二十九、Qt数据库（九）XML（二）

在上一节中我们用手写的方法建立了一个XML文档，并且用DOM的方法对其进行了读取。现在我们使用代码来创建那个XML文档，并且对它实现查找，更新，插入等操作。

首先，我们新建Qt4 Gui Application工程，工程名为xml02，然后添加QtXml模块，我们选择QWidget为Base class 。

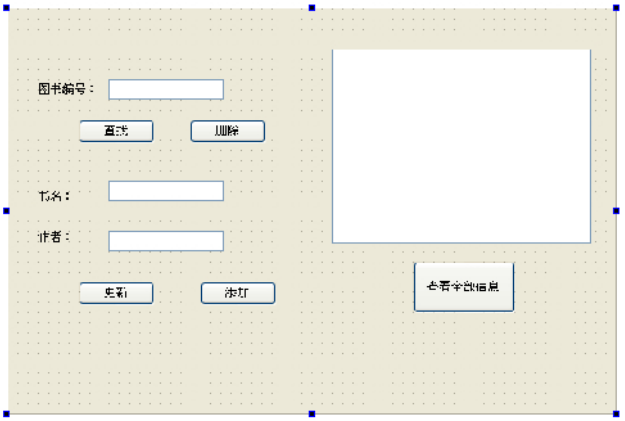
1.为了在程序中可以使用中文，我们先在main.cpp文件中添加头文件：

#include <QTextCodec>

然后在main()函数中添加一行代码：

QTextCodec::setCodecForTr(QTextCodec::codecForLocale());

2.然后到widget.ui文件中，将界面设计如下：



其中用到的部件有Push Button，ListWidget，Label和Line Edit 。

3.我们再到widget.cpp文件中，添加头文件：#include <QtXml>

然后在构造函数中添加代码如下：

QFile file(“my.xml”);

if(!file.open(QIODevice::WriteOnly | QIODevice::Truncate))
return ;

//只写方式打开，并清空以前的信息

QDomDocument doc;

QDomProcessingInstruction instruction; //添加处理指令

instruction = doc.createProcessingInstruction(“xml”,

“version=\”1.0\” encoding=\”UTF-8\”");

doc.appendChild(instruction);

QDomElement root = doc.createElement(tr(“书库”));

doc.appendChild(root); //添加根元素

//添加第一个book元素及其子元素

QDomElement book = doc.createElement(tr(“图书”));

QDomAttr id = doc.createAttribute(tr(“编号”));

QDomElement title = doc.createElement(tr(“书名”));

QDomElement author = doc.createElement(tr(“作者”));

QDomText text;

```
id.setValue(tr("1"));
book.setAttributeNode(id);
text = doc.createTextNode(tr("Qt"));
title.appendChild(text);
text = doc.createTextNode(tr("shiming"));
author.appendChild(text);
book.appendChild(title);
book.appendChild(author);
root.appendChild(book);
//添加第二个book元素及其子元素
book = doc.createElement(tr("图书"));
id = doc.createAttribute(tr("编号"));
title = doc.createElement(tr("书名"));
author = doc.createElement(tr("作者"));
id.setValue(tr("2"));
book.setAttributeNode(id);
text = doc.createTextNode(tr("Linux"));
title.appendChild(text);
text = doc.createTextNode(tr("yafei"));
author.appendChild(text);
book.appendChild(title);
book.appendChild(author);
root.appendChild(book);
QTextStream out(&file);
doc.save(out,4);  //将文档保存到文件，4为子元素缩进字符数
file.close();
```

这样我们就建立起了一个XML文档，过程并不复杂，只要注意一下元素的父子关系就可以了。最后我们用save()函数将数据从doc中保存到文件中。

4. 下面我们读取这个**XML** 文件，我们从**widget.ui**中单击“查看全部信息”按钮，进入它的单击事件响应函数，并更改如下：

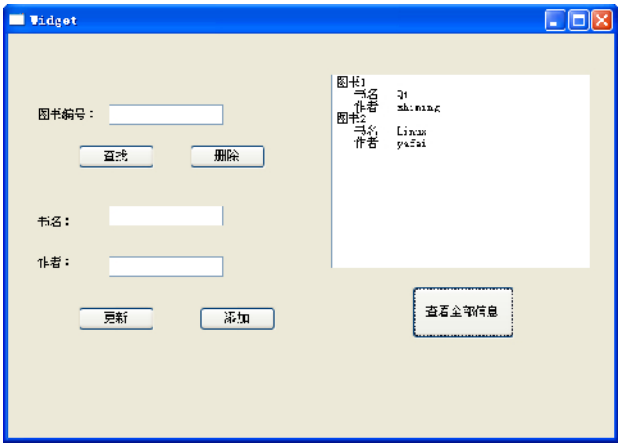
```
void Widget::on_pushButton_clicked()  //显示全部
```

```
{
ui->listWidget->clear(); //先清空显示
QFile file(“my.xml”);
if (!file.open(QIODevice::ReadOnly)) return ;
QDomDocument doc;
if (!doc.setContent(&file))
{
file.close();
return ;
}
file.close();
//返回根节点及其子节点的元素标记名
QDomElement docElem = doc.documentElement();  //返回根元素
QDomNode n = docElem.firstChild();   //返回根节点的第一个子节点
while(!n.isNull())  //如果节点不为空
{
if (n.isElement()) //如果节点是元素
{
QDomElement e = n.toElement(); //将其转换为元素
ui->listWidget->addItem(e.tagName()+e.attribute(tr(“编号”)));
QDomNodeList list = e.childNodes();
for(int i=0; i<list.count(); i++)
{
QDomNode node = list.at(i);
if(node.isElement())
ui->listWidget->addItem(“  “+node.toElement().tagName()
+” : “+node.toElement().text());
}
}
n = n.nextSibling();  //下一个兄弟节点
```

}

}

这里的代码就是上一节我们讲的读取XML文档所用的代码，只是将以前的qDebug()输出换成了在listWidget上进行输出。运行程序，单击按键，效果如下：



5. 下面我们加入添加功能，进入“添加”按钮的单击事件函数，更改如下：

void Widget::on_pushButton_5_clicked() //添加

{

ui->listWidget->clear(); //我们先清空显示，然后显示“无法添加！”

ui->listWidget->addItem(tr(“无法添加！”));

QFile file(“my.xml”);

if (!file.open(QIODevice::ReadOnly)) return;

QDomDocument doc;

if (!doc.setContent(&file))

{

file.close();

return;

}

file.close();

QDomElement root = doc.documentElement();

QDomElement book = doc.createElement(tr(“图书”));

QDomAttr id = doc.createAttribute(tr(“编号”));

QDomElement title = doc.createElement(tr(“书名”));

QDomElement author = doc.createElement(tr(“作者”));

QDomText text;

QString num = root.lastChild().toElement().attribute(tr(“编号”));

int count = num.toInt() +1;

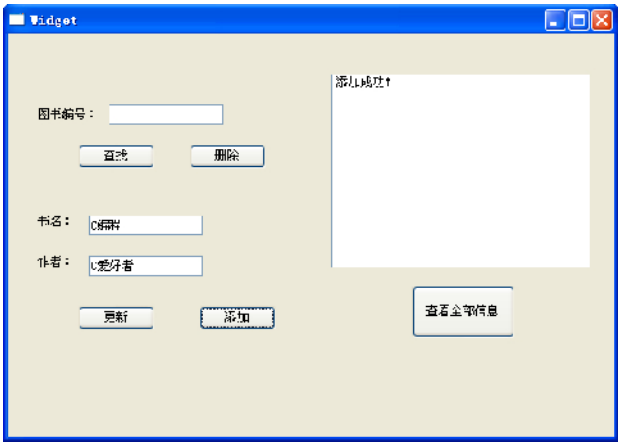
id.setValue(QString::number(count));

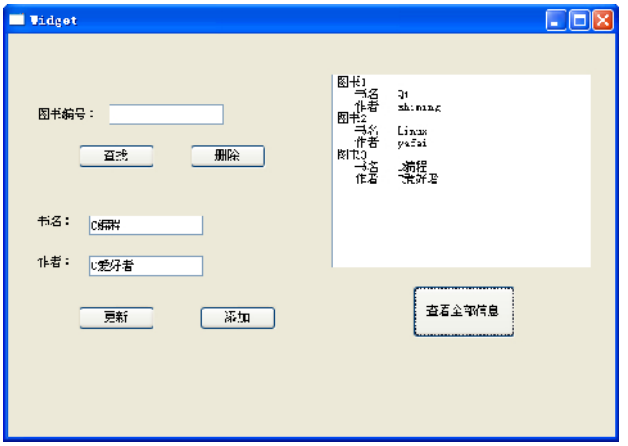
//我们获得了最后一个孩子结点的编号，然后加1，便是新的编号

book.setAttributeNode(id);

```
text = doc.createTextNode(ui->lineEdit_2->text());

//注意：你那可能不是lineEdit_2。

title.appendChild(text);

text = doc.createTextNode(ui->lineEdit_3->text());

author.appendChild(text);

book.appendChild(title);

book.appendChild(author);

root.appendChild(book);

if(!file.open(QIODevice::WriteOnly | QIODevice::Truncate)) return ;

QTextStream out(&file);

doc.save(out,4);  //将文档保存到文件，4为子元素缩进字符数

file.close();

ui->listWidget->clear(); //最后更改显示为“添加成功！”

ui->listWidget->addItem(tr(“添加成功！”));

}
```

这里先用只读方式打开XML文件，将其读入doc中，然后关闭。我们将新的节点加入到最后面，并使其“编号”为以前的最后一个节点的编号加1。最后我们再用只写的方式打开XML文件，将修改完的doc写入其中。运行程序，效果如下：





我们先添加一个节点，然后再查看全部信息，发现确实已经添加成功了。

6. 查找、删除、更新操作

因为这三个功能都要先利用“编号”进行查找，所以我们放在一起实现。

我们先在widget.h文件中添加一个函数声明：

```
void doXml(const QString operate);
```

它有一个参数，用来传递所要进行的操作。

然后到widget.cpp文件中对它进行定义：

```
void Widget::doXml(const QString operate)
{
ui->listWidget->clear();
ui->listWidget->addItem(tr(“没有找到相关内容！”));
QFile file(“my.xml”);
if (!file.open(QIODevice::ReadOnly)) return ;
QDomDocument doc;
if (!doc.setContent(&file))
{
file.close();
return ;
}
file.close();
QDomNodeList list = doc.elementsByTagName(tr(“图书”));
//以标签名进行查找
for(int i=0; i<list.count(); i++)
{
QDomElement e = list.at(i).toElement();
if(e.attribute(tr(“编号”)) == ui->lineEdit->text())
{ //如果元素的“编号”属性值与我们所查的相同
if(operate == “delete”) //如果是删除操作
{
QDomElement root = doc.documentElement(); //取出根节点
root.removeChild(list.at(i)); //从根节点上删除该节点
QFile file(“my.xml”);    //保存更改
if(!file.open(QIODevice::WriteOnly | QIODevice::Truncate)) return ;
QTextStream out(&file);
doc.save(out,4);
file.close();
```

```
ui->listWidget->clear();
ui->listWidget->addItem(tr(“删除成功！”));
}
else if(operate == “update”) //如果是更新操作
{
QDomNodeList child = list.at(i).childNodes();
//找到它的所有子节点，就是“书名”和“作者”
child.at(0).toElement().firstChild().setNodeValue(ui->lineEdit_2->text());
//将它子节点的首个子节点（就是文本节点）的内容更新
child.at(1).toElement().firstChild().setNodeValue(ui->lineEdit_3->text());
QFile file(“my.xml”);  //保存更改
if(!file.open(QIODevice::WriteOnly | QIODevice::Truncate)) return ;
QTextStream out(&file);
doc.save(out,4); //将文档保存到文件，4为子元素缩进字符数
file.close();
ui->listWidget->clear();
ui->listWidget->addItem(tr(“更新成功！”));
}
else if(operate == “find”) //如果是查找操作
{
ui->listWidget->clear();
ui->listWidget->addItem(e.tagName()+e.attribute(tr(“编号”)));
QDomNodeList list = e.childNodes();
for(int i=0; i<list.count(); i++)
{
QDomNode node = list.at(i);
if(node.isElement())
ui->listWidget->addItem(“  “+node.toElement().tagName()
+” : “+node.toElement().text());
}
```

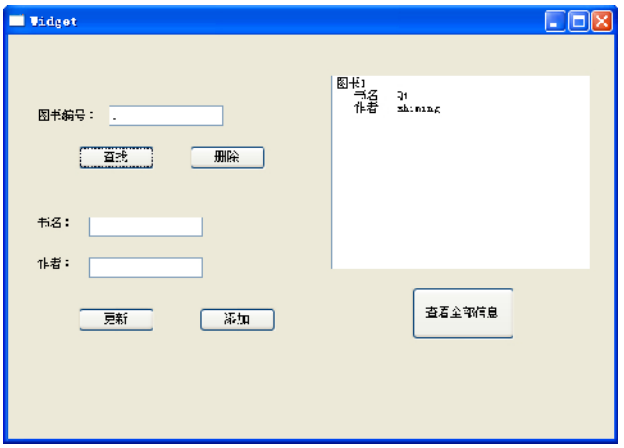
```
}
}
}
}
```

然后我们分别进入“查找”，“删除”，“更新”三个按键的单击事件槽函数，更改如下：

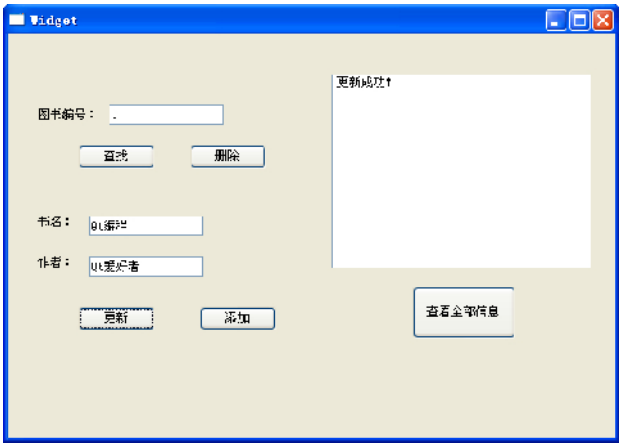
```
void Widget::on_pushButton_2_clicked()  //查找
{
doXml(“find”);
}
void Widget::on_pushButton_3_clicked()  //删除
{
doXml(“delete”);
}
void Widget::on_pushButton_4_clicked()  //更新
{
doXml(“update”);
}
```

这时，我们运行程序。

在“图书编号”中输入1，然后点击“查找”按键，效果如下：



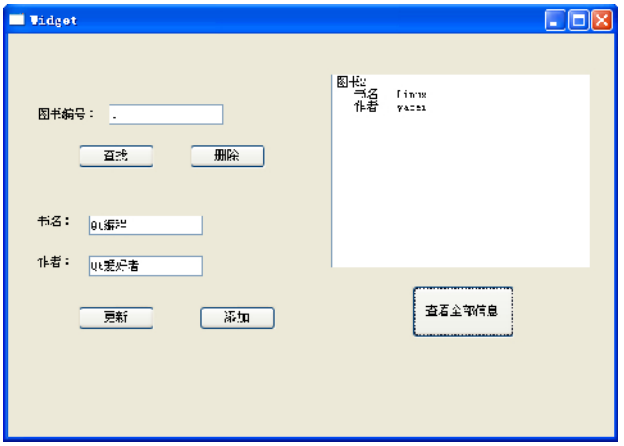
这时我们将下面的“书名”和“作者”进行更改，然后单击“更新”按键，效果如下：



再次点击“查找”按键：

然后点击“删除”按键：

再点击“查看全部信息”按键：



我们发现，所有的操作的结果都是正确的。

我们的**Dom**部分就讲到这里，下一节我们讲述用**SAX**的方法读取**XML**文档

分类：Qt系列教程 作者： yafeilinux 日期： 五月 5th, 2010.
239 views
Tags: creator, dom, qt, xml, yafeilinux, 教程, 数据库