简介

- XML被设计用来结构化、存储和传输数据。
- XML是纯文本
- XML可自定义标签
- XML是树形结构

语法

规则列表

- 所有 XML 元素都须有**结束标签**
- XML标签对**大小**写敏感
- XML必须正确地**嵌套**
- XML文档必须要有**根元素**
- XML 的属性值须加**双引号**
- XML中空格被保留
- XML 以 LF 存储换行
- XML的转义字符

<	**<**	**小于**
>	>	大于
&	&	和号
'	'	单引号
"	"	引号

- XML注释

元素

XML 元素指的是从（且包括）开始标签直到（且包括）结束标签的部分。

XML 元素必须遵循以下命名规则：

- 名称可以含字母、数字以及其他的字符
- 名称不能以数字或者标点符号开始
- 名称不能以字符 “xml”（或者 XML、Xml）开始
- 名称不能包含空格

避免使用XML属性而引起的一些问题：

- 属性无法包含多重的值（元素可以）
- 属性无法描述树结构（元素可以）
- 属性不易扩展（为未来的变化）
- 属性难以阅读和维护

下面的三个 XML 文档包含完全相同的信息：

第一个例子中使用了 date 属性：

```
<note date="08/08/2008">
<to>George</to>
<from>John</from>
<heading>Reminder</heading>
<body>Don't forget the meeting!</body>
</note>
```

第二个例子中使用了 date 元素：

```
<note>
<date>08/08/2008</date>
<to>George</to>
<from>John</from>
<heading>Reminder</heading>
<body>Don't forget the meeting!</body>
</note>
```

第三个例子中使用了扩展的 date 元素（这是我的最爱）：

```
<note>
<date>
  <day>08</day>
  <month>08</month>
  <year>2008</year>
</date>
<to>George</to>
<from>John</from>
<heading>Reminder</heading>
<body>Don't forget the meeting!</body>
</note>
```

命名空间

> XML 命名空间提供避免元素命名冲突的方法

默认的命名空间（Default Namespaces）

为元素定义默认的命名空间可以让我们省去在所有的子元素中使用前缀的工作。

请使用下面的语法：

```
xmlns="namespaceURI"
```

这个 XML 文档携带着某个表格中的信息：

```
<table xmlns="http://www.w3.org/TR/html4/">
   <tr>
   <td>Apples</td>
   <td>Bananas</td>
   </tr>
</table>
```

此 XML 文档携带着有关一件家具的信息：

```
<table xmlns="http://www.w3school.com.cn/furniture">
   <name>African Coffee Table</name>
   <width>80</width>
   <length>120</length>
</table>
```

XML DOM (Document Object Model)

DOM 把 XML 文档视为一种树结构。通过这个 DOM 树，可以访问所有的元素。可以修改它们的内容（文本以及属性），而且可以创建新的元素。元素，以及它们的文本和属性，均被视为节点。

XML 文档中的每个成分都是一个节点，DOM 把XML文档视为一棵节点树 (node-tree)，规定如下：

- 整个文档是一个文档节点(根元素节点)
- 每个 XML 标签是一个元素节点
- 包含在 XML 元素中的文本是文本节点
- 每一个 XML 属性是一个属性节点
- 注释属于注释节点

节点操作：

序号	操作	
1	获取节点	获取元素节点 获取属性节点
2	修改节点	setNodeValue setAttribute
3	删除节点	删除元素节点： removeChild 删除自身 - 删除当前的节点parentNode、removeChild 删除文本节点 removeChild 清空文本节点 nodeValue 根据名称删除属性节点removeAttribute 根据对象删除属性节点removeAttributeNode
4	替换节点	替换元素节点 replaceChild 替换文本节点中的数据 replaceData
5	创建节点	创建新的元素节点 : createElement 通过使用 setAttribute 来创建属性 创建新的属性节点 : createAttribute 创建文本节点 : createTextNode 创建注释节点 : createComment
6	添加节点	添加子节点：appendChild 添加父节点：insertBefore insertAfter
7	克隆节点	cloneNode

QT XML

> QDomProcessingInstruction

> QDomDocument
>
> QDomNode
>
> QDomNodeList
>
> QDomElement
>
> QDomAttr
>
> QDomText
>
> QDomComment

```
#include "mainwindow.h"

#include <QApplication>
#include <QtXml>
#include <QFile>


void createXmlHead(QString filename,QString workshop)
{
    //xml DOM对象
    QDomDocument doc;

    //xml文件头
    QDomProcessingInstruction xmlHead =
doc.createProcessingInstruction("xml","version=\"1.0\" encoding=\"UTF-
8\"");
    doc.appendChild(xmlHead);

    //创建根元素节点
    QDomElement root = doc.createElement(workshop);
    doc.appendChild(root);

    //xml 文件保存
    QFile file(filename);
    if(!file.open(QFile::WriteOnly|QFile::Truncate))
        return ;
    QTextStream outStream(&file);
    doc.save(outStream,4);
    file.close();
}

void addXml(QString filename,QString cs,QString CameraNum,QString
id,QString place,QString clientIP,QString serverIP)
{
    //打开或创建文件
    QFile file(filename); //相对路径、绝对路径、资源路径都行
    if(!file.open(QFile::ReadOnly))
    {
        qDebug() << "open failed";
        return;
    }

    QString err;
    int line,col;
    QDomDocument doc;
```

```
    if(!doc.setContent(&file,true,&err,&line,&col))
    {
        qDebug() << "setContent failed : " << err << "line:" <<line
<<"col:"<<col;
        file.close();
        return;
    }
    file.close();

    QDomNode root = doc.documentElement();

    QDomElement csElement = doc.createElement(cs);
    root.appendChild(csElement);

    csElement.setAttribute("id",id);

    QDomAttr attr = doc.createAttribute("place");
    attr.setNodeValue(place);
    csElement.setAttributeNode(attr);

    attr = doc.createAttribute("CameraNum");
    attr.setNodeValue(CameraNum);
    csElement.setAttributeNode(attr);

    QDomElement clientElement = doc.createElement("client");
    csElement.appendChild(clientElement);

    QDomElement ipElement = doc.createElement("ip");
    clientElement.appendChild(ipElement);

    QDomText text = doc.createTextNode(clientIP);
    ipElement.appendChild(text);

    QDomElement serverElement = doc.createElement("server");
    csElement.appendChild(serverElement);

    ipElement = doc.createElement("ip");
    serverElement.appendChild(ipElement);

    text = doc.createTextNode(serverIP);
    ipElement.appendChild(text);

    QDomComment comment = doc.createComment(QString::fromLocal8Bit("图像识
别客户端"));
    csElement.insertBefore(comment,clientElement);

    comment = doc.createComment(QString::fromLocal8Bit("图像识别服务端"));
    csElement.insertBefore(comment,serverElement);

    //xml 文件保存
    if(!file.open(QFile::WriteOnly|QFile::Truncate))
        return ;
    QTextStream outStream(&file);
    doc.save(outStream,4);
    file.close();
}

void readXml(QString filename)
```

```
{
    QFile file(filename);
    if(!file.open(QFile::ReadOnly))
    {
        qDebug() << "open failed";
        return;
    }

    QString err;
    int line,col;
    QDomDocument doc;
    if(!doc.setContent(&file,true,&err,&line,&col))
    {
       qDebug() << "setContent failed : " << err << "line:" <<line <<"col:"<<col;
        file.close();
        return;
    }
    file.close();

    //返回根节点
    QDomElement root=doc.documentElement();
    qDebug()<<root.nodeName();

    QDomNode node=root.firstChild(); //获得第一个子节点
    while(!node.isNull())  //如果节点不空
    {
        if(node.isElement()) //如果节点是元素
        {
            QDomElement e=node.toElement(); //转换为元素，注意元素和节点是两个数据结构，其实差不多
            qDebug()<<"    "<<e.tagName()<<" "<<e.attribute("id")<<" "<<e.attribute("place")<<e.attribute("CameraNum"); //打印键值对，tagName和nodeName是一个东西
            QDomNodeList list=e.childNodes();
            for(int i=0;i<list.count();i++) //遍历子元素，count和size都可以用，可用于标签数计数
            {
                QDomNode n=list.at(i);
                if(node.isElement())
                    qDebug()<<"        "<<n.nodeName()<<":"<<n.toElement().text();
            }
        }
        node=node.nextSibling(); //下一个兄弟节点,nextSiblingElement()是下一个兄弟元素，都差不多
    }


}

void removeXml(QString filename,QString id)
{
    //打开或创建文件
    QFile file(filename); //相对路径、绝对路径、资源路径都行
    if(!file.open(QFile::ReadOnly))
    {
        qDebug() << "open failed";
```

```
        return;
    }

    QString err;
    int line,col;
    QDomDocument doc;
    if(!doc.setContent(&file,true,&err,&line,&col))
    {
        qDebug() << "setContent failed : " << err << "line:" <<line
<<"col:"<<col;
        file.close();
        return;
    }
    file.close();

    QDomNodeList list=doc.elementsByTagName(QString::fromLocal8Bit("宽厚板
车间")); //由标签名定位
    for(int i=0;i<list.count();i++)
    {
        QDomNodeList listChild =list.at(i).childNodes();
        for(int j=0;j<listChild.count();j++)
        {
            QDomElement e = listChild.at(j).toElement();
            qDebug()<<e.tagName()<<e.attribute("id");
            if(e.attribute("id") == id)
                e.parentNode().removeChild(e);
        }
    }

    // 保存文件
    if(!file.open(QFile::WriteOnly|QFile::Truncate))
        return;
    QTextStream out_stream(&file);
    doc.save(out_stream,4); //缩进4格
    file.close();
}

void updateXml(QString filename)
{
    //更新一个标签项,如果知道xml的结构，直接定位到那个标签上定点更新
    //或者用遍历的方法去匹配tagname或者attribut，value来更新

    //打开或创建文件
    QFile file(filename); //相对路径、绝对路径、资源路径都行
    if(!file.open(QFile::ReadOnly))
    {
        qDebug() << "open failed";
        return;
    }

    QString err;
    int line,col;
    QDomDocument doc;
    if(!doc.setContent(&file,true,&err,&line,&col))
    {
        qDebug() << "setContent failed : " << err << "line:" <<line
<<"col:"<<col;
        file.close();
```

```
        return;
    }
    file.close();

    QDomElement root=doc.documentElement();

    QDomNodeList list=root.elementsByTagName(QString::fromLocal8Bit("宽厚板车间"));

    QDomNode node=list.at(list.size()-1).firstChild(); //定位到第三个一级子节点的子元素

    QDomNode oldnode=node.firstChild(); //标签之间的内容作为节点的子节点出现,当前是Pride and Projudice
    node.firstChild().setNodeValue("Emma");

    QDomNode newnode=node.firstChild();

    node.replaceChild(newnode,oldnode);

    if(!file.open(QFile::WriteOnly|QFile::Truncate))
        return;
    //输出到文件
    QTextStream out_stream(&file);
    doc.save(out_stream,4); //缩进4格
    file.close();
}

int main(int argc, char *argv[])
{
    createXmlHead(QString::fromLocal8Bit("远程监控.xml"),QString::fromLocal8Bit("宽厚板车间"));

    addXml(QString::fromLocal8Bit("远程监控.xml"),"CS_16","5","1","KHB2#","10.47.0.209","192.168.2.14");
    addXml(QString::fromLocal8Bit("远程监控.xml"),"CS_17","5","2","KHB2#","10.47.0.208","192.168.2.14");
    addXml(QString::fromLocal8Bit("远程监控.xml"),"CS_13","14","3","KHB1#","10.30.0.214","10.30.0.216");
    addXml(QString::fromLocal8Bit("远程监控.xml"),"CS_3500","7","5","KHB1#","10.6.7.101","10.6.7.102");
    addXml(QString::fromLocal8Bit("远程监控.xml"),"CS_NEW3500","7","6","KHB1#","10.6.4.25","10.6.7.102");

    readXml(QString::fromLocal8Bit("远程监控.xml"));

    removeXml(QString::fromLocal8Bit("远程监控.xml"),"5");

    readXml(QString::fromLocal8Bit("远程监控.xml"));

    return 0;
}
```