Archived:Handling softkeys in Qt for Symbian

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 QWidget) is deprecated.

Overview

Softkeys will be visible in the control pane of a Qt application, if a focused widget has actions (QAction) with a softkey role defined. This article discusses how softkeys are implemented and how to dynamically change the softkey actions in Qt on Symbian.

Detailed description

`QAction::setSoftKeyRole()` can be used for specifying the type (role) of a softkey for an action:

QAction::NoSoftKey	Action is not used as a softkey
QAction::PositiveSoftKey	Softkey with a positive or non-destructive role such as Ok, Select, or Options.
QAction::NegativeSoftKey	Softkey with a negative or destructive role role such as Cancel, Discard, or Close.
QAction::SelectSoftKey	Softkey that selects a particular item or widget in the application.

When the application is in portrait mode, locations for positive and negative softkeys are at the left-hand and right-hand side of the screen, respectively. In landscape mode, the positive softkey is at the bottom and the negative softkey on top.

Softkeys will be set according to the actions of the currently focused widget. By default, the top-level widget has *Options* and *Exit* softkeys. If a focused child widget has no defined softkeys, the framework will traverse up the widget parent hierarchy looking for a widget containing softkey actions.

Traversal through the widget hierarchy stops once at least one softkey action is found. This has some implications for widgets that want to modify a single softkey; for example, consider the following code used for defining a *Back* softkey in a child widget:

```
QAction* backAction = new QAction( tr("Back"), this );
backAction->setSoftKeyRole( QAction::NegativeSoftKey );
connect(backAction, SIGNAL(triggered()), this, SLOT(back()));
addAction( backAction );
```

If the application has other actions located in the *Options* menu, accessing the menu is not possible as the *Options* softkey is not visible as the focused widget has defined only a single (negative) softkey.

Solution

Managing the softkeys should be done in the implementation of a top-level parent widget (`QWidget`) or the main window (`QMainWindow`).

For example, to toggle between *Exit* and *Back* actions as the negative softkey, use the following code:

```
MainWindow::MainWindow(QWidget *parent) :
  QMainWindow(parent)
{
...
m_backAction = new QAction( tr("Back"), this );
m_backAction->setSoftKeyRole( QAction::NegativeSoftKey );
}
```


(C) Copyright Nokia 2014. All rights reserved.

```
void MainWindow::toggleRightSoftKey( bool back, QWidget* focusWidget )
{
  if( back ) {
    connect(m_backAction, SIGNAL(triggered()), focusWidget, SLOT(back()));
    addAction( m_backAction );
  } else {
    disconnect( m_backAction );
    removeAction( m_backAction );
  }
}
```

After this, a child widget can replace the default *Exit* right softkey with *Back* by calling `toggleRightSoftKey(true, this)` on the `MainWindow` object. Note that the code assumes the widget has a `back()` slot implemented.

(C) Copyright Nokia 2014. All rights reserved.