Dynamically creating/updating listbox in QML from C++

This article explains how to dynamically create/update the list view in qml for which the data model is supplied from c++

Introduction

Sometimes it is required to display data in list view in QML from c++, rather than loading it statically from QML only. In the article the same is described & also the way by which we can update or change the values of that model. To explain the same an xml file has been taken which is parsed in c++ only & the parsed data is then exposed to QML listview.

Summary

First of all the main qml file(Main.qml) is loaded from c++ as initial view which contains a button, on the click of which the list view qml is loaded:

Main.cpp:

```
GlobalData*iGlobalData;
int main(int argc, char *argv[])
{
    static const QString KListingQMLPath1 = "/opt/dynamicList/qml/dynamicList/main.qml";

    QApplication app(argc, argv);
    iGlobalData = GlobalData::ReturnObject();
    dynamiclist_listviewloader* ilistviewloader;

    QDeclarativeView view;
    view.setResizeMode(QDeclarativeView::SizeRootObjectToView);
    view.setAutoFillBackground(false);

    view.rootContext()->setContextProperty("appWidth", view.geometry().width());
    view.rootContext()->setContextProperty("appHeight", view.geometry().height());
    ilistviewloader = new dynamiclist_listviewloader(&view);
    view.setSource(QUrl::fromLocalFile(KListingQMLPath1));

    QObject *rootObject = dynamic_cast<QObject*>(view.rootObject());
    QObject::connect(rootObject, SIGNAL(loadlist()),ilistviewloader, SLOT(createlistview()));  // connecting the button click event

    view.showFullScreen();
    return app.exec();
}
```

Main.qml:

```
Rectangle {
    width: 854
    height: 480
    id:page1
    signal loadlist
    Button{

        id:btn
        y:140
        anchors {
            horizontalCenter: parent.horizontalCenter
            topMargin: 10
        }
        text: qsTr("Display the list")
        onClicked:
        {
            loadlist();
        }

    }

}
```

(C) Copyright Nokia 2013. All rights reserved.

Now let us see the class responsible for creating the list view qml (which is called as a result of button press from main.qml)

Header

```
#ifndef DYNAMICLIST_LISTVIEWLOADER_H
#define DYNAMICLIST_LISTVIEWLOADER_H

#include <QObject>
#include "dynamiclist_xmlparser_itemsdata.h"
#include <QGraphicsScene>
#include <QGraphicsView>
#include <QDeclarativeContext>
#include <QDeclarativeItem>

class QDeclarativeView;
class QDeclarativeComponent;
class QDeclarativeContext;
class QDeclarativeItem;


class dynamiclist_listviewloader : public QObject
{
    Q_OBJECT


public:
    explicit dynamiclist_listviewloader( QDeclarativeView *view,QObject *parent = 0);
    dynamiclist_xmlparser_itemsData* idynamiclist_xmlparser_itemsData;
    void setNewContextinit(QDeclarativeContext*acontext);
    Q_INVOKABLE void filtersearchfilmitems(const QString& afiltersearchitems);

private:
    QDeclarativeView* main_view;
    QDeclarativeComponent *m_mainComponent; // Not owned
    QGraphicsScene *scene;
    QDeclarativeView *view;
    QDeclarativeContext*icontext;
    QDeclarativeItem* mainQml;
    QString iFilesDirPath;
    QString iScreeningFileDirPath;


public slots:

    void createlistview();
    void setNewContext();
    void setNewContextPrevious();

};

#endif // DYNAMICLIST_LISTVIEWLOADER_H
```

Source

```
extern GlobalData*iGlobalData;
static const QString KListingQMLPath = "/opt/dynamicList/qml/dynamicList/ListingPage.qml";
static const QString KListXMLPath = "/opt/dynamicList/qml/dynamicList/xml/Items.xml";

dynamiclist_listviewloader::dynamiclist_listviewloader(QDeclarativeView *view,QObject *parent) :
    QObject(parent), main_view(view)
{

 '''//First of all parse the XML file & save its data'''

    iFilesDirPath.append(KListXMLPath);
    QFile film_xmlfile(iFilesDirPath);
    int ifFileExists = film_xmlfile.exists(iFilesDirPath);
    idynamiclist_xmlparser_itemsData = new dynamiclist_xmlparser_itemsData();
    idynamiclist_xmlparser_itemsData->ParseScreeningsXmlData(iFilesDirPath);
}

// for creating film listing view

void dynamiclist_listviewloader::createlistview()
{

    m_mainComponent = new QDeclarativeComponent(main_view->engine(),
                                                QUrl::fromLocalFile(KListingQMLPath));
    //QDeclarativeItem *mainQml =
    mainQml =  qobject_cast<QDeclarativeItem*>(m_mainComponent->create());
    main_view->scene()->addItem(mainQml);

    QDeclarativeContext *context = main_view->engine()->rootContext();

'''// Exposing this class to listview qml , so as to call its functions from it'''

    context->setContextProperty("listloader",this);
    context->setContextProperty("xmldatalistmodel1",idynamiclist_xmlparser_itemsData);

'''// Exposing the list data model  , so as to set it as model for list view qml'''
    context->setContextProperty("xmldatalistmodel", QVariant::fromValue(idynamiclist_xmlparser_itemsData->ExtractItems()));

    setNewContextinit(context);
    QObject::connect(mainQml, SIGNAL(loadnewmodel()),this, SLOT(setNewContext(/*context*/)));
    QObject::connect(mainQml, SIGNAL(loadnewmodelprevious()),this, SLOT(setNewContextPrevious()));

    main_view->showFullScreen();

}

void dynamiclist_listviewloader::setNewContextinit(QDeclarativeContext*acontext)
{
    icontext = acontext;
}
```

Now, the class responsible for XML parsing & saving the data:

(C) Copyright Nokia 2013. All rights reserved.

Header

```
class dynamiclist_xmlparser_itemsData : public QObject
{
    Q_OBJECT
    Q_PROPERTY(QString getiResultPageCount READ getiResultPageCount WRITE SetiResultPageCount NOTIFY ResultPageCountChanged)
    Q_PROPERTY(QString getiResultPageCountUpper_Lower READ getiResultPageCountUpper_Lower WRITE SetiResultUpper_Lower NOTIFY ResultP

public:
    explicit dynamiclist_xmlparser_itemsData(QObject *parent = 0);
    QList<QObject*>ExtractItems();
    QList<QObject*>ExtractItemsNext();
    QList<QObject*>ExtractItemsPrevious();
    QList<QObject*>ExtractItemsFromSearch(const QString& section);

    QList<QObject*> ItemsMasterList;
    QString iResultPageCountStr;
    QString iResultPageCountUpper_LowerStr;
    int iRemainingItemCount;

signals:
    void ResultPageCountChanged();
    void ResultPageCount_upperChanged();
    void uppercountchanged();
    void lowercountchanged();

public slots:
      void ParseScreeningsXmlData(const QString& aScreeningFilePath);

      void SetiResultPageCount(const QString& iResultPageCount );
      QString getiResultPageCount();

      void SetiResultUpper_Lower(const QString& iResultPageCount );
      QString getiResultPageCountUpper_Lower();


};
```

Source

```
#include <QDomDocument>
#include "dynamiclist_xmlparser_itemsData.h"
#include "dynamicList_common_globaldata.h"

static const QString KNodeElement = "item";
static const QString KItemNum = "ITEM_NO";
static const QString KItemId = "ITEM_ID";
static const QString KItemName = "NAME";

extern GlobalData*iGlobalData;

dynamiclist_xmlparser_itemsData::dynamiclist_xmlparser_itemsData(QObject *parent) :
    QObject(parent)
{
}

void dynamiclist_xmlparser_itemsData::ParseScreeningsXmlData(const QString& aScreeningFilePath)
{
    QFile* file = new QFile(aScreeningFilePath);
    /* If we can't open it, let's show an error message. */
    if (!file->open(QIODevice::ReadOnly | QIODevice::Text)) {

        return;
    }

    QDomDocument doc("mydocument");
    if (!doc.setContent(file))
    {
        return;
    }

    //Get the root element
    iGlobalData->iListDataArray.clear();
    QDomElement docElem = doc.documentElement();

    // you could check the root tag name here if it matters
    QString rootTag = docElem.tagName(); // == root

    // get the node's interested in, this time only caring about person's
    QDomNodeList nodeList = docElem.elementsByTagName(KNodeElement);

    //Check each node one by one.
    for(int iNodeCount = 0;iNodeCount < nodeList.count(); iNodeCount++)
    {
        dynamiclist_dto_listnodedata*idynamiclist_dto_listnodedata = new dynamiclist_dto_listnodedata();

        // get the current one as QDomElement
        QDomElement el = nodeList.at(iNodeCount).toElement();



        //get all data for the element, by looping through all child elements
        QDomNode pEntries = el.firstChild();
        while(!pEntries.isNull()) {
            QDomElement peData = pEntries.toElement();
            QString tagNam = peData.tagName();

            if(tagNam == KItemNum) {
                idynamiclist_dto_listnodedata->SetItemNumber(peData.text());
            }else if(tagNam == KItemId) {
                idynamiclist_dto_listnodedata->SetItemId(peData.text());
            }else if(tagNam == KItemName) {
                idynamiclist_dto_listnodedata->SetItemName(peData.text());
            }
            pEntries = pEntries.nextSibling();
        }

        iGlobalData->iListDataArray.append(idynamiclist_dto_listnodedata);
        ItemsMasterList.append(idynamiclist_dto_listnodedata);
    }
```


(C) Copyright Nokia 2013. All rights reserved.

```
    file->close();
    if(ItemsMasterList.count()%10==0)
        iGlobalData->iResultPageCount = (ItemsMasterList.count()/10);
    else
    {
        iGlobalData->iResultPageCount = (ItemsMasterList.count()/10);
        iGlobalData->iResultPageCount = iGlobalData->iResultPageCount+1;
    }

    QString qStr = QString::number(iGlobalData->iResultPageCount);
    SetiResultPageCount(qStr);

}
// For extracting first 10 elements from array

QList<QObject*> dynamiclist_xmlparser_itemsData::ExtractItems()   // called from dynamiclist_listviewloader::createlistview()
{
    QList<QObject*>iTempItemsList;

    for(int ifilmitemcount = iGlobalData->iInitialNumRecords;ifilmitemcount<iGlobalData->iFinalNumRecords;ifilmitemcount++)
    {
        iTempItemsList.append(ItemsMasterList.at(ifilmitemcount));
    }

    QString qStr = QString::number(iGlobalData->upperpagecount);
    SetiResultUpper_Lower(qStr);

    return iTempItemsList;
}
```

Listingpage. qml :

```
import QtQuick 1.1
import com.nokia.meego 1.0

Rectangle {
    id: listpage
    width: 854
    height: 480
    signal loadnewmodel     // for next
    signal loadnewmodelprevious
    property string itextfieldval
    Rectangle {

        width: 854;
        height: 300
        x:20
        y:120


        ListView {
            id: xmldatalistingview
            anchors.fill: parent
            model: xmldatalistmodel
            delegate: xmldatalistdelegate
            width:854

            focus: true
            clip : true

        }

        Component {
            id: xmldatalistdelegate
            XmlListingItem {
            }
        }
        Text {

            id:textid
            x: 400
            y: 130
            opacity:1
            text:xmldatalistingview.count > 0 ? "" : "No Record(s) Found"
            font.bold: true; font.pointSize: 20
            color:"black"

        }

    }

    Rectangle{
        width: 854
        height: 55
        x:0
        y:65

        TextField {
            id: textfield1
            x: 18
            width: 830
            Image {
                id: search
                x: 780
                y: 12

                width:30
                height:30
            }
            onTextChanged:
            {
                itextfieldval = textfield1.text
                listloader.filtersearchfilmitems(textfield1.text)  // keyword entered in text field for filtering
            }

        }
    }
    Image {
        id: toolbar
        width: 854
        height: 40

        x: 7
        y: 440

        source: "images/toolbarimage.png"
    }
```


(C) Copyright Nokia 2013. All rights reserved.

```
    Column
    {
        id:leftarrowcolumn
        x:250
        width:40
        height:40
        y:450

        Image {
            id: leftarrow
            width: 22
            height: 24

            x: 7
            y: 10

            source: "images/arrow_left.png"
        }

        MouseArea {
            id: la
            anchors.fill: parent
            onClicked: {
                loadnewmodelprevious(); //For loading previous 'N' results
            }
        }

    }

    Text {
        id:text

        width: 100
        height: 30

        x: 300
        y: 450

        text: xmldatalistingview.count > 0 ? "Page" + " "+ xmldatalistmodel1.getiResultPageCountUpper_Lower + " "+ "of" +" "+ xmldat
        opacity:1
        font.bold: true;
        font.pointSize: 16
        color: "white"

    }

    Column
    {
        id:rightarrowcolumn
        x:430
        width:65
        height:40
        y:450

        Image {
            id: righttarrow
            width: 22
            height: 24

            x: 13
            y: 10

            source: "images/arrow_right.png"
        }

        MouseArea {
            id: ra
            anchors.fill: parent
            onClicked: {

                loadnewmodel(); // for loading next 'N' results

            }
        }
    }
}
```

And the listing view delegate (Xmllistingitem.qml)

```
import QtQuick 1.0
import Qt 4.7
import com.nokia.meego 1.0


Item {

    id: listitems
    signal loadnewmodel
    property string samplename: "value1"
    property string fontName: "Helvetica"
    property int titleFontSize: 17
    property int fontSize: 14
    property color fontColorTitle: "black"
    property color fontColor: "black"
    property int margins: 8
    property real detailsOpacity : 0

    width: xmldatalistingview.width
    height: 85


    // Lay out the page: picture, title and ingredients at the top, and method at the
    // bottom.  Note that elements that should not be visible in the list
    // mode have their opacity set to recipe.detailsOpacity.
    Row {
        id: topLayout
        x: 10; y: 18; height: 100; width: parent.width
        spacing: 16


        Column {
            // Leave some room on top.
            y: 2
            x: 13
```


(C) Copyright Nokia 2013. All rights reserved.

```
            Text {
                id: listitemnum

                anchors {
                    leftMargin: listitems.margins
                    rightMargin: listitems.margins
                }

                text: model.modelData.getItemNumber //get item number
                wrapMode: Text.WordWrap
                font {
                    family: listitems.fontName
                    pointSize: listitems.titleFontSize

                }
                font.bold: true;
                color: fontColorTitle

            }
            Text {
                id: listitemid
                y: 4

                anchors {
                    left: parent.left
                    rightMargin: listitems.margins*3
                }

                text: model.modelData.getItemId // get item id
                font {
                    family: listitems.fontName
                    pointSize: listitems.fontSize
                }
                color: fontColor
            }


            Text {
                id: listitemname
                y: 6
                width: parent.width
                anchors {
                    leftMargin: listitems.margins
                    rightMargin: listitems.margins
                }
                text: model.modelData.getItemName // get item name
                wrapMode: Text.WordWrap
                font {
                    family: listitems.fontName
                    pointSize: listitems.fontSize
                }
                color: listitems.fontColor
            }


        }



    }

}
```

Now when the right arrow image is clicked the signal loadnewmodel() from listingpage.qml is called as shown above, which calls the slot dynamiclist_listviewloader::setNewContext(). This method extracts the next 'N'(here its 10) from the XML parsed data array & sets this as new context for listingpage.qml

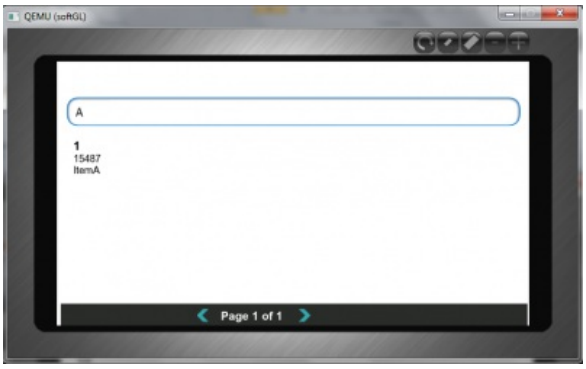
```
void dynamiclist_listviewloader::setNewContext()
{
    if(iGlobalData->upperpagecount<iGlobalData->iResultPageCount)
    {
//using the same QDeclarativeContext(icontext) variable which was used initially to set the context for listingpage.qml while it was
        icontext->setContextProperty("xmldatalistmodel", QVariant::fromValue(idynamiclist_xmlparser_itemsData->ExtractItemsNext()));
    }
}
```


(C) Copyright Nokia 2013. All rights reserved.

Similarly it can be done on the press of previous image icon & as a result previous 'N'(here 10) items from XML data array are loaded in listingpage.qml

Now a user may also use the text field as provided in the ListingPage.qml so as to filter the list items on the basis of keyword entered in the text field:

```
void dynamiclist_listviewloader::filtersearchfilmitems(const QString& afiltersearchitems)
{
   //afiltersearchitems: keyword entered in text field in ListingPage.qml
    icontext->setContextProperty("xmldatalistmodel", QVariant::fromValue(idynamiclist_xmlparser_itemsData->ExtractItemsFromSearch(af
}
```



Source Code

The full source code can be downloaded from here : File:DynamicList.zip

NOTE : The full implementation of classes : dynamiclist_xmlparser_itemsData, dynamiclist_listviewloader & dynamiclist_dto_listnodedata can be found in the project that is attached.

Some References

exposing c++ data models

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.