Archived:Create a page based UI with QStackedWidget and QToolbar

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.

This example shows how to create a page based user interface, this interface style is common in many applications of devices like the iPhone. This example doesn't shows how to add animated transitions to the page changes. See the Extending QStackedWidget for sliding page animations in Qt article if you are interested in add an sliding animated transition between the page changes.

Source Code

This section shows and describe the code of the example. A complete code example can be downloaded from here.

The Page class

The page class represents a page of the application. It's only a simple example of how a page could be. In this particular example the page only contains one label for the title, one label to show an image an one label for the contents of the page, but since the Page class inherits `QWidget` it can be as complex as any other widget or window.

Page Class Definition

```
#ifndef PAGE_H
#define PAGE_H

#include <QWidget>
#include <QString>

class QLabel;

class Page : public QWidget
{
    Q_OBJECT
public:
    explicit Page(const QString &title, const QString &content, const QString &imagePath, QWidget *parent = 0);

signals:

public slots:

private:
    QLabel *titleLabel;
    QLabel *contentLabel;
    QLabel *imageLabel;
};

#endif // PAGE_H
```

Page Class Implementation

```
#include "page.h"

#include <QLabel>
#include <QPixmap>
#include <QVBoxLayout>

Page::Page(const QString &title, const QString &content, const QString &imagePath, QWidget *parent) :
    QWidget(parent)
{
    //Layout for organizing the widgets of the page
    QVBoxLayout *mainLayout = new QVBoxLayout(this);
    //Labels with the content of the page
    titleLabel = new QLabel(title, this);
    contentLabel = new QLabel(content, this);
    imageLabel = new QLabel(this);

    //Set a bigger font for the title
    titleLabel->setFont(QFont("Sans-Serif", 15));
    //Set the pixmap for the image
    imageLabel->setPixmap(QPixmap(imagePath));
    //Tell the content label to expand vertically and horizontally to all the space available
    //Useful for device rotation
    contentLabel->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Expanding);

    //Add the content widgets to the main layout
    mainLayout->addWidget(titleLabel);
    mainLayout->addWidget(imageLabel);
    mainLayout->addWidget(contentLabel);
}
```


(C) Copyright Nokia 2013. All rights reserved.

MainWindow Class Definition

The **MainWindow** class inherits `QMainWindow` it's composed by:

- `QStackedWidget`: This is the central widget of the **MainWindow** class it can contain widgets of the Page class or any other `QWidget` subclass
- `QToolbar`: It's positioned at the bottom of the screen and contains the widgets to change between pages.

This class uses a `QSignalMapper` to map the value of a widget in the toolbar with a page on the central widget.

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QtGui/QMainWindow>

class Page;

class QStackedWidget;
class QToolButton;

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = 0);
    ~MainWindow();

private:
    QStackedWidget *pagesWidget;
    QToolButton *homeButton;
    QToolButton *page1Button;
    QToolButton *page2Button;
    Page *homePage;
    Page *page1;
    Page *page2;
};

#endif // MAINWINDOW_H
```

MainWinidow Class Implementation

```
#include "mainwindow.h"
#include "page.h"

#include <QHBoxLayout>
#include <QToolButton>
#include <QSignalMapper>
#include <QStackedWidget>
#include <QToolBar>
#include <QVBoxLayout>

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
{
    //Create toolbar
    QToolBar *toolBar = new QToolBar(this);
    //Create a QMapper to map the values of the tool buttons with the indexes of the stacked widget
    QSignalMapper *mapper = new QSignalMapper(this);

    //Create QStackedWidget used as central widget
    pagesWidget = new QStackedWidget(this);
    //Create QToolButton used to change between "pages" of the central widget
    homeButton = new QToolButton(this);
    page1Button = new QToolButton(this);
    page2Button = new QToolButton(this);

    //Create widgets of the custom class Page which would be used as pages
    homePage = new Page(tr("Home Page"), tr("Welcome, this is the home page!"), "home.png", this);
    page1 = new Page(tr("Page 1"), tr("This is the page 1"), "1.png", this);
    page2 = new Page(tr("Page 2"), tr("This is the page 2"), "2.png", this);

    //Set a descriptive icon for the toolbuttons
    homeButton->setIcon(QIcon("home.png"));
    page1Button->setIcon(QIcon("1.png"));
    page2Button->setIcon(QIcon("2.png"));

    //Add a toolbar at the bottom of the screen and set fixed in that position
    addToolBar(Qt::BottomToolBarArea, toolBar);
    toolBar->setMovable(false);

    //Add pages to the central widget
    pagesWidget->addWidget(homePage);
    pagesWidget->addWidget(page1);
    pagesWidget->addWidget(page2);

    //Add buttons to the toolbar
    toolBar->addWidget(homeButton);
    toolBar->addWidget(page1Button);
    toolBar->addWidget(page2Button);

    //Tell the toolbuttons that expand horizontally to all the available space in the screen
    //Useful for screen rotation
    homeButton->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Preferred);
    page1Button->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Preferred);
    page2Button->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Preferred);

    //Connect the clicked signal with the QSignalMapper
    connect(homeButton, SIGNAL(clicked()), mapper, SLOT(map()));
```

 (C) Copyright Nokia 2013. All rights reserved.

```
    connect(page1Button, SIGNAL(clicked()), mapper, SLOT(map()));
    connect(page2Button, SIGNAL(clicked()), mapper, SLOT(map()));

    //Set a value for each button
    mapper->setMapping(homeButton, 0);
    mapper->setMapping(page1Button, 1);
    mapper->setMapping(page2Button, 2);

    //Connect the mapper to the central widget
    //The mapper will set a value to each button and set that value to the central widget
    connect(mapper, SIGNAL(mapped(int)), pagesWidget, SLOT(setCurrentIndex(int)));

    //Set the stacked widget as the central widget of the main window
    setCentralWidget(pagesWidget);
}

MainWindow::~MainWindow()
{
}
```

Main file

The main file is just the one that `QtCreator` automatically generates, it simply creates a `QApplication` and an object of the `MainWindow` class, shows the **MainWindow** and starts the application.

main.cpp

```
#include <QtGui/QApplication>
#include "mainwindow.h"

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
#if defined(Q_WS_S60)
    w.showMaximized();
#else
    w.show();
#endif

    return a.exec();
}
```

Screenshots

 (C) Copyright Nokia 2013. All rights reserved.

 (C) Copyright Nokia 2013. All rights reserved.