Archived:Creating a custom custom line editor from QLineEdit

Archived: This article is **archived** because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

Qt Quick should be used for all UI development on mobile devices. The approach described in this article (based on C++ for the Qt UI) deprecated.

This example shows how to derive our own Line Widget from QLineEdit. QLineEdit is a one-line text editor. A line edit allows the user to enter and edit a single line of plain text with a useful collection of editing functions, including undo and redo, cut and paste, and drag and drop.

Implementation

We will derive a class named `QCustomEditLine`

```
class QCustomEditLine : public QLineEdit

QCustomEditLine::QCustomEditLine(QWidget *parent) : QLineEdit(parent)
```

Add a function to take the default text once we get it we will refresh the control to re-draw.

```
emptyMessage = msg;
drawEmptyMsg = text().isEmpty();
update();
```

Derive `paintEvent( QPaintEvent *ev )` to draw the text that we want to show when there is no text is entered.

```
QPainter p(this);
QFont f = font();
f.setItalic(true);
p.setFont(f);

QColor color(palette().color(foregroundRole()));
color.setAlphaF(0.5);
p.setPen(color);

QStyleOptionFrame opt;
initStyleOption(&opt);
QRect cr = style()->subElementRect(QStyle::SE_LineEditContents, &opt, this);
cr.setLeft(cr.left() + 2);
cr.setRight(cr.right() - 2);

p.drawText(cr, Qt::AlignLeft|Qt::AlignVCenter, emptyMessage);
```

control the drawing to show the default text when control is focused out and blank it when cointrol focused in.

Now we are ready to add to our widget. To use do the following:

```
m_lineout = new QCustomEditLine(this);
m_lineout->setEmptyMessage("Enter you Email Address");
```


(C) Copyright Nokia 2014. All rights reserved.

qcustomeditline.h

```
#ifndef QCUSTOMEDITLINE_H
#define QCUSTOMEDITLINE_H

#include <QtGui>

class QCustomEditLine : public QLineEdit
{
    Q_OBJECT
public:
    QCustomEditLine(QWidget *parent = 0);
    virtual ~QCustomEditLine();

public:
    void setEmptyMessage( const QString &msg );

protected:
    void paintEvent( QPaintEvent *ev );
    void focusInEvent( QFocusEvent *ev );
    void focusOutEvent( QFocusEvent *ev );

private:
    QString emptyMessage;
    bool drawEmptyMsg;
};

#endif // QCUSTOMEDITLINE_H
```

qcustomeditline.cpp

```
#include "qcustomeditline.h"

QCustomEditLine::QCustomEditLine(QWidget *parent)
                : QLineEdit( parent ){



}

QCustomEditLine::~QCustomEditLine() {

}

void QCustomEditLine::setEmptyMessage( const QString &msg ) {
    emptyMessage = msg;
    drawEmptyMsg = text().isEmpty();
    update();
}

void QCustomEditLine::paintEvent( QPaintEvent *ev ) {
    QLineEdit::paintEvent( ev );
```


(C) Copyright Nokia 2014. All rights reserved.

```
    if ( text().isEmpty() ) {
        QPainter p(this);
        QFont f = font();
        f.setItalic(true);
        p.setFont(f);

        QColor color(palette().color(foregroundRole()));
        color.setAlphaF(0.5);
        p.setPen(color);

        QStyleOptionFrame opt;
        initStyleOption(&opt);
        QRect cr = style()->subElementRect(QStyle::SE_LineEditContents, &opt, this);
        cr.setLeft(cr.left() + 2);
        cr.setRight(cr.right() - 2);

        p.drawText(cr, Qt::AlignLeft|Qt::AlignVCenter, emptyMessage);
    }
}

void QCustomEditLine::focusInEvent( QFocusEvent *ev ) {
    if ( drawEmptyMsg ) {
        drawEmptyMsg = false;
        update();
    }
    QLineEdit::focusInEvent( ev );
}

void QCustomEditLine::focusOutEvent( QFocusEvent *ev ) {
    if ( text().isEmpty() ) {
        drawEmptyMsg = true;
        update();
    }
    QLineEdit::focusOutEvent( ev );
}
```

Reference

(C) Copyright Nokia 2014. All rights reserved.

- QLineEdit

(C) Copyright Nokia 2014. All rights reserved.