Человеко-машинное взаимодействие

Лекция 6

Мерзлякова Екатерина Юрьевна
к.т.н. доцент ПМиК

XML и QT

- XML (Extensible Markup Language, расширяемый язык разметки).

- средство хранения структурных данных в текстовом файле.

XML и QT

```
<?xml version = "1.0"?>
<!-- My Address Book -->
<addressbook>
   <contact number = "1">
      <name>Piggy</name>
      <phone>+49 631322187</phone>
      <email>piggy@mega.de</email>
   </contact>
   <contact number = "2">
      <name>Kermit</name>
      <phone>+49 631322181</phone>
      <email>kermit@mega.de</email>
   </contact>
</addressbook>
```

XML и QT

<empty></empty> → *<empty/>*

<empty number = "1"></empty>
<empty number = "1"/>

<!-- комментарии
-->.
<Tag></Tag> и <tag></tag>

для описания документов можно использовать теги с любыми подходящими названиями.

ТЕГИ

: Сохранение информации

ПРИЛОЖЕНИЕ

: Интерпретация

XML и QT

- QT += xml
- #include <QtXml>

- **DOM** (Document Object Model, объектная модель документа)
- **SAX** (Simple API for XML, простой API для XML)

XML и QT: DOM

- **DOM** (Document Object Model, объектная модель документа) — это стандартное API для анализа XML-документов, разработаное W3C.
- возможность представления XML-документа в виде древовидной структуры, в памяти компьютера.
- **QDomNode**, **QDomElement**, **QDomAttr** и **QDomText**.

XML и QT: чтение XML-документа

- **QDomElement –** представление элементов
- **QDomNode –** любые типы узлов
- **QDomNode** → **QDomElement** - *QDomNode::toElement()*
- *isNull()*

XML и QT : чтение XML-документа

- `TEMPLATE = app`
- `QT += xml`
- `SOURCES = main.cpp`
- `win32:CONFIG += console`
- `win32:TARGET = ../XmlDOMRead`

XML и QT : чтение XML-документа

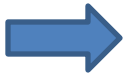
```
int main(){
 QDomDocument domDoc;
 QFile  file("addressbook.xml");
 if(file.open(QIODevice::ReadOnly)) {
  if(domDoc.setContent(&file)) {
     QDomElement domElement=domDoc.documentElement();
     traverseNode(domElement);
      }
   file.close();
 }
return 0;}
```

XML и QT : чтение XML-документа

```
#include <QtXml>
void traverseNode(const QDomNode& node)
 {
QDomNode domNode = node.firstChild();
while(!domNode.isNull()) {
      if(domNode.isElement()) {
         QDomElement domElement = domNode.toElement();
if(!domElement.isNull()) {
       if(domElement.tagName() == "contact") {
           qDebug() << "Attr: «
                    << domElement.attribute("number", "");  }
   else {
       qDebug() << "TagName: " << domElement.tagName()
                << "\tText: " << domElement.text(); }
  }
} traverseNode(domNode);
domNode = domNode.nextSibling();
} }
```

XML и QT : создание и запись XML-документа

- **QDomDocument**
- *createElement(),*
- *createTextNode(),*
- *createAttribute()*.



Объект узла

XML и QT : создание и запись XML-документа

```
int main() {
QDomDocument doc("addressbook");
QDomElement domElement = doc.createElement("adressbook");
doc.appendChild(domElement);

QDomElement contact1 = contact(doc, "Piggy", "+49 631322187",
"piggy@mega.de");
QDomElement contact2 = contact(doc, "Kermit", "+49 631322181",
"kermit@mega.de");
QDomElement contact3 = contact(doc, "Gonzo", "+49 631322186",
"gonzo@mega.de");

domElement.appendChild(contact1);
domElement.appendChild(contact2);
domElement.appendChild(contact3);

QFile file("adressbook.xml");
if(file.open(QIODevice::WriteOnly)) {
QTextStream(&file) << doc.toString();
file.close();
}
return 0;
}
```

XML и QT: создание и запись XML-документа

```
QDomElement contact(   QDomDocument& domDoc,
                       const QString& strName,
                       const QString& strPhone,
                       const QString& strEmail  )
{
static int nNumber = 1;

QDomElement domElement = makeElement(domDoc, "contact",
QString().setNum(nNumber) );

 domElement.appendChild(makeElement(domDoc, "name", "",
strName));
domElement.appendChild(makeElement(domDoc, "phone", "",
strPhone));
domElement.appendChild(makeElement(domDoc, "email", "", strEmail));

nNumber++;

return domElement;
}
```

XML и QT: создание и запись XML-документа

```
QDomElement makeElement(      QDomDocument& domDoc,
                         const QString& strName,
                         const QString& strAttr = QString::null,
                         const QString& strText = QString::null
                        )
{
  QDomElement domElement = domDoc.createElement(strName);

  if (!strAttr.isEmpty()) {
      QDomAttr domAttr = domDoc.createAttribute("number");
      domAttr.setValue(strAttr);
      domElement.setAttributeNode(domAttr);
   }

   if (!strText.isEmpty()) {
     QDomText domText = domDoc.createTextNode(strText);
     domElement.appendChild(domText);
   }
   return domElement;
}
```

XML и QT: SAX

- **SAX** (Simple API for XML, простой API для XML) является стандартом JavaAPI для считывания XML-докуменов.

- **QXmlSimpleReader** – XML-анализатор, базирующийся на SAX. Читает XML-документ блоками и сообщает о том, что было найдено, с помощью соответствующих методов.
- В память помещаются только фрагменты, а не весь XML-документ.

- **QXmlContentHandler**
- **QXmlEntityResolver**
- **QXmlDTDHandler**
- **QXmlErrorHandler**
- **QXmlDeclHandler**
- **QXmlLexicalHandler**

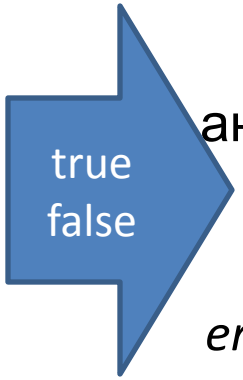
XML и QT: чтение XML-документа

- **QXmlContentHandier**
- **QXmlErrorHandler**

- *startDocument()*
- *startElement()*
- *characters* ()
- *endElement()*
- *endDocument()*

XML и QT : чтение XML-документа

- **QXmlDefaultHandler** пустые реализации виртуальных методов
- *startDocument()*,
- *startElement()*,
- *characters()*,
- *endElement()*,
- *endDocument()*
- *fatalError()* - **QXmlErrorHandler**



анализ файлов

QXmlSimpleReader

errorString()

XML и QT : чтение XML-документа

```
int main()
 {
AddressBookParser handler;
QFile file("addressbook.xml");
QXmlInputSource source(&file);
QXmlSimpleReader reader;

reader.setContentHandler(&handler);
reader.parse(source);

 return 0;
 }
```

XML и QT: чтение XML-документа

```
class AddressBookParser : public QXmlDefaultHandler {
private:
    QString m_strText;

public:
     bool startElement(const QString&,
                       const QString&,
                       const QString&,
                       const QXmlAttributes& attrs
                      )
{
      for(int i = 0; i < attrs.count(); i++) {
          if(attrs.localName(i) == "number") {
              qDebug() << "Attr:" << attrs.value(i);
            }
       } return true;
}
```

XML и QT: чтение XML-документа

```
bool characters(const QString& strText)
 {
      m_strText = strText;
      return true;
}
bool endElement(const QString&, const QString&, const
QString& str)
  {
       if (str != "contact" && str != "addressbook") {
           qDebug() << "TagName:" << str
                        << "\tText:" << m_strText;
}
return true;

}
```

XML и QT: чтение XML-документа

```
bool fatalError(const QXmlParseException&
exception)
 {
      qDebug() << "Line:" <<
exception.lineNumber()
                  << ", Column:" <<
exception.columnNumber()
                  << ", Message:" <<
exception.message();
      return false;
   }
};
```

XML и QT