How to move a QObject to a thread

Overview

Usually developers subclass QThread in order to have multi-threads Qt applications. This snippet shows to the reader another way to use threads with Qt. The QObject::moveToThread method, indeed, permits developers to move any QObject from the main application thread to another one. The moveToThread() function tells Qt to ensure that event handlers, and by extension signals and slots, are called from the specified thread context.

Source code

main.cpp

```
#include <QtCore>
#include <stdio.h>

enum {
    Limit = 123456,
    BlockSize = 7890
};

class Producer : public QObject
{
    Q_OBJECT
    QByteArray data;
    int bytes;

public:
    inline Producer() : bytes(0) { }

public slots:
    void produce()
    {
        int remaining = Limit - bytes;
        if (remaining == 0) {
            emit finished();
            return;
        }

        // this will never happen
        if (data.size() != 0)
            qFatal("Producer: Consumer failed to consume!");

        int size = qMin(int(BlockSize), remaining);
        bytes += size;
        remaining -= size;
        data.fill('Q', size);

        printf("Producer: produced %d more bytes, %d of %d total\n", size, bytes, Limit);
        emit produced(&data);
    }

signals:
    void produced(QByteArray *data);
    void finished();
};

class Consumer : public QObject
{
    Q_OBJECT
    int bytes;

public:
    inline Consumer() : bytes(0) { }

public slots:
    void consume(QByteArray *data)
    {
        // this will never happen
        if (data->size() == 0)
            qFatal("Consumer: Producer failed to produce!");

        int remaining = Limit - bytes;
        int size = data->size();
        remaining -= size;
        bytes += size;
        data->clear();

        printf("Consumer: consumed %d more bytes, %d of %d total\n", size, bytes, Limit);
        emit consumed();
        if (remaining == 0)
            emit finished();
    }

signals:
    void consumed();
    void finished();
};

int main(int argc, char **argv)
{
    QCoreApplication app(argc, argv);
```


(C) Copyright Nokia 2013. All rights reserved.

```
    // create the producer and consumer and plug them together
    Producer producer;
    Consumer consumer;
    producer.connect(&consumer,
                     SIGNAL(consumed()),
                     SLOT(produce()));
    consumer.connect(&producer,
                     SIGNAL(produced(QByteArray *)),
                     SLOT(consume(QByteArray *)));

    // they both get their own thread
    QThread producerThread;
    producer.moveToThread(&producerThread);
    QThread consumerThread;
    consumer.moveToThread(&consumerThread);

    // start producing once the producer's thread has started
    producer.connect(&producerThread,
                     SIGNAL(started()),
                     SLOT(produce()));

    // when the consumer is done, it stops its thread
    consumerThread.connect(&consumer,
                           SIGNAL(finished()),
                           SLOT(quit()));
    // when consumerThread is done, it stops the producerThread
    producerThread.connect(&consumerThread,
                           SIGNAL(finished()),
                           SLOT(quit()));
    // when producerThread is done, it quits the application
    app.connect(&producerThread,
                SIGNAL(finished()),
                SLOT(quit()));

    // go!
    producerThread.start();
    consumerThread.start();

    return app.exec();
}

#include "main.moc"
```

producerconsumer.pro

```
SOURCES = main.cpp
QT = core
CONFIG += console
```

Links

- http://labs.qt.nokia.com/2006/12/04/threading-without-the-headache/
- http://labs.qt.nokia.com/2007/07/05/qthreads-no-longer-abstract/
- http://labs.qt.nokia.com/2010/06/17/youre-doing-it-wrong/

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.