



ID	CS001432	Creation date	June 16, 2009
Platform	Qt	Tested on devices	5800 XpressMusic
Category	Qt for Symbian	Subcategory	Networking

Keywords (APIs, classes, methods, functions): QNetworkAccessManager, HTTP redirect

Overview

This code shows how to handle an HTTP redirect with `QNetworkAccessManager`.

Note: In order to use this code, you need to have Qt for S60 installed on your platform.

Preconditions

- Install latest Qt for S60 see [Qt for S60 - Installation packages](#)
- Check this link for installation guide: [How to install the package.](#)
- Go through this article: [Getting started with Qt for S60](#)

Header file

```

/*
 * Copyright (c) 2009 Nokia Corporation
 */

#ifndef QNAMREDIRECT_H
#define QNAMREDIRECT_H

#include <QtGui/QMainWindow>
#include <QtGui/QFrame>
#include <QtGui/QVBoxLayout>
#include <QtGui/QLabel>
#include <QtGui/QPushButton>

#include <QtNetwork/QNetworkAccessManager>
#include <QtNetwork/QNetworkRequest>
#include <QtNetwork/QNetworkReply>

#include <QtCore/QPointer>
#include <QtCore/QUrl>

```

CS001432_-_Handling_an_HTTP_redirect_with_QNetworkAccessManager

```
class QNAMRedirect : public QMainWindow
{
    Q_OBJECT

public:
    QNAMRedirect(QWidget *parent = 0);
    ~QNAMRedirect() {}

private slots:
    void doRequest();
    void replyFinished(QNetworkReply* reply);

private:
    QNetworkAccessManager* _qnam;
    QUrl _originalUrl;
    QUrl _urlRedirectedTo;

    QLabel* _textContainer;

    QNAMRedirect(QNAM());
    QNAMRedirect(const QUrl& possibleRedirectUrl,
                 const QUrl& oldRedirectUrl) const;
};

#endif // QNAMREDIRECT_H
```

Source file

```
/*
 * Copyright (c) 2009 Nokia Corporation
 */

#include "qnamredirect.h"

QNAMRedirect::QNAMRedirect(QWidget *parent) : QMainWindow(parent) {
    _qnam = new QNetworkAccessManager(this);
    _originalUrl = QUrl("http://www.wikipedia.org/wiki/URL_redirection");

    QVBoxLayout = new QVBoxLayout;

#ifdef Q_OS_SYMBIAN
    setStyleSheet("QPushButton { font: 5pt; }");
#endif

    _textContainer = new QLabel("Click the button to test URL redirect!");
    _textContainer->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);
    _textContainer->setAlignment(Qt::AlignCenter);
    _textContainer->setWordWrap(true);
    addWidget(_textContainer);

    QPushButton = new QPushButton("Click here!");
    /* If the button is clicked, we'll do a request. */
    connect(button, SIGNAL(clicked()),
            this, SLOT(doRequest()));
    addWidget(testButton);

    * QFrame = new QFrame;
    setLayout(layout);
    setCentralWidget(* QFrame);
}
```

Header file

CS001432_-_Handling_an_HTTP_redirect_with_QNetworkAccessManager

```

QNetworkAccessManager* QNAMRedirect::createQAM() {
    QNetworkAccessManager* qnam = new QNetworkAccessManager(this);
    if(_qnam) {
        ->deleteLater(qnam);
    }
    if(_textContainer) {
        ->deleteLater(text);
    }
}

QNetworkAccessManager* QNAMRedirect::createQAM() {
    QNetworkAccessManager* qnam = new QNetworkAccessManager(this);
    /* We'll handle the finished reply in replyFinished */
    connect(qnam, SIGNAL(finished(QNetworkReply*)),
            this, SLOT(replyFinished(QNetworkReply*)));
    return qnam;
}

void QNAMRedirect::doRequest() {
    QString text("QNAMRedirect::doRequest doing request to ");
    append(this->_originalUrl.toString());
    ->_textContainer->setText(text);
    /* Let's just create network request for this predefined URL... */
    QNetworkRequest request(this->_originalUrl);
    /* ...and ask the manager to do the request. */
    ->_qnam->get(request);
}

void QNAMRedirect::replyFinished(QNetworkReply* reply) {
    /*
     * Reply is finished!
     * We'll ask for the reply about the Redirection attribute
     * http://doc.trolltech.com/qnetworkrequest.html#Attribute-enum
     */
    QVariant possibleRedirectUrl
        = reply->attribute(QNetworkRequest::RedirectionTargetAttribute);

    /* We'll deduct if the redirection is valid in the redirectUrl function */
    _urlRedirectedTo = possibleRedirectUrl.toUrl(),
        _urlRedirectedTo

    /* If the URL is not empty, we're being redirected. */
    if(!_urlRedirectedTo.isEmpty()) {
        QString text("QNAMRedirect::replyFinished: Redirected to ")
            + _urlRedirectedTo.toString();
        ->_textContainer->setText(text);

        /* We'll do another request to the redirection url. */
        ->_qnam->get(QNetworkRequest(_urlRedirectedTo));
    }
    else {
        /*
         * We weren't redirected anymore
         * so we arrived to the final destination...
         */
        QString text("QNAMRedirect::replyFinished: Arrived to ")
            + reply->url().toString();
        ->_textContainer->setText(text);
        /* ...so this can be cleared. */
        _urlRedirectedTo = QUrl();
    }
    /* Clean up. */
}

```

CS001432 - Handling an HTTP redirect with QNetworkAccessManager

```
->deferLater();
}

QUrl QNAMRedirect::redirectUrl(const QUrl& possibleRedirectUrl,
                              const QUrl& oldRedirectUrl) const {
    QUrl redirectUrl
/*
    * Check if the URL is empty and
    * that we aren't being fooled into a infinite redirect loop.
    * We could also keep track of how many redirects we have been to
    * and set a limit to it, but we'll leave that to you.
    */
    if(!possibleRedirectUrl.isEmpty() &&
        possibleRedirectUrl == oldRedirectUrl) {
        = possibleRedirectUrl;
    }
    return redirectUrl;
}
```

Postconditions

You are able to handle HTTP redirects with QNetworkAccessManager.

Supplementary material

- You can test QNAMRedirect with a test application. The application is available for download at [Media:QNAMRedirect.zip](#).

See also

- [CS001431 - Creating an HTTP network request in Qt](#)
- [QNetworkAccessManager documentation](#)
- [QNetworkRequest documentation](#)