Exposing QObjects to Qt Webkit

This snippet demonstrates how to expose QObjects to the Qt WebKit.

Code

sampleqobject.h

Naturally we need a QObject to expose. When exposing QObjects to the Qt Webkit to be used from JavaScript, remember that only signals and slots will be available.

```
#ifndef SAMPLEQOBJECT_H
#define SAMPLEQOBJECT_H

#include <QtCore/QObject>
#include <QtCore/QMap>
#include <QtCore/QString>
#include <QtCore/QVariant>

class SampleQObject : public QObject
{
    Q_OBJECT
public:
    explicit SampleQObject(QObject *parent = 0);

signals:
    void signal(QMap<QString, QVariant> object);
public slots:
    QMap<QString, QVariant> slotThatReturns(const QMap<QString,
                                            QVariant>& object);
    void slotThatEmitsSignal();

private:
    int m_signalEmited;
    QMap<QString, QVariant> m_returnObject;
    QMap<QString, QVariant> m_emitSignal;
};

#endif // SAMPLEQOBJECT_H
```

sampleobject.cpp

Here are the implementations for signals and slots.

```
#include "sampleqobject.h"

#include <QtCore/QDebug>

SampleQObject::SampleQObject(QObject *parent) :
    QObject(parent)
{
    m_signalEmited = 0;
}

QMap<QString, QVariant>
```


(C) Copyright Nokia 2014. All rights reserved.

```
SampleQObject::slotThatReturns(const QMap<QString, QVariant>& object)
{
    qDebug() << "SampleQObject::slotThatReturns";
    this->m_returnObject.clear();
    this->m_returnObject.unite(object);
    QString addedBonus = QString::number(object["intValue"].toInt(),
                                         10).append(" added bonus.");
    this->m_returnObject["stringValue"] = QVariant(addedBonus);
    qDebug() << "SampleQObject::slotThatReturns" << this->m_returnObject;
    return this->m_returnObject;
}

void
SampleQObject::slotThatEmitsSignal()
{
    qDebug() << "SampleQObject::slotThatEmitsSignal";
    this->m_signalEmited++;
    this->m_emitSignal.clear();
    this->m_emitSignal["signalsEmited"] = QVariant(this->m_signalEmited);
    this->m_emitSignal["sender"] = QVariant("SampleQObject::slotThatEmitsSignal");
    qDebug() << "SampleQObject::slotThatEmitsSignal" << this->m_emitSignal;
    emit signal(this->m_emitSignal);
}
```

mainwindow.ui

This is the UI file for the snippet. It references the QWebView which will be used to expose the QObjects.

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>600</width>
    <height>400</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget" name="centralWidget">
   <layout class="QVBoxLayout" name="verticalLayout">
    <property name="spacing">
     <number>0</number>
    </property>
    <property name="margin">
     <number>0</number>
    </property>
    <item>
     <widget class="QWebView" name="webView">
      <property name="url">
       <url>
        <string>qrc:/view.html</string>
```


(C) Copyright Nokia 2014. All rights reserved.

```
        </url>
       </property>
      </widget>
     </item>
    </layout>
   </widget>
  </widget>
  <layoutdefault spacing="6" margin="11"/>
  <customwidgets>
   <customwidget>
    <class>QWebView</class>
    <extends>QWidget</extends>
    <header>QtWebKit/QWebView</header>
   </customwidget>
  </customwidgets>
  <resources/>
  <connections/>
 </ui>
```

mainwindow.h

Next, we need to tie these all together in a QMainWindow descendant.

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include "sampleqobject.h"

namespace Ui {
    class MainWindow;
}

class MainWindow : public QMainWindow {
    Q_OBJECT
public:
    MainWindow(QWidget* parent = 0);
    ~MainWindow();

private slots:
    void addJavaScriptObject();

protected:
    void changeEvent(QEvent *e);

private:
    Ui::MainWindow* ui;
    SampleQObject* m_sampleQObject;
};

#endif // MAINWINDOW_H
```

mainwindow.cpp

```
#include "mainwindow.h"
```


(C) Copyright Nokia 2014. All rights reserved.

```
#include "ui_mainwindow.h"

#include <QtWebKit/QWebPage>
#include <QtWebKit/QWebFrame>

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    /**
      * This is the interesting part, here we create the QObject
      * and connect the QWebViews pages main frames
      * javaScriptWindowObjectCleared signal to our addJavaScriptObject slot
      * which then exposes the QObject to the WebKit.
      *
      * QWebFrame::javaScriptWindowObjectCleared signal is emitted whenever
      * the global window object of the JavaScript environment is cleared,
      * e.g., before starting a new load.
      * If you intend to add QObjects to a QWebFrame using
      * addToJavaScriptWindowObject(), you should add them in a slot connected
      * to this signal. This ensures that your objects remain accessible when
      * loading new URLs.
      * http://doc.qt.nokia.com/4.7/qwebframe.html#javaScriptWindowObjectCleared
      */
    m_sampleQObject = new SampleQObject(this);
    connect(ui->webView->page()->mainFrame(),
            SIGNAL(javaScriptWindowObjectCleared()),
            this,
            SLOT(addJavaScriptObject()));
}

MainWindow::~MainWindow()
{
    delete ui;
    delete m_sampleQObject;
}

void
MainWindow::changeEvent(QEvent *e)
{
    QMainWindow::changeEvent(e);
    switch (e->type()) {
    case QEvent::LanguageChange:
        ui->retranslateUi(this);
        break;
    default:
        break;
    }
}

void
MainWindow::addJavaScriptObject()
{
    /**
      * This code calls another interesting method
```


(C) Copyright Nokia 2014. All rights reserved.

```
     * QWebFrame::addJavaScriptObject
     * which can expose any QObject to the JavaScript context with the
     * name given as the first parameter.
     */
    this->ui->webView->page()
                     ->mainFrame()
                     ->addToJavaScriptWindowObject("sampleQObject",
                                                   this->m_sampleQObject);
}
```

Postconditions

You've successfully exposed the QObject to the Qt WebKit's web context.

See also

Code Snippets for calling slots and connecting signals to JavaScript slots:

- Calling an exposed QObject slot from Qt WebKit with JavaScript
- Connecting to a QObjects signal with JavaScript slot in Qt WebKit

(C) Copyright Nokia 2014. All rights reserved.