

Qt 史上最详细 - “操作XML文件” - 笔记

原创

cpp_learners

于 2021-06-13 17:20:04 发布 10211 收藏 219

分类专栏:

Qt

文章标签:

xml

qt

超星计划

欢迎使用 CSDNGreener, 绿化设定按钮在这里!
好的, 以后不再提示我

版权



Qt 专栏收录该内容

47 订阅

33 篇文章

订阅专栏

XML 文件可以用来存储项目中的数据, 它相当于一个简单的数据库。

XML文件在实际开发中可以经常使用的, 因为它非常方便的存储数据, 方便存储、方便读取、方便查看、方便管理等。

下面将使用QT语言编写代码操作XML文件 - 增删改查。

目录

一、准备工作

二、创建

1. 创建一个xml文件
2. 创建一个XML类
3. 创建XML头部格式
4. 创建根节点
5. 添加子节点
6. 添加带属性的子节点
7. 将doc写入xml文件
8. 创建xml文件并插入元素节点, 总代码如下

三、添加

1. 打开文件, 并定义xml文档类, 并关联file文件
2. 获取根节点
3. 创建一级子节点, 并设置属性
4. 为一级子节点创建子节点
5. 将一级子节点添加到根节点中
6. 使用文件流写入xml文件中
7. 为已存在的xml文件添加节点元素, 总代码如下

四、删除

1. 与上类似, 打开文件, 关联文件, 获取根节点
2. 删除Book节点
3. 删除指定的节点
4. 最后再使用文件流输出到文件即可
5. 删除节点, 代码汇总

五、修改

1. 修改属性值
2. 修改节点值
3. 修改xml中的节点值, 代码汇总

六、读取

1. 首先打开文件, 关联对象
2. 获取属性中的值
3. 获取尖括号间的值
4. 最后记得关闭文件哦
5. 读取xml节点值, 代码汇总

七、总结

一、准备工作

首先看一下XML文件的组成部分



描述信息：这个是xml文件必须要有一段描述语句；

根节点：xml文件由只有一个根节点（仅有），它是所有子节点的父节点；

子节点：可有多个，必须在根节点内；

属性：存储数据的一种方式。

Qt中提供了QtXml模块来进行xml文件的处理，其中它有3中方式进行操作Xml文件；这里主要讲解DOM方式。

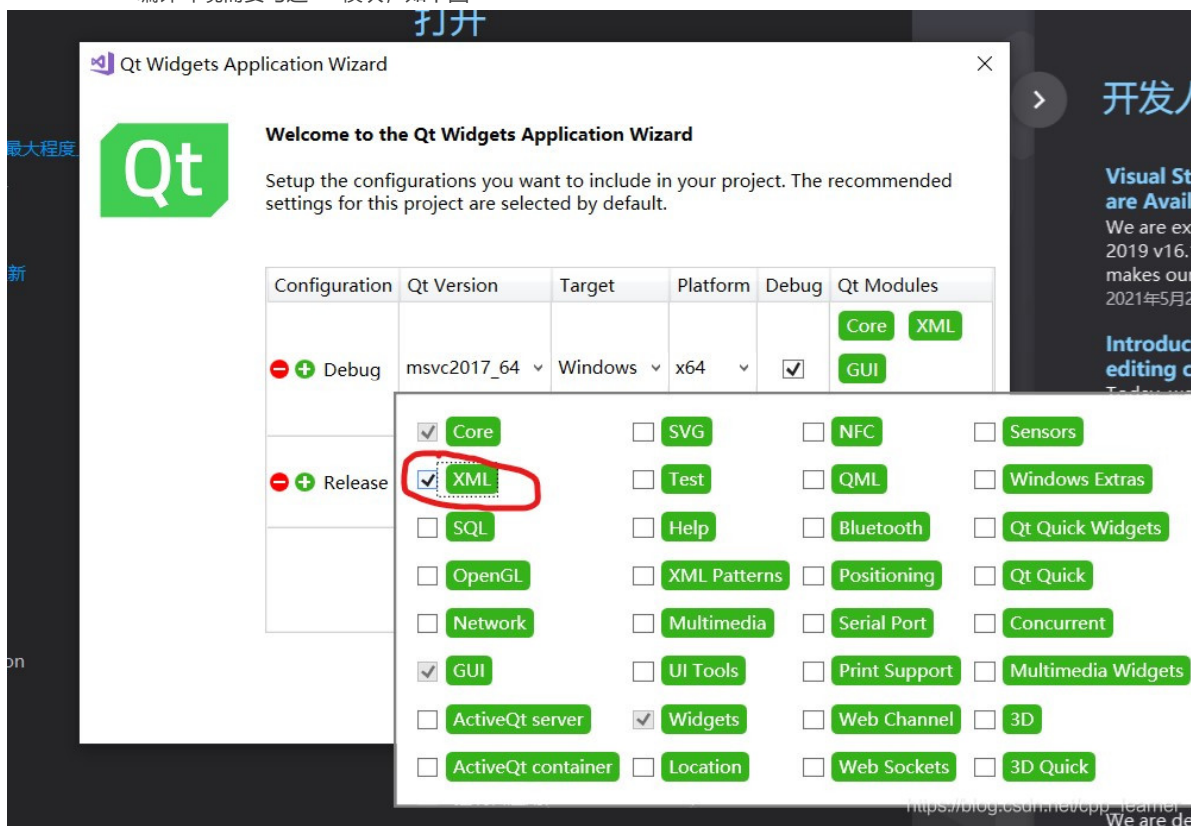
操作xml文件需要添加头文件：

```
#include < QtXml >
```

```
#include < QDomDocument >
```

并且在创建项目时：

1. Visual Studio 编译环境需要勾选xml模块，如下图



2. QT编译环境需要向.pro文件里面添加QT+=xml

二、创建

创建XML文件,并写入内容

```
1 // 打开或者创建一个XML文件
2 QFile file("QT_XML.xml");
3 // 文件存在则创建, 否则创建一个文件
4 if (!file.open(QIODevice::ReadWrite | QIODevice::Truncate)) {
5     QMessageBox::information(NULL, "提示", "文件打开或创建失败!");
6     return;
7 }
```

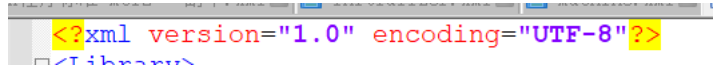
2. 创建一个XML类

```
1 QDomDocument doc;
```

3. 创建XML头部格式

```
1 // 创建XML处理类, 通常用于处理第一行描述信息
2 QDomProcessingInstruction instruction;
3
4 // 创建XML头部格式
5 instruction = doc.createProcessingInstruction("xml", "version=\"1.0\" encoding=\"UTF-8\"");
6
7 // 添加到XML文件中
8 doc.appendChild(instruction);
```

如下图, 就是创建这么一行



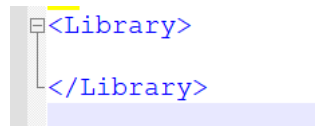
```
<?xml version="1.0" encoding="UTF-8"?>
```

4. 创建根节点

使用createElement方法进行创建, 创建完成后使用appendChild方法添加到xml文档中

```
1 // 创建根节点
2 QDomElement root = doc.createElement("Library");
3 doc.appendChild(root);
```

如下图



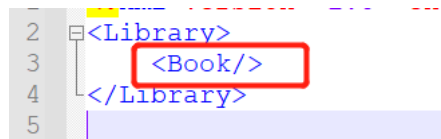
```
<Library>
</Library>
```

5. 添加子节点

这里只是单纯的添加一个子节点, 并没有其他元素

```
1 QDomElement book = doc.createElement("Book");
2 root.appendChild(book);
```

如下图



```
<Library>
  <Book/>
</Library>
```

6. 添加带属性的子节点

使用setAttribute可以简单的给节点添加属性值; 使用createElement是创建一个节点元素;

```
1 QDomElement book1 = doc.createElement("Book1");
2
3 // 给节点创建属性
4 book1.setAttribute("ID", 1);
5 book1.setAttribute("Name", "水浒传"); // 参数一是字符串, 参数二可以是任何类型
6 book1.setAttribute("Price", 64.5);
7
```

```
10
11 // 设置尖括号中的值
12 QDomText text = doc.createTextNode("108个拆迁户");
13
14 // 添加关系
15 description.appendChild(text);
16 book1.appendChild(description);
17
18 // 添加子元素2
19 QDomElement Page = doc.createElement("Page");
20 QDomText text2 = doc.createTextNode("100页");
21 Page.appendChild(text2);
22 book1.appendChild(Page);
23
24 // 添加到根节点中
25 root.appendChild(book1);
```

流程:

首先创建一个节点, 为该节点设置属性值, 然后在创建两个子节点, 并设置尖括号间的值, 最后他们两被添加到第一个子节点中, 然后第一个子节点被添加到根节点中, 形成一种树结构。如下图

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <Library>
3   <Book/>
4   <Book1 Name="水浒传" Price="64.5" ID="1">
5     <Description>108个拆迁户</Description>
6     <Page>100页</Page>
7   </Book1>
8 </Library>
9
```

https://blog.csdn.net/cpp_learner

7. 将doc写入xml文件

需要使用到文件流QTextStream

```
1 QTextStream stream(&file);
2 doc.save(stream, 4); // 缩进四格
3
4 file.close();
```

8. 创建xml文件并插入元素节点, 总代码如下

```
1 // 创建XML文件, 并写入内容
2 void createXML() {
3     // 打开或者创建一个XML文件
4     QFile file("QT_XML.xml");
5     // 文件存在则创建, 否则创建一个文件
6     if (!file.open(QIODevice::ReadWrite | QIODevice::Truncate)) {
7         QMessageBox::information(NULL, "提示", "文件打开或创建失败!");
8         return;
9     }
10
11     // 创建一个XML类
12     QDomDocument doc;
13
14     // 创建XML处理类, 通常用于处理第一行描述信息
15     QDomProcessingInstruction instruction;
16
17     // 创建XML头部格式
18     instruction = doc.createProcessingInstruction("xml", "version=\"1.0\" encoding=\"UTF-8\"");
19
20     // 添加到XML文件中
21     doc.appendChild(instruction);
22
23     // 创建根节点
24     QDomElement root = doc.createElement("Library");
25     doc.appendChild(root);
```

```
28
29 // 添加子节点
30 QDomElement book = doc.createElement("Book");
31 root.appendChild(book);
32
33 /*****
34
35 // 添加带属性的子节点
36 QDomElement book1 = doc.createElement("Book1");
37
38 // 给节点创建属性
39 book1.setAttribute("ID", 1);
40 book1.setAttribute("Name", "水浒传"); // 参数一是字符串, 参数二可以是任何类型
41 book1.setAttribute("Price", 64.5);
42
43 // 创建子元素
44 QDomElement description = doc.createElement("Description");
45
46 // 设置尖括号中的值
47 QDomText text = doc.createTextNode("108个拆迁户");
48
49 // 添加关系
50 description.appendChild(text);
51 book1.appendChild(description);
52
53 // 添加子元素2
54 QDomElement Page = doc.createElement("Page");
55 QDomText text2 = doc.createTextNode("100页");
56 Page.appendChild(text2);
57 book1.appendChild(Page);
58
59 // 添加到根节点中
60 root.appendChild(book1);
61
62 /*****
63
64 // 添加子节点3
65 QDomElement book3 = doc.createElement("Book1");
66 book3.setAttribute("ID", 2);
67 book3.setAttribute("Name", "三国演义");
68 book3.setAttribute("Price", 66.5);
69 QDomElement description2 = doc.createElement("Description");
70 QDomText text3 = doc.createTextNode("魏蜀吴打架");
71 description2.appendChild(text3);
72 book3.appendChild(description2);
73 QDomElement page = doc.createElement("Page");
74 QDomText text4 = doc.createTextNode("300页");
75 page.appendChild(text4);
76
77 book3.appendChild(page);
78 root.appendChild(book3);
79
80 /*****
81
82 // 将其写入到xml文件中
83 QTextStream stream(&file);
84 doc.save(stream, 4); // 缩进四格
85
86 file.close();
87 }
```

当执行完成后, 会在项目路径创建一个xml文件, 内容如下图:

```
3      <Book/>
4      <Book1 Name="水浒传" Price="64.5" ID="1">
5          <Description>108个拆迁户</Description>
6          <Page>100页</Page>
7      </Book1>
8      <Book1 Name="三国演义" Price="66.5" ID="2">
9          <Description>魏蜀吴打架</Description>
10         <Page>300页</Page>
11     </Book1>
12 </Library>
```

https://blog.csdn.net/cpp_learner

三、添加

往xml文件中添加节点元素

例:

/* 创建一个一级子节点，子节点包含属性，包含二级子节点 */

1. 打开文件，并定义xml文档类，并关联file文件

简单来说，对已经存在的xml文件，我们操作它前，需要先使用xml对象与文件对象进行关联

```
1 // 打开文件
2 QFile file("QT_XML.xml");
3 if (!file.open(QIODevice::ReadOnly)) {
4     QMessageBox::information(NULL, "提示", "文件打开失败!");
5     return;
6 }
7
8 QDomDocument doc;
9
10 // 将doc与file关联起来
11 // 这个函数从IO设备dev中读取XML文档，如果内容被成功解析，则返回true;否则返回false。
12 if (!doc.setContent(&file)) {
13     QMessageBox::information(NULL, "提示", "操作的文件不是XML文件!");
14     file.close();
15     return;
16 }
17 // 关联后可以关闭文件了
18 file.close();
```

2. 获取根节点

获取根节点对象

```
1 QDomElement root = doc.documentElement();
```

3. 创建一级子节点，并设置属性

使用createElement可以创建子节点元素；使用setAttribute可以设置属性

```
1 QDomElement book = doc.createElement("Book1");
2 book.setAttribute("ID", 3); // 参数1是字符串，参数2可以是任何类型
3 book.setAttribute("Name", "西游记");
4 book.setAttribute("Price", "68.5");
```

假设创建好后，在xml文件中是这样显示的：

```
3      <Book/>
4      <Book1 ID="1" Name="水浒传" Price="64.5">
5          <Description>108个拆迁户</Description>
6          <Page>100页</Page>
7      </Book1>
8      <Book1 ID="2" Name="三国演义" Price="66.5">
9          <Description>魏蜀吴打架</Description>
10         <Page>300页</Page>
11     </Book1>
12     <Book1 ID="3" Name="西游记" Price="68.5">
13     </Book1>
14 </Library>
15
```

https://blog.csdn.net/cpp_learner

4. 为一级子节点创建子节点

```
1 QDomElement description = doc.createElement("Description"); // 创建节点
2 QDomText text = doc.createTextNode("两个人，一匹马，两个动物"); // 创建节点尖括号间的值
3 description.appendChild(text); // 尖括号间的值的对象被添加到2级子节点中
4 book.appendChild(description); // 2级子节点被添加到1级子节点中
5
6 QDomElement page = doc.createElement("Page");
7 text = doc.createTextNode("81页");
8 page.appendChild(text);
9 book.appendChild(page);
```

假设创建好后，在xml文件中是这样显示的：

```
1 <?xml version='1.0' encoding='UTF-8'?>
2 <Library>
3     <Book/>
4     <Book1 ID="1" Name="水浒传" Price="64.5">
5         <Description>108个拆迁户</Description>
6         <Page>100页</Page>
7     </Book1>
8     <Book1 ID="2" Name="三国演义" Price="66.5">
9         <Description>魏蜀吴打架</Description>
10        <Page>300页</Page>
11    </Book1>
12    <Book1 ID="3" Name="西游记" Price="68.5">
13        <Description>两个人，一匹马，两个动物</Description>
14        <Page>81页</Page>
15    </Book1>
16 </Library>
```

https://blog.csdn.net/cpp_learner

5. 将一级子节点添加到根节点中

记得，当子节点们操作完成后，需要添加到根节点中，否则是无法添加成功的

```
1 // 添加到根节点
2 root.appendChild(book);
```

6. 使用文件流写入xml文件中

```
1 if (!file.open(QIODevice::WriteOnly | QIODevice::Truncate)) {
2     QMessageBox::information(NULL, "提示", "文件打开失败!");
3     return;
4 }
5
6 // 输出到文件
7 QTextStream stream(&file);
8 doc.save(stream, 4); // 缩进4格
9 file.close();
```

7. 为已存在的xml文件添加节点元素，总代码如下


```
3  /* 创建一个一级子节点, 子节点包含属性, 包含二级子节点*/
4
5  // 打开文件
6  QFile file("QT_XML.xml");
7  if (!file.open(QIODevice::ReadOnly)) {
8      QMessageBox::information(NULL, "提示", "文件打开失败!");
9      return;
10 }
11
12 QDomDocument doc;
13
14 // 将doc与file关联起来
15 // 这个函数从IO设备dev中读取XML文档, 如果内容被成功解析, 则返回true;否则返回false。
16 if (!doc.setContent(&file)) {
17     QMessageBox::information(NULL, "提示", "操作的文件不是XML文件!");
18     file.close();
19     return;
20 }
21 file.close();
22
23 // 获取根节点
24 QDomElement root = doc.documentElement();
25
26 // 创建一级节点
27 QDomElement book = doc.createElement("Book1");
28 book.setAttribute("ID", 3);
29 book.setAttribute("Name", "西游记");
30 book.setAttribute("Price", "68.5");
31
32 // 创建二级节点1
33 QDomElement description = doc.createElement("Description");
34 QDomText text = doc.createTextNode("两个人, 一匹马, 两个动物");
35 description.appendChild(text);
36 book.appendChild(description);
37 QDomElement page = doc.createElement("Page");
38 text = doc.createTextNode("81页");
39 page.appendChild(text);
40 book.appendChild(page);
41
42 // 创建二级节点2,3,4
43 //QDomElement author2 = doc.createElement("Author");
44 //author2.setAttribute("Name", "吴承恩");
45 //QDomElement author3 = doc.createElement("Author");
46 //author3.setAttribute("Sex", "男");
47 //QDomElement author4 = doc.createElement("Author");
48 //author4.setAttribute("Age", 82);
49 //book.appendChild(author2);
50 //book.appendChild(author3);
51 //book.appendChild(author4);
52
53 // 添加到根节点
54 root.appendChild(book);
55
56
57
58 if (!file.open(QIODevice::WriteOnly | QIODevice::Truncate)) {
59     QMessageBox::information(NULL, "提示", "文件打开失败!");
60     return;
61 }
62
63 // 输出到文件
64 QTextStream stream(&file);
65 doc.save(stream, 4); // 缩进4格
66 file.close();
67 }
```

执行完如上代码, 会在xml文件中添加指定的节点元素, 如下图


```
3      <Book/>
4      <Book1 ID="1" Name="水浒传" Price="64.5">
5          <Description>108个拆迁户</Description>
6          <Page>100页</Page>
7      </Book1>
8      <Book1 ID="2" Name="三国演义" Price="66.5">
9          <Description>魏蜀吴打架</Description>
10         <Page>300页</Page>
11     </Book1>
12     <Book1 ID="3" Name="西游记" Price="68.5">
13         <Description>两个人，一匹马，两个动物</Description>
14         <Page>81页</Page>
15     </Book1>
16 </Library>
```

https://blog.csdn.net/cpp_learner

四、删除

删除子节点

例：

/* 删除book 和 book1其中一个子节点 */

1. 与上类似，打开文件，关联文件，获取根节点

...

2. 删除Book节点

```
1 // 获取节点名字为Book的所有节点
2 QDomNodeList list = doc.elementsByTagName("Book");
3 for (int i = 0; i < list.size(); i++) {
4     // 获取节点
5     QDomElement element = list.at(i).toElement();
6     // 删除子节点
7     root.removeChild(element);
8 }
```

流程：

首先通过elementsByTagName获取名字为“Book”的所有节点；遍历，获取其中每一个元素，转换为QDomElement 类型，进行删除

假设执行完，xml中将会是这样：

```
1 <?xml version='1.0' encoding='UTF-8'?>
2 <Library>
3     <Book1 ID="1" Price="64.5" Name="水浒传">
4         <Description>108个拆迁户</Description>
5         <Page>100页</Page>
6     </Book1>
7     <Book1 ID="2" Price="66.5" Name="三国演义">
8         <Description>魏蜀吴打架</Description>
9         <Page>300页</Page>
10    </Book1>
11    <Book1 ID="3" Price="68.5" Name="西游记">
12        <Description>两个人，一匹马，两个动物</Description>
13        <Page>81页</Page>
14    </Book1>
15 </Library>
```

Book节点
被删除了

https://blog.csdn.net/cpp_learner

3. 删除指定的节点

上面是将节点名字为Book的节点都删除；那么，如果我只是向删除其中一个呢？

这里可以进行判断一下，在进行删除，使用属性的唯一值进行判断即可

```
1 list = doc.elementsByTagName("Book1");
2 int size = list.count();
3 for (int i = 0; i < list.count(); i++) {
```

```
6
7 // 进行判断 (返回属性的值进行判断)
8 if (element.attribute("ID") == "3") {
9     root.removeChild(list.at(i));
10 }
11 }
```

这里使用attribute方法获取对应参数的属性的值，然后与指定字符串进行判断，相等则使用removeChild方法进行删除

```
1 <?xml version='1.0' encoding='UTF-8'?>
2 <Library>
3   <Book1 ID="1" Name="水浒传" Price="64.5">
4     <Description>108个拆迁户</Description>
5     <Page>100页</Page>
6   </Book1>
7   <Book1 ID="2" Name="三国演义" Price="66.5">
8     <Description>魏蜀吴打架</Description>
9     <Page>300页</Page>
10  </Book1>
11 </Library>
12
```

ID为3的节点被删除了

4. 最后再使用文件流输出到文件即可

与上面是相同的...

5. 删除节点，代码汇总

```
1 // 删除XML文件中的某些内容
2 void delXML() {
3     /* 删除book 和 book1的一个子节点*/
4
5     // 打开文件
6     QFile file("QT_XML.xml");
7     if (!file.open(QIODevice::ReadOnly)) {
8         QMessageBox::information(NULL, "提示", "文件打开失败!");
9         return;
10    }
11
12    QDomDocument doc;
13    if (!doc.setContent(&file)) {
14        QMessageBox::information(NULL, "提示", "操作的文件不是XML文件!");
15        file.close();
16        return;
17    }
18    file.close();
19
20
21    // 获得根节点
22    QDomElement root = doc.documentElement();
23
24
25    /* 删除所有的Book节点 */
26    // 获取节点名字为Book的所有节点
27    QDomNodeList list = doc.elementsByTagName("Book");
28    for (int i = 0; i < list.size(); i++) {
29        // 获取节点
30        QDomElement element = list.at(i).toElement();
31        // 删除子节点
32        root.removeChild(element);
33    }
34
35    // 获取节点名字为Book1的所有节点
36    list = doc.elementsByTagName("Book1");
37    int size = list.count();
38    for (int i = 0; i < list.count(); i++) {
39        // 获取节点
40        QDomElement element = list.at(i).toElement();
41    }
```

```
44     }
45 }
46
47
48 if (!file.open(QIODevice::WriteOnly | QIODevice::Truncate)) {
49     QMessageBox::information(NULL, "提示", "文件打开失败!");
50     return;
51 }
52
53 // 输出到文件
54 QTextStream stream(&file);
55 doc.save(stream, 4);    // 缩进4格
56 file.close();
57 }
```

执行后xml文件如下图:

```
1  <?xml version='1.0' encoding='UTF-8'?>
2  <Library>
3      <Book1 ID="1" Name="水浒传" Price="64.5">
4          <Description>108个拆迁户</Description>
5          <Page>100页</Page>
6      </Book1>
7      <Book1 ID="2" Name="三国演义" Price="66.5">
8          <Description>魏蜀吴打架</Description>
9          <Page>300页</Page>
10     </Book1>
11 </Library>
```

https://blog.csdn.net/cpp_learner

当然你也可以删除子节点的子节点, 方法类似, 定位到对应节点然后使用父节点removeChild就可以了。

QDomNode QDomNode::parentNode() const此方法可以返回父节点。

五、修改

修改(更改)XML中节点的值

例:

/* 将Book1中id为2的节点的Price属性更改数值为88.5; 修改其子节点Description内容为 "魏蜀吴在玩过家家" */

1. 修改属性值

```
1 // 获取所有Book1节点
2 QDomNodeList list = root.elementsByTagName("Book1");
3
4 /* 修改属性的值 */
5 for (int i = 0; i < list.size(); i++) {
6     QDomElement element = list.at(i).toElement();
7
8     // 找到ID等于2的节点
9     if (element.attribute("ID") == "2") {
10         // 修改属性值
11         element.setAttribute("Price", 88.5);
12         break;
13     }
14 }
```

当执行完代码后:

```
3  <Book1 Price="64.5" Name="水浒传" ID="1">
4      <Description>108个拆迁户</Description>
5      <Page>100页</Page>
6  </Book1>
7  <Book1 Price="88.5" Name="三国演义" ID="2">
8      <Description>魏蜀吴打架</Description>
9      <Page>300页</Page>
10 </Book1>
11 </Library>
```

已经由86.5修改为88.5

2. 修改节点值

这个可能有点复杂

首先当然是先获取到待修改节点对象，然后通过调用firstChild方法，将尖括号间的内容实例化为QDomNode对象，然后待修改节点对象调用firstChild().setNodeValue()设置尖括号中的值，然后再次将尖括号中的内容实例化为对象，最后，调用replaceChild函数将两个尖括号的对象互相替换。

看字面有点抽象，没事，看代码应该就会懂了

```
1  for (int i = 0; i < list.size(); i++) {
2      QDomElement element = list.at(i).toElement();
3
4      // 找到ID等于2的节点
5      if (element.attribute("ID") == "2") {
6          // 获得子节点
7          QDomNode node = element.namedItem("Description");
8
9          // 尖括号之间的内容作为子节点出现(修改前)
10         QDomNode oldNode = node.firstChild();
11
12         // 修改尖括号之间的值
13         node.firstChild().setNodeValue("魏蜀吴在玩过家家");
14
15         // 尖括号之间的内容作为子节点出现(修改后)
16         QDomNode newNode = node.firstChild();
17
18         // 将新旧内容子节点进行替换
19         node.replaceChild(newNode, oldNode);
20         break;
21     }
22 }
```

解释：

首先获取需要修改尖括号间的值的对象

// 获得子节点

QDomNode node = element.namedItem("Description");

也就是下图方框中的对象

```
1  <?xml version='1.0' encoding='UTF-8'?>
2  <Library>
3      <Book1 Price="64.5" Name="水浒传" ID="1">
4          <Description>108个拆迁户</Description>
5          <Page>100页</Page>
6      </Book1>
7      <Book1 Price="88.5" Name="三国演义" ID="2">
8          <Description>魏蜀吴打架</Description>
9          <Page>300页</Page>
10 </Book1>
11 </Library>
```

https://blog.csdn.net/cpp_learner

然后将尖括号间的内容实例化为对象

// 尖括号之间的内容作为子节点出现(修改前)

QDomNode oldNode = node.firstChild();

也即是下图方框中的对象

```
3 <Book1 Price="64.5" Name="水浒传" ID="1">
4     <Description>108个拆迁户</Description>
5     <Page>100页</Page>
6 </Book1>
7 <Book1 Price="88.5" Name="三国演义" ID="2">
8     <Description>魏蜀吴打架</Description>
9     <Page>300页</Page>
10 </Book1>
11 </Library>
```

https://blog.csdn.net/cpp_learner

然后修改尖括号间的值

```
node.firstChild().setNodeValue("魏蜀吴在玩过家家");
```

当然，修改后，并不是值立马就改变了，还需要做其他操作。

然后再次将尖括号间的内容实例化为对象

```
// 尖括号之间的内容作为子节点出现(修改后)
```

```
QDomNode newNode = node.firstChild();
```

最后将修改前的节点和修改后的节点进行替换

```
// 将新旧内容子节点进行替换
```

```
node.replaceChild(newNode, oldNode);
```

得出如下效果：

```
1 <?xml version='1.0' encoding='UTF-8'?>
2 <Library>
3     <Book1 Name="水浒传" Price="64.5" ID="1">
4         <Description>108个拆迁户</Description>
5         <Page>100页</Page>
6     </Book1>
7     <Book1 Name="三国演义" Price="88.5" ID="2">
8         <Description>魏蜀吴在玩过家家</Description>
9         <Page>300页</Page>
10    </Book1>
11 </Library>
```

https://blog.csdn.net/cpp_learner

最后使用文件流写入xml文件即可。

3. 修改xml中的节点值，代码汇总

```
1 // 更新XML中的内容
2 void altXML() {
3     /* 将Book1中id为2的节点的Price属性更改数值为88.5; 修改其子节点Description内容为“魏蜀吴在玩过家家” */
4
5     // 打开文件
6     QFile file("QT_XML.xml");
7     if (!file.open(QIODevice::ReadOnly)) {
8         QMessageBox::information(NULL, "提示", "文件打开失败!");
9         return;
10    }
11
12    QDomDocument doc;
13    if (!doc.setContent(&file)) {
14        QMessageBox::information(NULL, "提示", "操作的文件不是XML文件!");
15        file.close();
16        return;
17    }
18    file.close();
19
20
21    // 获得根节点
22    QDomElement root = doc.documentElement();
23
24    // 获取所有Book1节点
25    QDomNodeList list = root.elementsByTagName("Book1");
26
27    /* 修改属性的值 */
```

```
30
31 // 找到ID等于2的节点
32 if (element.attribute("ID") == "2") {
33     // 修改属性值
34     element.setAttribute("Price", 88.5);
35     break;
36 }
37 }
38
39
40 /* 修改尖括号之间的值 */
41 for (int i = 0; i < list.size(); i++) {
42     QDomElement element = list.at(i).toElement();
43
44     // 找到ID等于2的节点
45     if (element.attribute("ID") == "2") {
46         // 获得子节点
47         QDomNode node = element.namedItem("Description");
48
49         // 尖括号之间的内容作为子节点出现(修改前)
50         QDomNode oldNode = node.firstChild();
51
52         // 修改尖括号之间的值
53         node.firstChild().setNodeValue("魏蜀吴在玩过家家");
54
55         // 尖括号之间的内容作为子节点出现(修改后)
56         QDomNode newNode = node.firstChild();
57
58         // 将新旧内容子节点进行替换
59         node.replaceChild(newNode, oldNode);
60         break;
61     }
62 }
63
64
65
66 if (!file.open(QIODevice::WriteOnly | QIODevice::Truncate)) {
67     QMessageBox::information(NULL, "提示", "文件打开失败! ");
68     return;
69 }
70
71 // 输出到文件
72 QTextStream stream(&file);
73 doc.save(stream, 4); // 缩进4格
74 file.close();
75 }
```

执行完如下效果:

```
1 <?xml version='1.0' encoding='UTF-8'?>
2 <Library>
3     <Book1 Name="水浒传" ID="1" Price="64.5">
4         <Description>108个拆迁户</Description>
5         <Page>100页</Page>
6     </Book1>
7     <Book1 Name="三国演义" ID="2" Price="88.5">
8         <Description>魏蜀吴在玩过家家</Description>
9         <Page>300页</Page>
10    </Book1>
11 </Library>
```

https://blog.csdn.net/cpp_learner

六、读取

读取指定节点中值

例:

/* 读取Book1节点, ID为1的节点的所有数据 */

1. 首先打开文件，关联对象

记得别关闭文件哦！

```
1 // 打开文件
2 QFile file("QT_XML.xml");
3 if (!file.open(QIODevice::ReadOnly)) {
4     QMessageBox::information(NULL, "提示", "文件打开失败！");
5     return;
6 }
7
8 QDomDocument doc;
9 if (!doc.setContent(&file)) {
10    QMessageBox::information(NULL, "提示", "操作的文件不是XML文件！");
11    file.close();
12    return;
13 }
```

2. 获取属性中的值

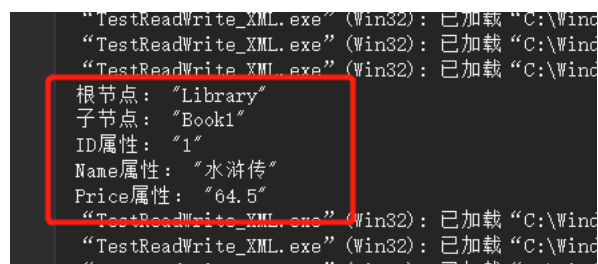
```
1 /* 获取属性中的值 */
2 for (int i = 0; i < list.count(); i++) {
3     // 获取链表中的值
4     QDomElement element = list.at(i).toElement();
5
6     // 找到需要读取的节点
7     if (element.attribute("Name") == "水浒传") {
8         // 通过attribute方法返回属性中的值
9         qDebug() << "子节点: " << element.nodeName();
10        qDebug() << "ID属性: " << element.attribute("ID");
11        qDebug() << "Name属性: " << element.attribute("Name");
12        qDebug() << "Price属性: " << element.attribute("Price");
13    }
14 }
```

思路:

定位到当前节点后，使用attribute函数可以返回当前属性中的值

使用nodeName方法可以获取节点名字

如下效果:



3. 获取尖括号间的值

```
1 for (int i = 0; i < list.count(); i++) {
2     // 获取链表中的值
3     QDomElement element = list.at(i).toElement();
4
5     // 找到需要读取的节点
6     if (element.attribute("Name") == "水浒传") {
7         // 获得子节点
8         QDomNode description = element.namedItem("Description");
9         QDomNode page = element.namedItem("Page");
10        qDebug() << "Description: " << description.firstChild().nodeValue();
11        qDebug() << "Page: " << page.firstChild().nodeValue();
12
13        // 也可以使用childNodes方法获取所有的子节点
14        /*QDomNodeList nodeList = element.childNodes();
```



```
17     }  
18 }
```

思路:

获得尖括号节点对象后, 使用.firstChild().nodeValue();就可以获得尖括号间的值了

使用childNodes方法可以获取所有子节点, 返回值是QDomNodeList。

运行如下图:

```
“TestReadWrite_XML.exe” (Win32): 已加载 “C:\Windows\Sys  
根节点: “Library”  
子节点: “Book1”  
ID属性: “1”  
Name属性: “水浒传”  
Price属性: “64.5”  
Description: “108个拆迁户”  
Page: “100页”  
“TestReadWrite_XML.exe” (Win32): 已加载 “C:\Windows\Sys  
“TestReadWrite_XML.exe” (Win32): 已加载 “C:\Windows\Sys
```

4. 最后记得关闭文件哦

```
file.close();
```

5. 读取xml节点值, 代码汇总

```
1 // 读取xml中的内容  
2 void readXML() {  
3  
4     /* 读取Book1节点, ID为1的节点的所有数据 */  
5  
6     // 打开文件  
7     QFile file("QT_XML.xml");  
8     if (!file.open(QIODevice::ReadOnly)) {  
9         QMessageBox::information(NULL, "提示", "文件打开失败!");  
10        return;  
11    }  
12  
13    QDomDocument doc;  
14    if (!doc.setContent(&file)) {  
15        QMessageBox::information(NULL, "提示", "操作的文件不是XML文件!");  
16        file.close();  
17        return;  
18    }  
19  
20  
21  
22    // 获得根节点  
23    QDomElement root = doc.documentElement();  
24    qDebug() << "根节点: " << root.nodeName();  
25  
26    // 获取所有Book1节点  
27    QDomNodeList list = root.elementsByTagName("Book1");  
28  
29  
30    /* 获取属性中的值 */  
31    for (int i = 0; i < list.count(); i++) {  
32        // 获取链表中的值  
33        QDomElement element = list.at(i).toElement();  
34  
35        // 找到需要读取的节点  
36        if (element.attribute("Name") == "水浒传") {  
37            // 通过attribute方法返回属性中的值  
38            qDebug() << "子节点: " << element.nodeName();  
39            qDebug() << "ID属性: " << element.attribute("ID");  
40            qDebug() << "Name属性: " << element.attribute("Name");  
41            qDebug() << "Price属性: " << element.attribute("Price");  
42        }  
43    }  
44 }
```

```
47     for (int i = 0; i < list.count(); i++) {
48         // 获取链表中的值
49         QDomElement element = list.at(i).toElement();
50
51         // 找到需要读取的节点
52         if (element.attribute("Name") == "水浒传") {
53             // 获得子节点
54             QDomNode description = element.namedItem("Description");
55             QDomNode page = element.namedItem("Page");
56             qDebug() << "Description: " << description.firstChild().nodeValue();
57             qDebug() << "Page: " << page.firstChild().nodeValue();
58
59             // 也可以使用childNodes方法获取所有的子节点
60             /*QDomNodeList nodeList = element.childNodes();
61             qDebug() << nodeList.at(0).firstChild().nodeValue();
62             qDebug() << nodeList.at(1).firstChild().nodeValue();*/
63         }
64     }
65
66     file.close();
67 }
68 }
```

七、总结

到了这里，那么恭喜你已经完全学会QT如何操作XML文件了，可以去写项目去了。

```
1 <?xml version='1.0' encoding='UTF-8'?>
2 <Library>
3   <Books>
4     <Book ID="1" Name="水浒传" Price="64.5" Description="108个拆迁户" Page="100页" />
5     <Book ID="2" Name="三国演义" Price="88.5" Description="魏蜀吴在玩过家家" Page="300页" />
6     <Book ID="3" Name="西游记" Price="68.5" Description="两个人，一匹马，两个动物" Page="81页" />
7   </Books>
8 </Library>
```

我所接触过的项目，大多都是这种写法；当然也有在Book中再添加子节点的。

https://blog.csdn.net/cpp_learner

显示推荐内容

觉得还不错? 一键收



cpp_learners

关注

50

219