int main(void)
41. {
42. const char\* str = "< ? xml version = \"1.0\" ?>";
43. TiXmlDocument doc("phonebook.xml");
45. doc.Parse(str);
46. TiXmlNode\* phoneBook = new TiXmlElement("phonebook");
47. doc.InsertEndChild(\*phoneBook); /\* 插入节点phonebook \*/
49. /\* 获取doc的根元素，即phonebook \*/
50. TiXmlNode\* root = doc.RootElement();
52. PhoneBook\_InsertEndChild(root, "hdu1", "hang zhou", "110", "hdu1@hdu.com");
53. PhoneBook\_InsertEndChild(root, "hdu2", "hang zhou", "111", "hdu2@hdu.com");
54. PhoneBook\_InsertEndChild(root, "hdu3", "hang zhou", "112", "hdu3@hdu.com");
56. doc.Print();
57. cout << endl;
59. /\* 依次遍历item，然后输出每个item中的节点(name,addr,tel,email)信息 \*/
60. for (TiXmlNode\* item = root->FirstChild(); item; item = item->NextSibling("item"))
61. {
62. cout << "-----------------------------" << endl;
63. for (TiXmlElement\* element = item->FirstChildElement(); element; element = element->NextSiblingElement())
64. {
65. if (!strcmp(element->Value(), "name"))
66. cout << "name: " << (element->GetText() ? element->GetText() : "") << endl;
67. else if (!strcmp(element->Value(), "addr"))
68. cout << "addr: " << (element->GetText() ? element->GetText() : "") << endl;
69. else if (!strcmp(element->Value(), "tel"))
70. cout << "tel: " << (element->GetText() ? element->GetText() : "") << endl;
71. else if (!strcmp(element->Value(), "email"))
72. cout << "email: " << (element->GetText() ? element->GetText() : "") << endl;
73. else
74. cout << "I don't know" << endl;
75. }
76. }
77. //doc.SaveFile(); /\* 取消注释后会该doc保存为phonebook.xml文件 \*/
79. TiXmlElement\* element = root->FirstChild()->FirstChildElement();
80. root->FirstChild()->RemoveChild(element); /\* 删除了第一项item节点中的 <name>hdu1</name> 节点 \*/
82. TiXmlNode\* node = root->LastChild();
83. root->RemoveChild(node);   /\* 删除了节点<item>
84. <name>hdu3</name>
85. <addr>hang zhou</addr>
86. <tel>112</tel>
87. <email>hdu3@hdu.com</email>
88. </item> \*/
90. doc.Print();
91. cout << endl;
93. for (TiXmlNode\* item = root->FirstChild(); item; item = item->NextSibling("item"))
94. {
95. cout << "-----------------------------" << endl;
96. for (TiXmlElement\* element = item->FirstChildElement(); element; element = element->NextSiblingElement())
97. {
98. if (!strcmp(element->Value(), "name"))
99. cout << "name: " << (element->GetText() ? element->GetText() : "") << endl;
100. else if (!strcmp(element->Value(), "addr"))
101. cout << "addr: " << (element->GetText() ? element->GetText() : "") << endl;
102. else if (!strcmp(element->Value(), "tel"))
103. cout << "tel: " << (element->GetText() ? element->GetText() : "") << endl;
104. else if (!strcmp(element->Value(), "email"))
105. cout << "email: " << (element->GetText() ? element->GetText() : "") << endl;
106. else
107. cout << "I don't know" << endl;
108. }
109. }
111. return 0;
112. }

/\*

\* create a new xml file, it's name is phonebook.xml

\* timyxml version: 2.6.2

\* 程序新建一个空的doc文件，然后往该文件中添加一个phonebook节点，此时doc文件的根节点就是phonebook，

\* 向根节点中添加3项联系人信息(name、addr、tel、email)，然后打印出每项联系人的信息。

\* 接着删除其中一项联系人中的一条信息，比如第一项联系人的name信息，然后再删除一项联系人，接着打印出来

\*/

#include <iostream>

#include <sstream>

#include "tinyxml.h"

using namespace std;

/\*\*

\* 向root节点中插入name、addr、tel、email等节点

\*/

int PhoneBook\_InsertEndChild(TiXmlNode\* root, char\* \_name = "", char\* \_addr = "",

char\* \_tel = "", char\* \_email = "")

{

if (!root)

return -1;

TiXmlElement\* node = new TiXmlElement("item");

TiXmlNode\* name = new TiXmlElement("name");

node->InsertEndChild(\*name)->InsertEndChild(TiXmlText(\_name));

TiXmlNode\* addr = new TiXmlElement("addr");

node->InsertEndChild(\*addr)->InsertEndChild(TiXmlText(\_addr));

TiXmlNode\* tel = new TiXmlElement("tel");

node->InsertEndChild(\*tel)->InsertEndChild(TiXmlText(\_tel));

TiXmlNode\* email = new TiXmlElement("email");

node->InsertEndChild(\*email)->InsertEndChild(TiXmlText(\_email));

/\* 向root中添加该element，element中包含4个节点 \*/

root->InsertEndChild(\*node);

return 0;

}

int main(void)

{

const char\* str = "< ? xml version = \"1.0\" ?>";

TiXmlDocument doc("phonebook.xml");

doc.Parse(str);

TiXmlNode\* phoneBook = new TiXmlElement("phonebook");

doc.InsertEndChild(\*phoneBook); /\* 插入节点phonebook \*/

/\* 获取doc的根元素，即phonebook \*/

TiXmlNode\* root = doc.RootElement();

PhoneBook\_InsertEndChild(root, "hdu1", "hang zhou", "110", "hdu1@hdu.com");

PhoneBook\_InsertEndChild(root, "hdu2", "hang zhou", "111", "hdu2@hdu.com");

PhoneBook\_InsertEndChild(root, "hdu3", "hang zhou", "112", "hdu3@hdu.com");

doc.Print();

cout << endl;

/\* 依次遍历item，然后输出每个item中的节点(name,addr,tel,email)信息 \*/

for (TiXmlNode\* item = root->FirstChild(); item; item = item->NextSibling("item"))

{

cout << "-----------------------------" << endl;

for (TiXmlElement\* element = item->FirstChildElement(); element; element = element->NextSiblingElement())

{

if (!strcmp(element->Value(), "name"))

cout << "name: " << (element->GetText() ? element->GetText() : "") << endl;

else if (!strcmp(element->Value(), "addr"))

cout << "addr: " << (element->GetText() ? element->GetText() : "") << endl;

else if (!strcmp(element->Value(), "tel"))

cout << "tel: " << (element->GetText() ? element->GetText() : "") << endl;

else if (!strcmp(element->Value(), "email"))

cout << "email: " << (element->GetText() ? element->GetText() : "") << endl;

else

cout << "I don't know" << endl;

}

}

//doc.SaveFile(); /\* 取消注释后会该doc保存为phonebook.xml文件 \*/

TiXmlElement\* element = root->FirstChild()->FirstChildElement();

root->FirstChild()->RemoveChild(element); /\* 删除了第一项item节点中的 <name>hdu1</name> 节点 \*/

TiXmlNode\* node = root->LastChild();

root->RemoveChild(node); /\* 删除了节点<item>

<name>hdu3</name>

<addr>hang zhou</addr>

<tel>112</tel>

<email>hdu3@hdu.com</email>

</item> \*/

doc.Print();

cout << endl;

for (TiXmlNode\* item = root->FirstChild(); item; item = item->NextSibling("item"))

{

cout << "-----------------------------" << endl;

for (TiXmlElement\* element = item->FirstChildElement(); element; element = element->NextSiblingElement())

{

if (!strcmp(element->Value(), "name"))

cout << "name: " << (element->GetText() ? element->GetText() : "") << endl;

else if (!strcmp(element->Value(), "addr"))

cout << "addr: " << (element->GetText() ? element->GetText() : "") << endl;

else if (!strcmp(element->Value(), "tel"))

cout << "tel: " << (element->GetText() ? element->GetText() : "") << endl;

else if (!strcmp(element->Value(), "email"))

cout << "email: " << (element->GetText() ? element->GetText() : "") << endl;

else

cout << "I don't know" << endl;

}

}

return 0;

}

在没有删除任何元素时，phonebook.xml文件内容为：

**[html]** [view plain](http://blog.csdn.net/u012796139/article/details/46706853" \o "view plain) [copy](http://blog.csdn.net/u012796139/article/details/46706853" \o "copy) [print](http://blog.csdn.net/u012796139/article/details/46706853" \o "print)[?](http://blog.csdn.net/u012796139/article/details/46706853" \o "?)

1. **<** ? xml version = "1.0" **?>**
2. **<phonebook>**
3. **<item>**
4. **<name>**hdu1**</name>**
5. **<addr>**hang zhou**</addr>**
6. **<tel>**110**</tel>**
7. **<email>**hdu1@hdu.com**</email>**
8. **</item>**
9. **<item>**
10. **<name>**hdu2**</name>**
11. **<addr>**hang zhou**</addr>**
12. **<tel>**111**</tel>**
13. **<email>**hdu2@hdu.com**</email>**
14. **</item>**
15. **<item>**
16. **<name>**hdu3**</name>**
17. **<addr>**hang zhou**</addr>**
18. **<tel>**112**</tel>**
19. **<email>**hdu3@hdu.com**</email>**
20. **</item>**
21. **</phonebook>**

< ? xml version = "1.0" ?>

<phonebook>

<item>

<name>hdu1</name>

<addr>hang zhou</addr>

<tel>110</tel>

<email>hdu1@hdu.com</email>

</item>

<item>

<name>hdu2</name>

<addr>hang zhou</addr>

<tel>111</tel>

<email>hdu2@hdu.com</email>

</item>

<item>

<name>hdu3</name>

<addr>hang zhou</addr>

<tel>112</tel>

<email>hdu3@hdu.com</email>

</item>

</phonebook>

删除了一个item节点项和第一个item节点中name项后的phonebook.xml文件内容

**[html]** [view plain](http://blog.csdn.net/u012796139/article/details/46706853" \o "view plain) [copy](http://blog.csdn.net/u012796139/article/details/46706853" \o "copy) [print](http://blog.csdn.net/u012796139/article/details/46706853" \o "print)[?](http://blog.csdn.net/u012796139/article/details/46706853" \o "?)

1. **<** ? xml version = "1.0" **?>**
2. **<phonebook>**
3. **<item>**
4. **<addr>**hang zhou**</addr>**
5. **<tel>**110**</tel>**
6. **<email>**hdu1@hdu.com**</email>**
7. **</item>**
8. **<item>**
9. **<name>**hdu2**</name>**
10. **<addr>**hang zhou**</addr>**
11. **<tel>**111**</tel>**
12. **<email>**hdu2@hdu.com**</email>**
13. **</item>**
14. **</phonebook>**

< ? xml version = "1.0" ?>

<phonebook>

<item>

<addr>hang zhou</addr>

<tel>110</tel>

<email>hdu1@hdu.com</email>

</item>

<item>

<name>hdu2</name>

<addr>hang zhou</addr>

<tel>111</tel>

<email>hdu2@hdu.com</email>

</item>

</phonebook>

参考资料：

        1、[XML新手入门基础知识](http://www.ibm.com/developerworks/cn/xml/x-newxml/" \l "list1" \t "http://blog.csdn.net/u012796139/article/details/_blank)

        2、[C++之tinyXML使用](http://blog.csdn.net/L_Andy/article/details/40615517" \t "http://blog.csdn.net/u012796139/article/details/_blank)  
        3、[tinyxml源码下载地址](http://sourceforge.net/projects/tinyxml/" \t "http://blog.csdn.net/u012796139/article/details/_blank)