Symbian C++ and Qt toolbar comparison

Overview

This article compares toolbar support in S60 platform and Qt.

Comparison

S60 supports fixed (for touch UI) and floating toolbars. Toolbar items can be added from the resource file or dynamically. Toolbars can be application specific or view specific.

Qt also supports fixed and floating toolbars. Actions and widgets can be added to the toolbar. A toolbar can also be an independent window.

Operations	S60	Qt
Class	CAknToolBar	QToolBar
Header and library	AknToolbar.h, eikcoctl.lib	QToolBar, QtGui module.
Creating toolbar	Use CAknToolBar::NewL(const Tint aResourceId), Or use CAknAppUi::CurrentPopupToolbar() or CAknAppUi::CurrentFixedToolbar() to get a pointer to the current fixed/pop-up toolbar.	Using QToolBar ::QToolBar(QWidget *parent = 0) or QToolBar :: QToolBar(const QString &title, QWidget *parent = 0)
Adding item (dynamically)	CAknToolBar::AddItemL(). Leaves with value KErrNotSupported if the control type is not supported by the toolbar.	QToolBar::addAction(). If the button is not appropriate, you can add a widget using addWidget().
Decorating toolbar items	Icon or tooltip can be set for the CCoeControl being added to the toolbar.	Icon or tooltip can be set for the QAction being added to the toolbar.
Item event handling	Via observer class MAknToolbarObserver. OfferToolbarEventL (TInt aCommand) can be used to handle toolbar events. The observer is set using SetToolbarObserver (MAknToolbarObserver *aObserver)	Via QtoolBar::actionTriggered (QAction * action) signal. This signal can be connected to an appropriate slot to receive menu selection notifications.
Orientation setting	Using SetOrientation( ) function. Orientation can be vertical(EAknOrientationVertical) or horizontal(EAknOrientationHorizontal).	Using SetOrientation ( ) function. Orientation can be vertical (Qt::Vertical) or horizontal (Qt::Horizontal).
Enabling/disabling toolbar items dynamically	Using HideItem() inside MAknToolbarObserver:: DynInitToolbarL( ).	Using QAction::setenabled(bool).
Removing an item from the toolbar	Using RemoveItem (const TInt aCommandId).	Using removeAction(QAction*).
Disabling/enabling the toolbar	Using DisableToolbarL (TBool aDisable).	Using QWidget::setEnabled(bool).
Setting the tool button styles	Different ccoecontrols can be used for different styles.	Apart from setting style for the added widgets, setToolButtonStyle (Qt::ToolButtonStyle toolButtonStyle) can also be used. Default mode is icon only.

(C) Copyright Nokia 2014. All rights reserved.

Toolbar types	Supports fixed (for touch UI) and floating (mainly for non-touch UI) toolbars. KAknToolbarFixed flag is for fixed toolbars. KAknToolbarWithoutCba flag is for floating toolbars.	Supports fixed and floating type through floating property. QtoolBar::setFloatable (bool floatable) can be used to make the toolbar a drag and drop window.

Sample Qt code for toolbar support

Toolbat creation

```
QToolbarEx::QToolbarEx( QWidget *parent )
: QWidget( parent )
{
  toolbar = new QToolBar( this );

  // Create actions with icon and text, add them to the toolbar
  QAction* action1 =  new QAction( QIcon("c:\\data\\610x.bmp"), "item1", this );
  toolbar->addAction( action1 );

  QAction* action2 =  new QAction( QIcon("c:\\data\\Nokia-logo.bmp"), "item2", this );
  toolbar->addAction( action2 );

  QAction* action3 =  new QAction( QIcon("c:\\data\\5.bmp"), "item3", this );
  toolbar->addAction(action3);

  // Connect the actionTriggered signal to handle toolbar events
  connect( toolbar, SIGNAL(actionTriggered ( QAction *)),
           this, SLOT( menuClicked( QAction* ) ) );

  toolbar->setIconSize( QSize( 100, 50 ) );
  toolbar->setToolButtonStyle( Qt::ToolButtonTextUnderIcon );
  // Set the geometry, the default position is at the top the screen
  toolbar->setGeometry( QRect( 0, 400, 400, 100 ) );
  setGeometry( QRect( 0, 0, 400, 600 ) );
}
```

Handling toolbar actions

```
void QToolbarEx::menuClicked( QAction* action )
{
  // action is the clicked menu item, displaying the text associated with it
  QMessageBox::information ( this, action->text(), action->text() );
}
```

Enabling/disabling a toolbar button

```
// Listing all the toolbar actions in the toolbar
QList<QAction*> actionList( toolbar->actions() );

// Use your own logic to select the action to be enabled/disabled
```


(C) Copyright Nokia 2014. All rights reserved.

```
// Checking if the action is already enabled
if( !actionList[0]->isEnabled() ) {
  //Enabling the action
  actionList[0]->setEnabled( true );
}

//Checking if the action is already disabled
if( actionList[0]->isEnabled() ) {
  //Disabling the action
  actionList[1]->setEnabled( false );
}
```

Removing a tool button

```
// Listing all the toolbar actions in the toolbar
QList<QAction*> actionList( toolbar->actions() );

//Use your own logic to select the action to be removed
toolbar->removeAction( actionList[2] );
```

Sample application

File:QToolbarEx.zip

(C) Copyright Nokia 2014. All rights reserved.