Министерство связи и массовых коммуникаций Российской Федерации (Минкомсвязь РФ)  
Федеральное государственное образовательное бюджетное учреждение высшего профессионального образования  
"Сибирский государственный университет телекоммуникаций и информатики" (ФГОБУ ВПО "СибГУТИ")

*Кафедра ТС и ВС*

Лабораторная работа № 7

Создание редактора XML-файлов в Qt

**Выполнили:** студенты группы *ИП-513*

*Майоров С.А.*

*Санин И.В.*

**Проверил:** старший преподаватель кафедры ТС и ВС

Лапова С.Г.

Новосибирск, 2017

**1. Цель работы**

Создание, чтение, запись XML-файла в среде QtCreator, иерархические списки.

**2. Листинг**

**mainwindow.h:**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QFile>

#include <QTemporaryFile>

#include <QFileDialog>

#include <QTextStream>

#include <QDebug>

#include <QTimer>

#include <QMessageBox>

#include "domparser.h"

#include "saxparser.h"

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~*MainWindow*();

private slots:

void on\_save\_clicked();

void on\_save\_as\_clicked();

void temporaryFileSave();

void on\_open\_clicked();

void on\_textEdit\_textChanged();

void on\_clear\_clicked();

void on\_exit\_clicked();

void on\_update\_clicked();

private:

Ui::MainWindow \*ui;

QTimer \*time;

QString fileName;

QString tmpFileName;

};

#endif // MAINWINDOW\_H

**mainwindow.cpp:**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

ui->progressBar->setMinimum(0);

ui->progressBar->setMaximum(100);

ui->progressBar->hide();

ui->progressBar\_2->hide();

ui->Label2->hide();

time = new QTimer(this);

time->setInterval(1000 \* 60 \* 3);

time->start();

connect(time, SIGNAL(timeout()), this, SLOT(temporaryFileSave()));

fileName = "";

tmpFileName = "";

}

MainWindow::~*MainWindow*()

{

if (!tmpFileName.isEmpty())

{

QFile tmpFile(tmpFileName);

tmpFile.remove();

}

delete ui;

}

void MainWindow::on\_save\_clicked()

{

QFile file(fileName);

if(file.*open*(QIODevice::WriteOnly | QIODevice::Text))

{

QTextStream textStream(&file);

textStream << ui->textEdit->toPlainText();

file.*close*();

temporaryFileSave();

ui->update->setEnabled(true);

} else {

QString lFileName = QFileDialog::getSaveFileName(0,

QObject::tr("Save File..."),

QDir::currentPath(),

QObject::tr("XML Files(\*.xml);;All Files (\*.\*)"));

if(!lFileName.isEmpty())

{

file.setFileName(lFileName);

file.*open*(QIODevice::WriteOnly | QIODevice::Text);

QTextStream textStream(&file);

textStream << ui->textEdit->toPlainText();

file.*close*();

fileName = lFileName;

ui->Label1->setText(fileName);

if (!tmpFileName.isEmpty())

{

QFile tmpFile(tmpFileName);

tmpFile.remove();

}

temporaryFileSave();

ui->update->setEnabled(true);

}

}

}

void MainWindow::on\_save\_as\_clicked()

{

QString lFileName = QFileDialog::getSaveFileName(0,

QObject::tr("Save File..."),

QDir::currentPath(),

QObject::tr("XML Files(\*.xml);;All Files (\*.\*)"));

if(!lFileName.isEmpty())

{

QFile file(lFileName);

file.*open*(QIODevice::WriteOnly | QIODevice::Text);

QTextStream textStream(&file);

textStream << ui->textEdit->toPlainText();

file.*close*();

fileName = lFileName;

ui->Label1->setText(fileName);

if (!tmpFileName.isEmpty())

{

QFile tmpFile(tmpFileName);

tmpFile.remove();

}

temporaryFileSave();

ui->update->setEnabled(true);

}

}

void MainWindow::temporaryFileSave()

{

qDebug() << "Save";

ui->progressBar->show();

ui->Label2->show();

int speed\_index = 5000;

for (int i = 0; i < 100 \* speed\_index; ++i)

{

ui->progressBar->setValue(i/speed\_index);

ui->Label2->setText("Cохранение...");

}

QFile\* tmpFile;

if (fileName.isEmpty())

tmpFileName = QDir::tempPath() + "/" + qApp->applicationName() + "~";

else

tmpFileName = fileName + "~";

tmpFile = new QFile(tmpFileName);

tmpFile->*open*(QIODevice::WriteOnly | QIODevice::Text);

QTextStream textStream(tmpFile);

textStream << ui->textEdit->toPlainText();

tmpFile->*close*();

ui->progressBar->hide();

ui->Label2->hide();

time->start();

}

void MainWindow::on\_open\_clicked()

{

QString lFileName = QFileDialog::getOpenFileName(0,

QObject::tr("Open File..."),

QDir::currentPath(),

QObject::tr("XML Files(\*.xml);;All Files (\*.\*)"));

if(!lFileName.isEmpty())

{

QFile file(lFileName);

file.*open*(QIODevice::ReadOnly | QIODevice::Text);

QTextStream out(&file);

ui->textEdit->setText(out.readAll());

file.*close*();

fileName = lFileName;

ui->Label1->setText(fileName);

if (!tmpFileName.isEmpty())

{

QFile tmpFile(tmpFileName);

tmpFile.remove();

}

temporaryFileSave();

ui->update->setEnabled(true);

}

ui->update->setEnabled(true);

}

void MainWindow::on\_textEdit\_textChanged()

{

ui->update->setEnabled(false);

}

void MainWindow::on\_clear\_clicked()

{

ui->textEdit->clear();

}

void MainWindow::on\_exit\_clicked()

{

QMessageBox question(QMessageBox::Question, "Save file?", "Do you want to save file before exit?",

QMessageBox::Ok, this);

question.addButton(QMessageBox::No);

question.addButton(QMessageBox::Cancel);

question.*exec*();

if (question.clickedButton() == question.button(QMessageBox::Ok)){

on\_save\_clicked();

} else {

if (question.clickedButton() == question.button(QMessageBox::Cancel)){

return;

}

}

this->close();

}

void MainWindow::on\_update\_clicked()

{

ui->treeWidget->clear();

QTreeWidget\* treeWidget = ui->treeWidget;

QProgressBar\* bar = ui->progressBar\_2;

QFile file(fileName);

if (ui->dom->isChecked() && !fileName.isEmpty())

DomParser(&file, treeWidget, bar);

if (ui->sax->isChecked() && !fileName.isEmpty())

SaxParser(&file, treeWidget, bar);

treeWidget->resizeColumnToContents(0);

treeWidget->resizeColumnToContents(1);

}

**domparser.cpp:**

#include "domparser.h"

DomParser::DomParser(QIODevice \*device, QTreeWidget \*tree, QProgressBar \*bar) :

treeWidget(tree),

progress(bar)

{

QString errorStr;

int errorLine;

int errorColumn;

QDomDocument doc;

if (!doc.setContent(device, true, &errorStr, &errorLine, &errorColumn)) {

QMessageBox::warning(0,

QObject::tr("DOM Parser"),

QObject::tr("Parse error at line %1, column %2:\n%3")

.arg(errorLine)

.arg(errorColumn)

.arg(errorStr)

);

return;

}

nodeCount = 1;

nodeCount += doc.elementsByTagName("value").count();

nodeCount += doc.elementsByTagName("attr").count();

nodeCount += doc.elementsByTagName("food").count();

progress->setMinimum(1);

progress->setMaximum(nodeCount);

progress->setValue(1);

progress->show();

QDomElement root = doc.documentElement();

if (root.tagName() != "breakfast\_menu")

return;

QDomNode node = root.firstChild();

while (!node.isNull()) {

if (node.toElement().tagName() == "food")

parseEntry(node.toElement(), 0);

progress->setValue(progress->value() + 1);

node = node.nextSibling();

}

}

void DomParser::parseEntry(const QDomElement &element, QTreeWidgetItem \*parent)

{

QTreeWidgetItem \*item;

if (parent) {

item = new QTreeWidgetItem(parent);

} else {

item = new QTreeWidgetItem(treeWidget);

}

item->setText(0, element.attribute("name"));

QDomNode node = element.firstChild();

while (!node.isNull()) {

if (node.toElement().tagName() == "food" || node.toElement().tagName() == "attr") {

parseEntry(node.toElement(), item);

} else if (node.toElement().tagName() == "value") {

QDomNode childNode = node.firstChild();

while (!childNode.isNull()) {

if (childNode.nodeType() == QDomNode::TextNode) {

QString page = childNode.toText().data();

QString allPages = item->text(1);

if (!allPages.isEmpty())

allPages += ", ";

allPages += page;

item->setText(1, allPages);

break;

}

childNode = childNode.nextSibling();

}

}

progress->setValue(progress->value() + 1);

node = node.nextSibling();

}

}

**saxparser.cpp:**

#include "saxparser.h"

SaxParser::SaxParser(QIODevice \*device, QTreeWidget\* tree, QProgressBar \*bar)

{

progress = bar;

treeWidget = tree;

inputSource = new QXmlInputSource(device);

currentItem = NULL;

QXmlSimpleReader reader;

progress->setMinimum(1);

progress->setMaximum(100);

progress->setValue(1);

progress->show();

for(int i = 2; i < 50; ++i)

progress->setValue(i);

reader.*setContentHandler*(this);

reader.*setErrorHandler*(this);

reader.*parse*(inputSource);

for(int i = 50; i < 101; ++i)

progress->setValue(i);

}

bool SaxParser::*startElement*(const QString &namespaceURI,

const QString &localName,

const QString &qName,

const QXmlAttributes &atts)

{

if (qName == "food" || qName == "attr") {

currentItem = new QTreeWidgetItem(currentItem ?

currentItem : treeWidget->invisibleRootItem());

currentItem->setText(0, atts.value("name"));

} else if (qName == "value") {

currentText.clear();

}

return true;

}

bool SaxParser::*characters*(const QString& str)

{

currentText += str;

return true;

}

bool SaxParser::*endElement*(const QString &namespaceURI,

const QString &localName,

const QString &qName)

{

if(qName == "food" || qName == "attr") {

currentItem = currentItem->parent();

} else if (qName == "value") {

if(currentItem) {

QString allPages = currentItem->text(1);

if(!allPages.isEmpty())

allPages += ", ";

allPages += currentText;

currentItem->setText(1, allPages);

}

}

return true;

}

bool SaxParser::*fatalError*(const QXmlParseException &exception)

{

QMessageBox::warning(0,

QObject::tr("SAX Parser"),

QObject::tr("Parse error at line %1, column %2:\n%3")

.arg(exception.lineNumber())

.arg(exception.columnNumber())

.arg(qPrintable(exception.message()))

);

}

**3. Результат работы**



