Calling Qt class methods from QML

This code snippet shows how to call Qt class methods directly from QML.

Overview

In order to enable the calling of Qt class methods from QML code, the methods must be declared as public slots or conventional methods with the `Q_INVOKABLE` macro. In both cases, Qt methods are available to the Qt Meta-Object system and the methods are callable from QML. One drawback in the public slot method is that the slot cannot return a value. Conversely, the method with the `Q_INVOKABLE` macro is able to do this.

In the following demonstration we define a new Qt class, `StringHelper`, which has one invokable `echo` method and one public slot, `toggleEcho`. The `echo` method will return the given string as is, or as reversed if the public slot `toggleEcho(true)` has been called.

The `StringHelper` class is instantiated in **main.cpp**. The `QObject`-derived object `stringHelper` is then set as a context property to the root element of `QDeclarativeView`. The result is that the root element `rect` in the `ui.qml` document will see `StringHelper` as its child element.

Preconditions

- Qt 4.7 or higher is installed on your platform.

Qt Project File

```
#To make sure we use declarative
QT += declarative

#To get files deployed on device / emulator
files.sources += ui.qml
DEPLOYMENT += files
```

Source

stringhelper.h

```
#ifndef STRINGHELPER_H
#define STRINGHELPER_H

#include <QObject>
#include <QString>

class StringHelper : public QObject
{
    Q_OBJECT
public:
    StringHelper(QObject *parent = 0) : QObject(parent), reverse(false) { }

    Q_INVOKABLE QString echo(const QString &text) const {
        if(reverse == false) { return text; }

        QString reversed;
        for(QString::const_iterator it = text.begin(); it != text.end(); it++) {
            reversed.push_front(*it);
        }

        return reversed;
    }

public slots:
    void toggleEcho(bool reverse) { this->reverse = reverse; }

protected:
    bool reverse;
};

#endif // STRINGHELPER_H
```

ui.qml

```
import Qt 4.7

Rectangle {
    id: rect

    property string text: "Using Qt class to echo this"

    function updateUI() {
        StringHelper.toggleEcho(button.pressed); // calling StringHelper::toggleEcho
        text.text = StringHelper.echo(rect.text) // calling StringHelper::echo
    }

    anchors.fill: parent
    color: "black"

    Component.onCompleted: updateUI()

    Text {
        id: text
```


(C) Copyright Nokia 2013. All rights reserved.

```
        anchors.centerIn: parent
        color: "white"
    }

    Rectangle {
        id: button

        property bool pressed: false

        width: 100; height: 40
        anchors.right: parent.right; anchors.rightMargin: 20
        anchors.bottom: parent.bottom; anchors.bottomMargin: 20
        radius: 6
        color: pressed ? "gray" : "white"

        Text {
            anchors.centerIn: parent
            text: "Reverse"
        }

        MouseArea {
            anchors.fill: parent
            onClicked: { button.pressed = !button.pressed; updateUI() }
        }
    }
}
```

main.cpp

```
#include <QApplication>
#include <QDeclarativeView>
#include <QDeclarativeContext>
#include "stringhelper.h"

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    StringHelper stringHelper;

    QDeclarativeView view;
    view.setResizeMode(QDeclarativeView::SizeRootObjectToView);
    view.rootContext()->setContextProperty("StringHelper", &stringHelper);
    view.setSource(QUrl("./ui.qml"));

#if defined(Q_WS_S60) || defined(Q_WS_MAEMO)
    view.showMaximized();
#else
    view.setGeometry(100, 100, 800, 480);
    view.show();
#endif

    return a.exec();
}
```

Postconditions

The code snippet has demonstrated how to call Qt class methods from QML. This required setting the `QObject`-derived object as a context property to the root object of `QDeclarativeView`. The methods of the Qt-derived class had to be either public slots or regular methods declared with the `Q_INVOKABLE` macro.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.