Handling right button clicks using Qt

This article discusses two strategies for handling right button clicks using Qt: the first and simplest being to implement the `QObject` virtual method `eventFilter`, the second being to create a generic event handler for your objects. The article covers both strategies and provides working code snippets. It is possible to apply these techniques to Maemo or PC programs, without modifications. However, in Maemo programs it is necessary to press and hold the screen for some instants in order to generate the right click event since a mouse is not available.

Right click by sub-classing

The virtual **QObject** method called **eventFilter** can be used to handler any event received by the object. It is possible to decide if you want to consume the event (returning false in your reimplementation) or delegate it for further processing by other software layers (calling the event handler of parent class). Moreover, it is necessary to enable the event filter as well, calling the method **installEventFilter**.

This technique may be used for any object derived from **QObject**. For instance, suppose you have a QListWidget and want to create a popup menu when right button is clicked. The next two files, **rcqlistwidget.h** and **rcqlistwidget.cpp** implement the new list widget (called RCQListWidget) and the event handler.

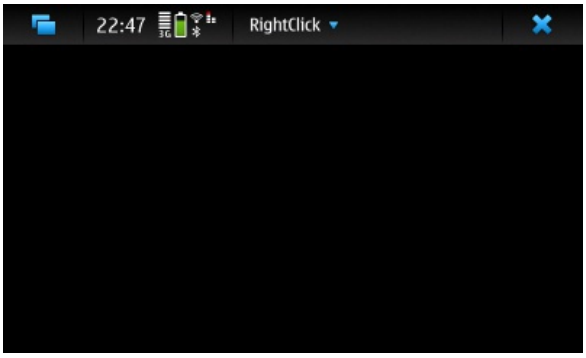
rcqlistwidget.h

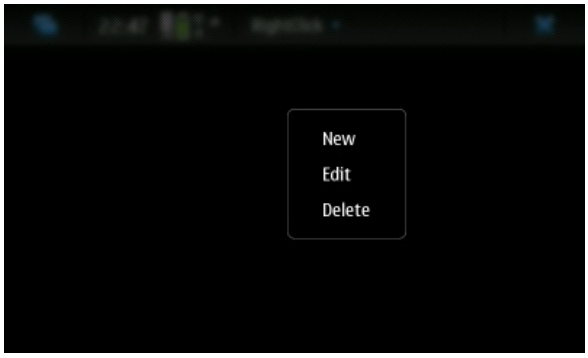
```
#ifndef RCQLISTWIDGET_H
#define RCQLISTWIDGET_H

#include <QListWidget>

class RCQListWidget : public QListWidget
{
Q_OBJECT
public:
    explicit RCQListWidget(QWidget *parent = 0);

private:
    bool eventFilter(QObject *, QEvent *);

};

#endif // RCQLISTWIDGET_H
```

rcqlistwidget.cpp

```
#include "rcqlistwidget.h"
#include <QMouseEvent>
#include <QMenu>

RCQListWidget::RCQListWidget(QWidget *parent) : QListWidget(parent)
{
    installEventFilter(this);
}

bool RCQListWidget::eventFilter(QObject *obj, QEvent *event)
{
    if(event->type() == QEvent::ContextMenu)
    {
        QMouseEvent *mouseEvent = static_cast<QMouseEvent*> (event);
        QMenu *menu = new QMenu(this);

        menu->addAction(new QAction("New",this));
        menu->addAction(new QAction("Edit",this));
        menu->addAction(new QAction("Delete",this));
        menu->exec(mouseEvent->globalPos());

        return false;
    }
    else
        return QListWidget::eventFilter(obj, event);
}
```

When the right button is clicked (or long press in Maemo), the menu is displayed, as showed in the following screenshots:

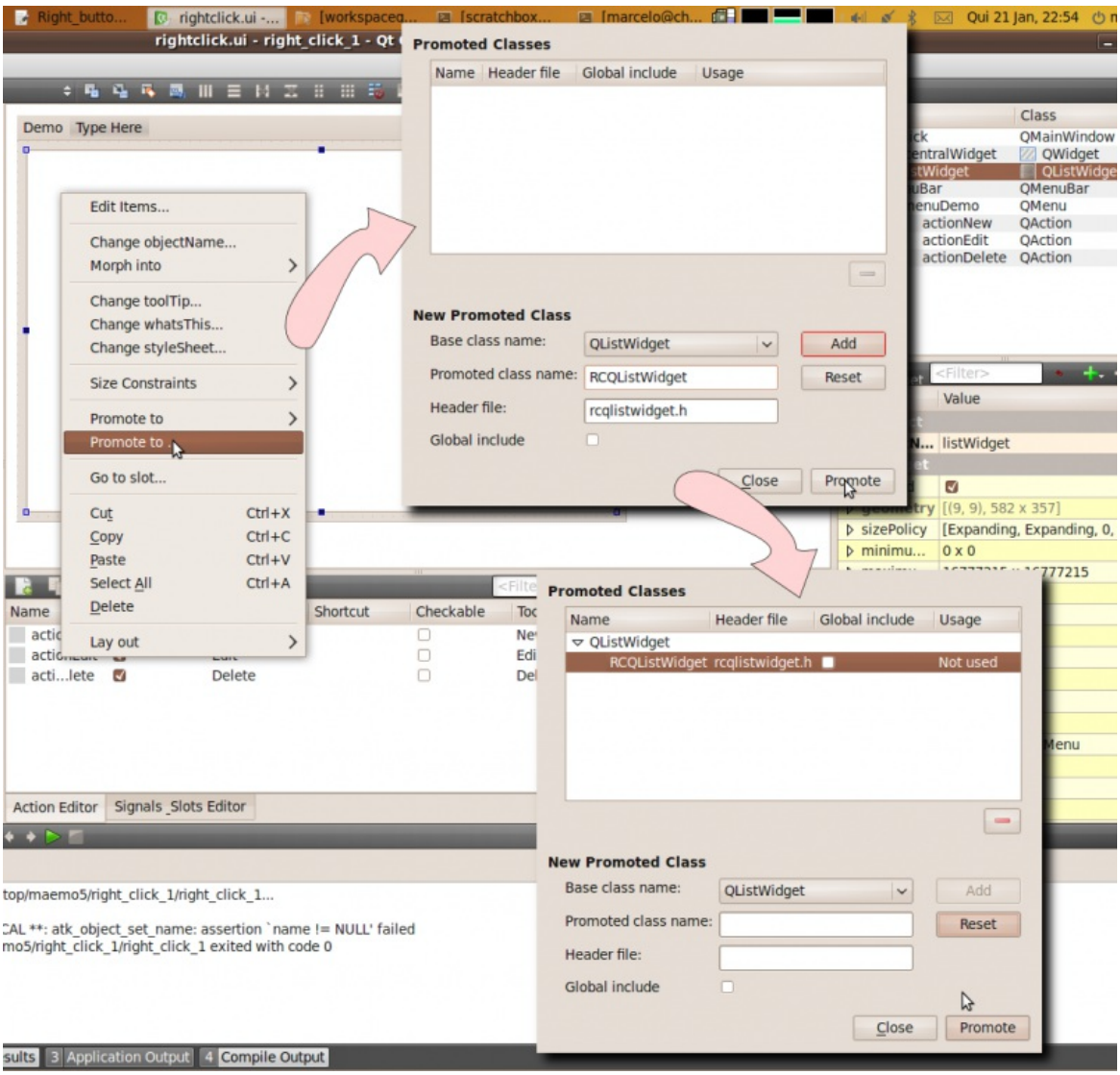




Using Qt Creator UI designer

(C) Copyright Nokia 2013. All rights reserved.

If you want to design your interface using Qt Creator you need to change the base class for the desired components. Click over the component and select "Promote to", type the base class name and press "Add", as indicated in the next picture.



Right click by generic event handler

If you do not want to sub-class the object, it is possible to handle the right button creating a new **QObject** subclass just for this purpose. The method **eventFilter** must be coded for this class. In this example a signal will be used to dispatch the event to the main program. See **rceventhandler.h** and **rceventhandler.cpp** implementations below.

rceventhandler.h

```
#ifndef RCEVENTHANDLER_H
#define RCEVENTHANDLER_H

#include <QObject>
#include <QPoint>

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

protected:
    bool eventFilter(QObject *obj, QEvent *event);

signals:
    void send_rightButtonClicked(const QPoint &p);

};

#endif // RCEVENTHANDLER_H
```

rceventhandler.cpp

```
#include "rceventhandler.h"
#include <QContextMenuEvent>
#include <QMouseEvent>

bool RCEventHandler::eventFilter(QObject *obj, QEvent *event)
{
    // you may handle multiple objects checking "obj" parameter
    if (event->type() == QEvent::ContextMenu) {
```

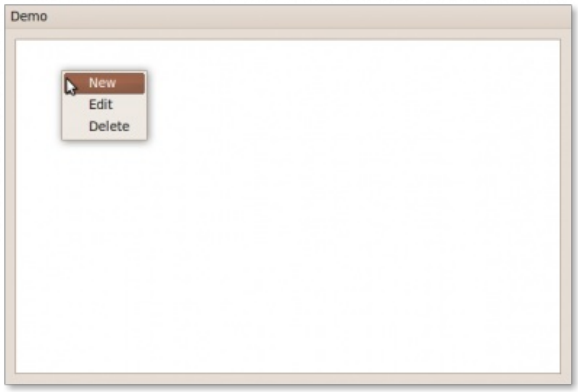

(C) Copyright Nokia 2013. All rights reserved.

```
        QMouseEvent *mouseEvent = static_cast<QMouseEvent*> (event);
        send_rightButtonClicked(mouseEvent->globalPos());
        return true;
    }
    else
        return QObject::eventFilter(obj, event);
}
```

In your main application, create an object from this class and use it as a parameter when calling **installEventFilter** method, as in the next code snippet. In this example, only the list widget was used but it is possible to add more object to the event handler.

```
RightClick::RightClick(QWidget *parent) : QMainWindow(parent), ui(new Ui::RightClick)
{
    ui->setupUi(this);
    // create the event handler object
    RCEventHandler *listEvHandler = new RCEventHandler(this);
    // connect signals
    connect(listEvHandler, SIGNAL(send_rightButtonClicked(const QPoint&)),
            this, SLOT(rightButtonClicked(const QPoint&)));
    // install event filter
    ui->listWidget->installEventFilter(listEvHandler);
}
```

The result is the same, as you can see in the next screenshot taken from Linux desktop.



Source code

The two strategies were tested using the next two examples. It was used Qt Creator 4.6.1.

- First strategy.
- Second strategy.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.