Archived:Getting started with Carbide.c++ and Qt

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 Creator is highly recommended (and Carbide.c++ deprecated) for Qt development. In addition, user interfaces for mobile devices should be created using Qt Quick rather than QWidget) as shown.

Introduction

04 Jan 2009

This article is for experienced developers who want to start development in Qt with Carbide.c++.

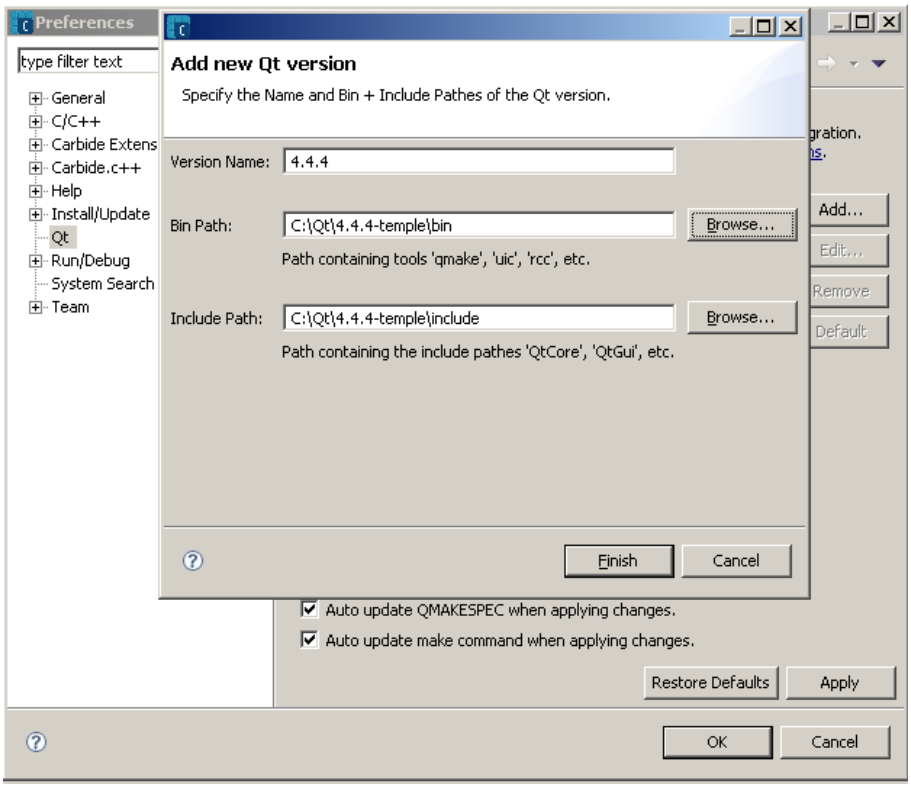
Before following the steps given here please make sure that Carbide.c++, S60 SDK and Qt are properly installed.

Prerequisite

- Download and install the Qt

- Carbide.c++ 2.0 does not automatically locate your Qt installation, so you need to tell it where Qt is installed. This is done through the following settings:

```
Carbide.c++ preferences: Window > Preferences > Qt > Add...
```

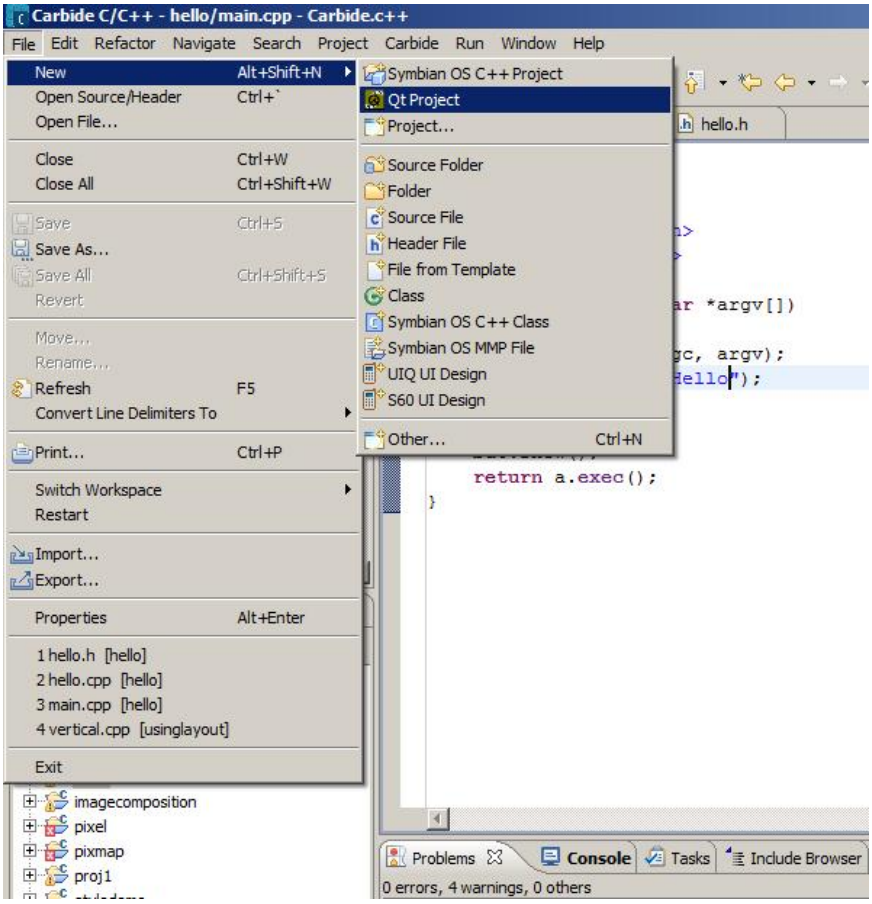


In the above screen you will need to use the directory path for the current version e.g. **c:\Qt\4.6.0\bin and c:\Qt\4.6.0\include**

First Step

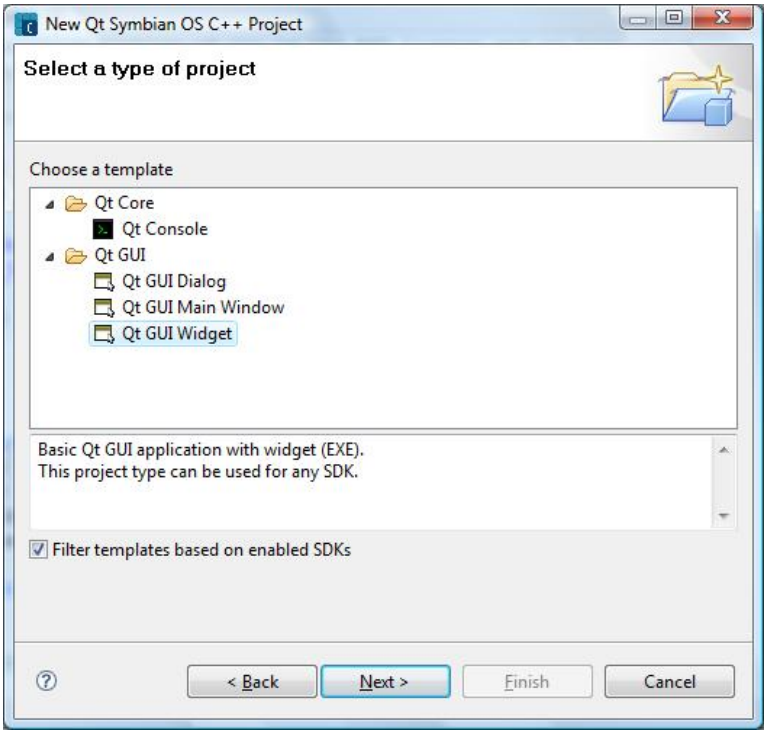
Open the Carbide IDE and click File > New > Qt Project.

(C) Copyright Nokia 2013. All rights reserved.



Second Step

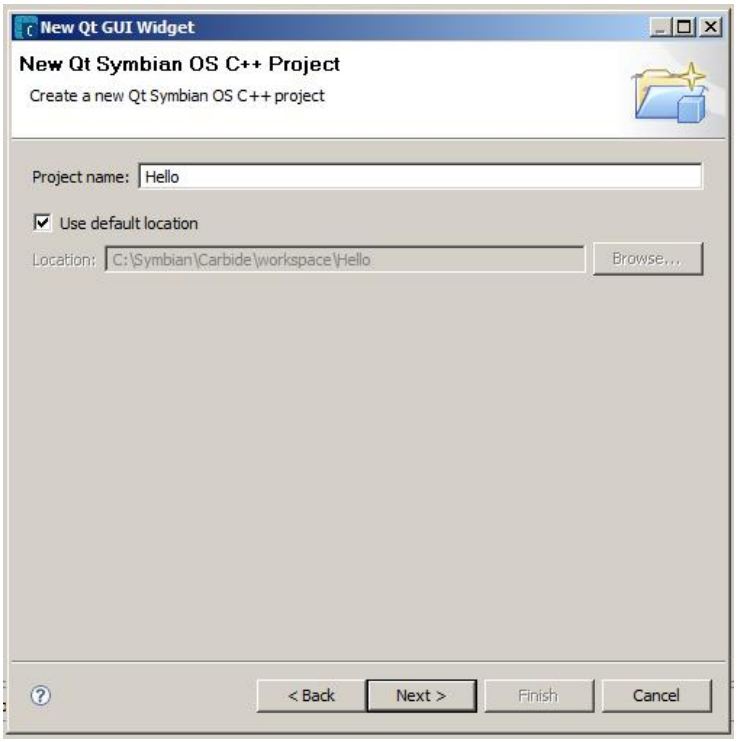
A new window will open on the screen. This will show the application types that can be created. Click Qt GUI Widget and then click Next.



Third Step

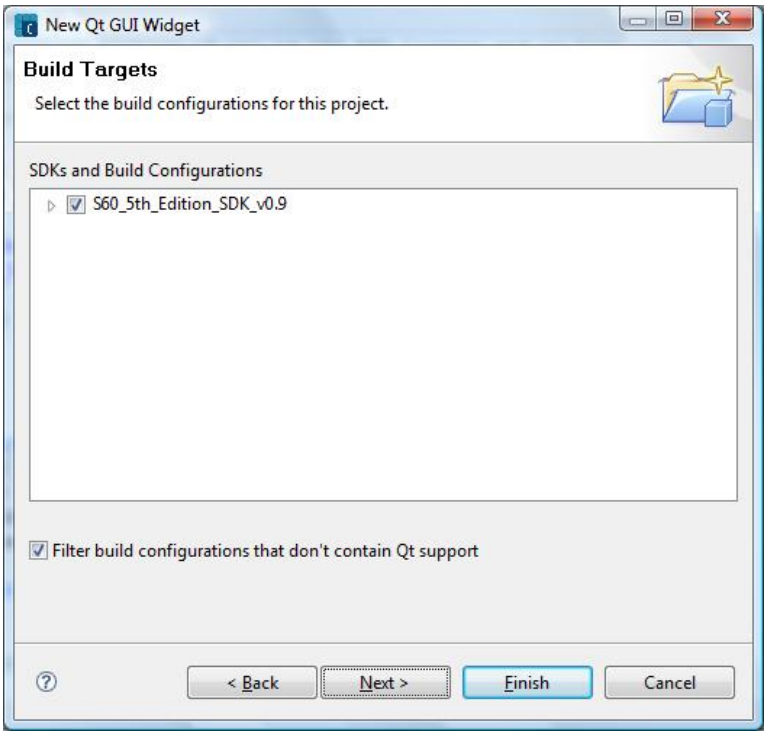
Enter a name for the project and click Next.

(C) Copyright Nokia 2013. All rights reserved.



Fourth Step

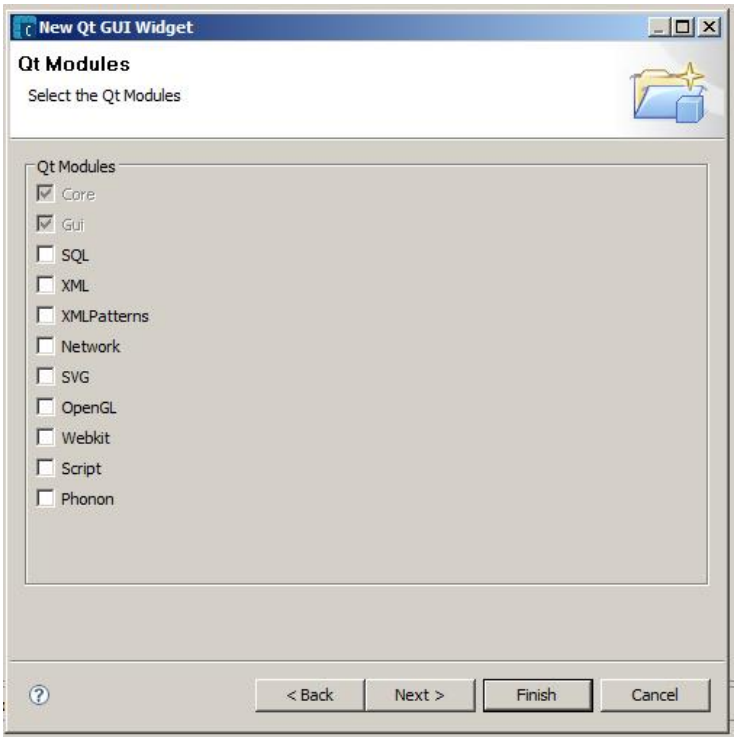
Select the SDK to create the project for and click Next.



Fifth Step

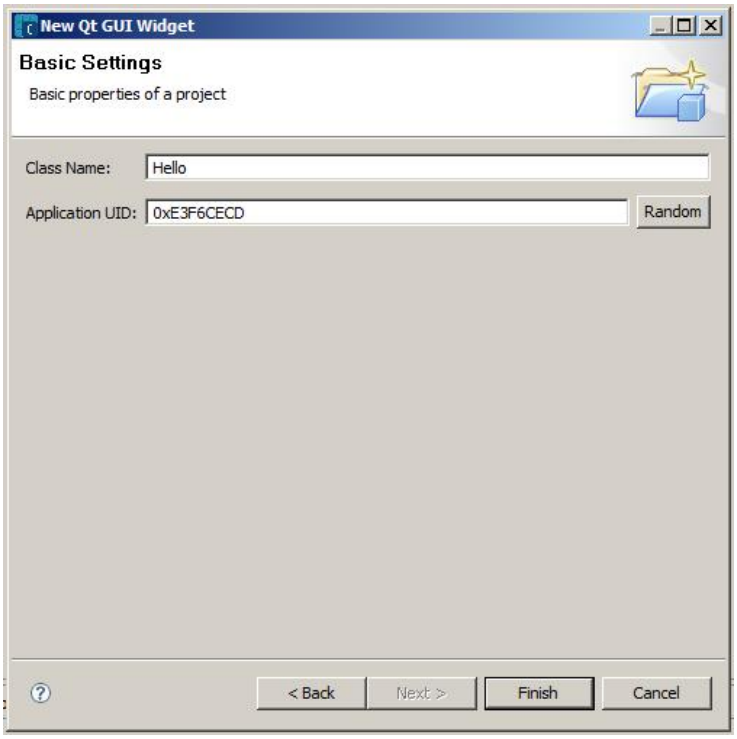
Select the module(s) and click Next. For our example, click Next without selecting any extra modules.

(C) Copyright Nokia 2013. All rights reserved.



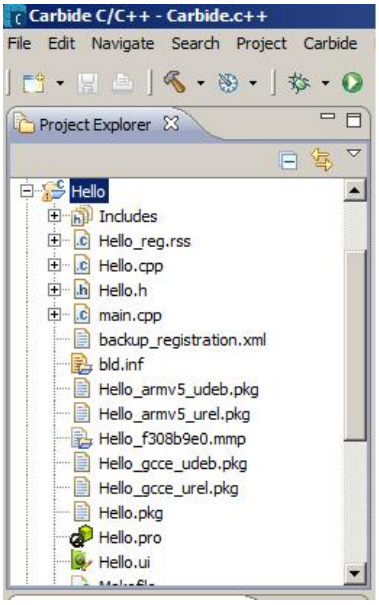
Sixth Step

The class name is required. By default it is the application name. Click Finish to proceed.

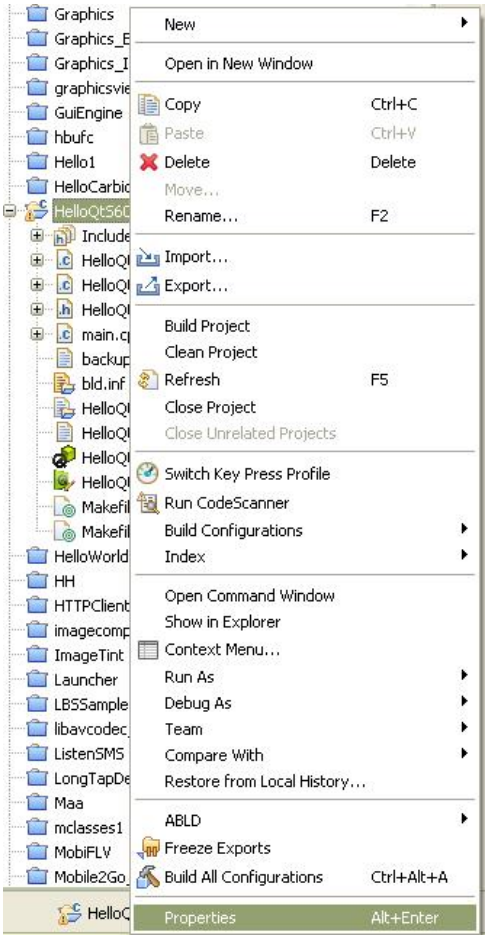


After finishing these steps, the project is ready. Several files are created automatically as shown in the following screenshot.

(C) Copyright Nokia 2013. All rights reserved.



If you have more than one version of Qt installed on your machine, choose the correct version as shown here:

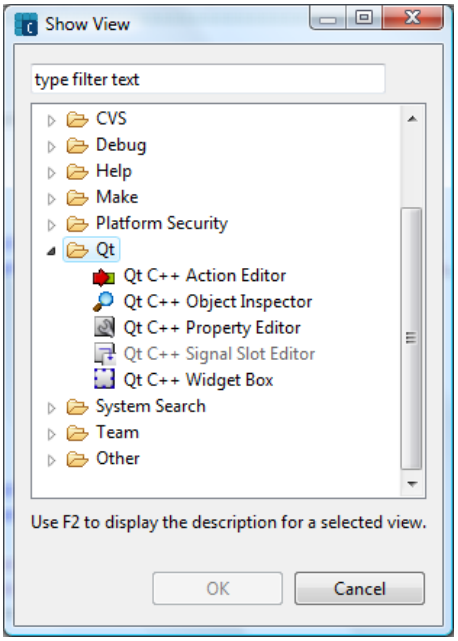


Include Qt Views

- Include views for Qt from:

```
Carbide.c++ > Windows > Show View > Other... > Qt
```


(C) Copyright Nokia 2013. All rights reserved.

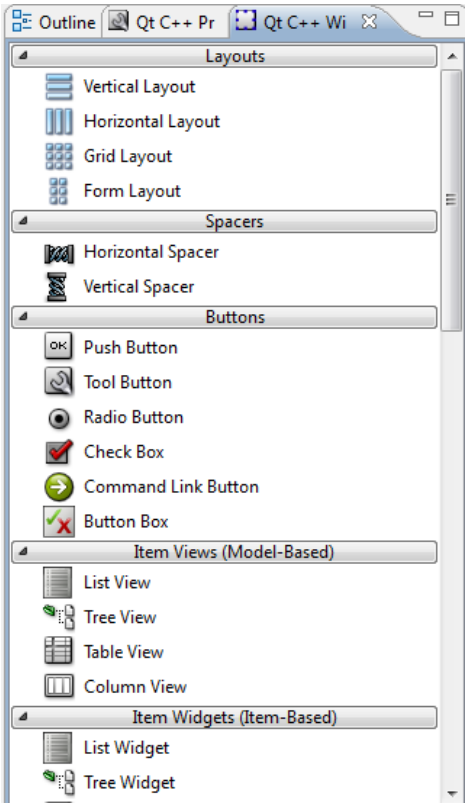


- View selection depends on the application. However, at least the following views must be selected:

1. Qt C++ Widget Box
2. Qt C++ Property Editor
3. Qt C++ Signal Slot Editor

Adding controls

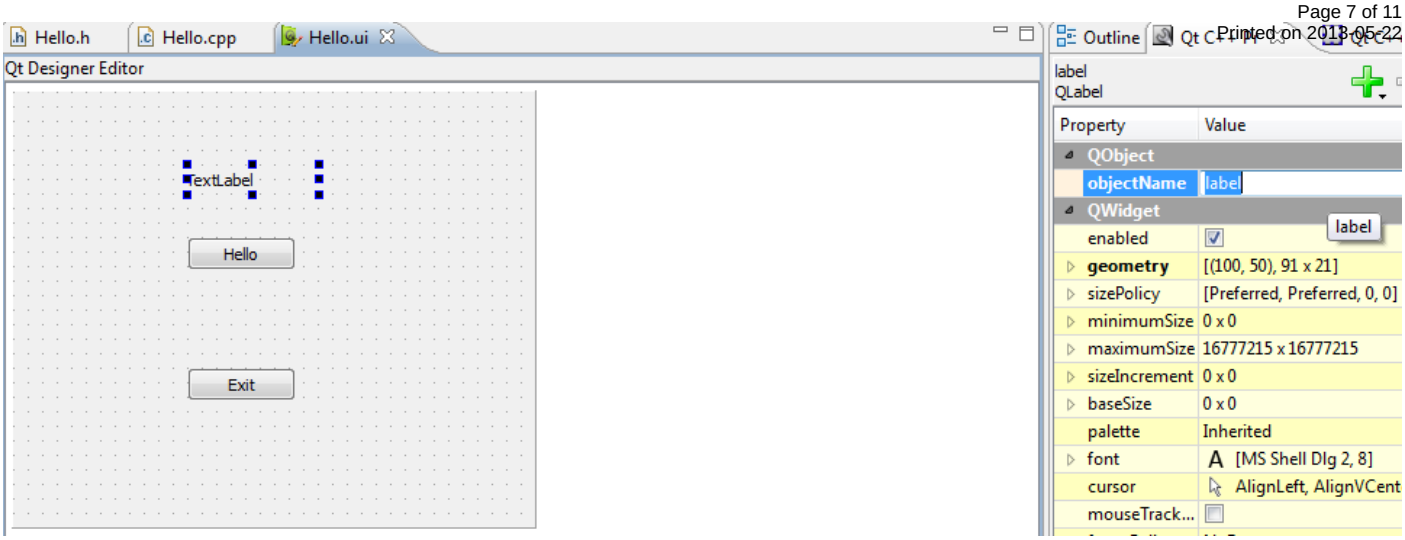
- Open `Hello.ui` and place the control from the **Qt C++ Widget Box**.



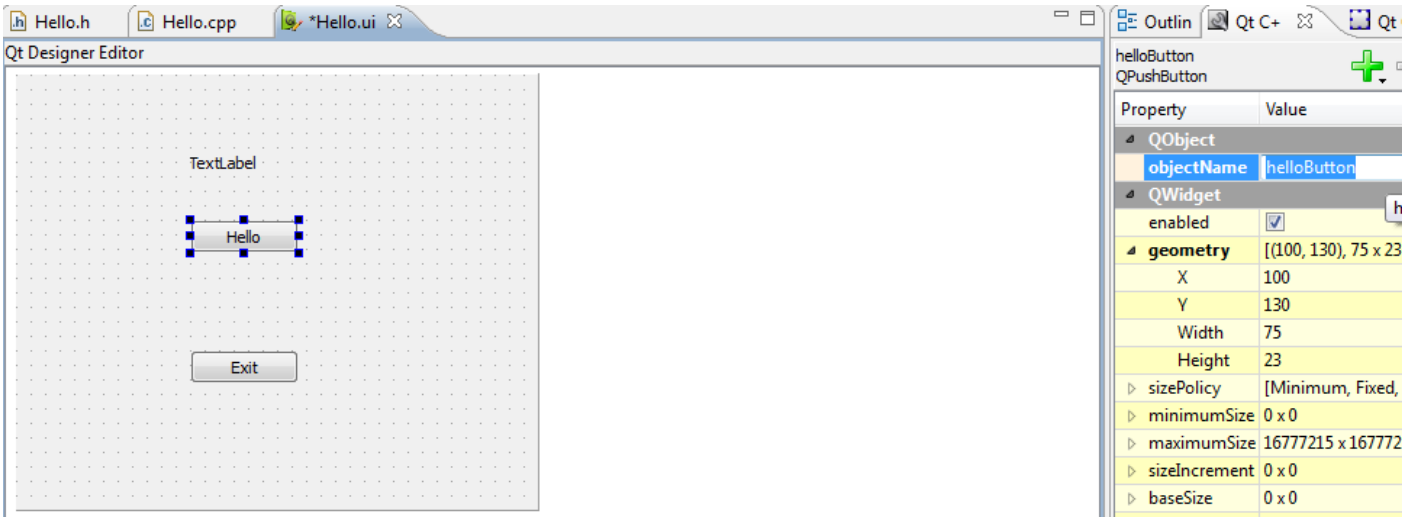
- In this example, two push buttons and one text label will be added.
- The display text of an object can be changed by right clicking on object > Change text...
- The name of an object can be changed as shown in the following screenshots.

Text label

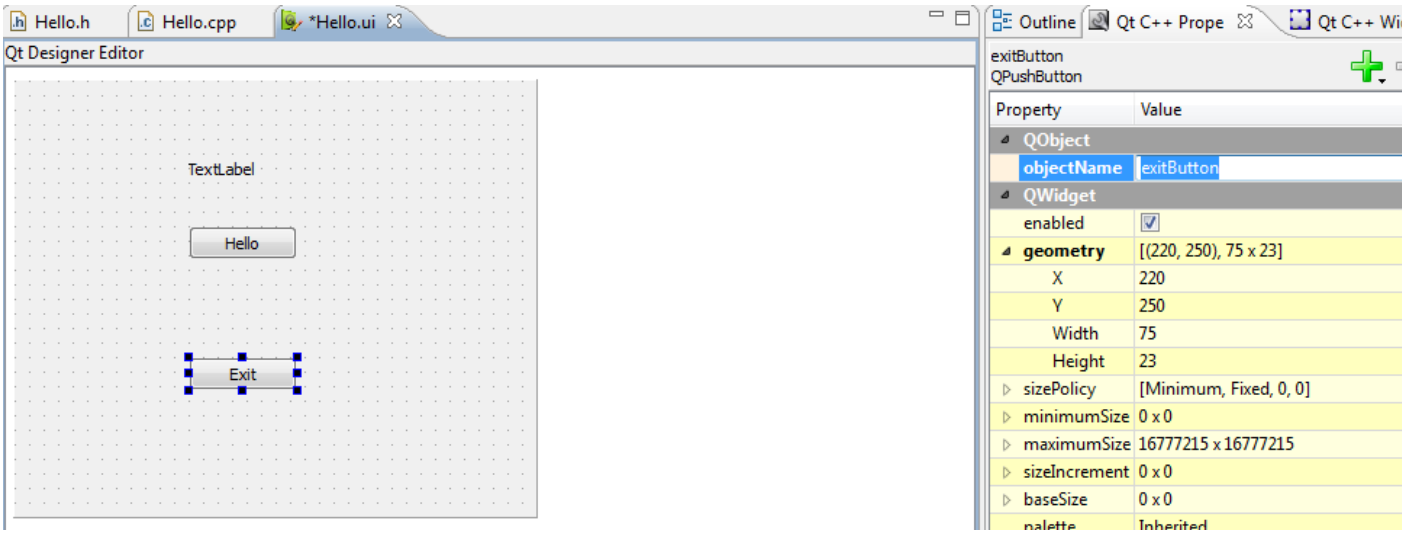
(C) Copyright Nokia 2013. All rights reserved.



Hello push button



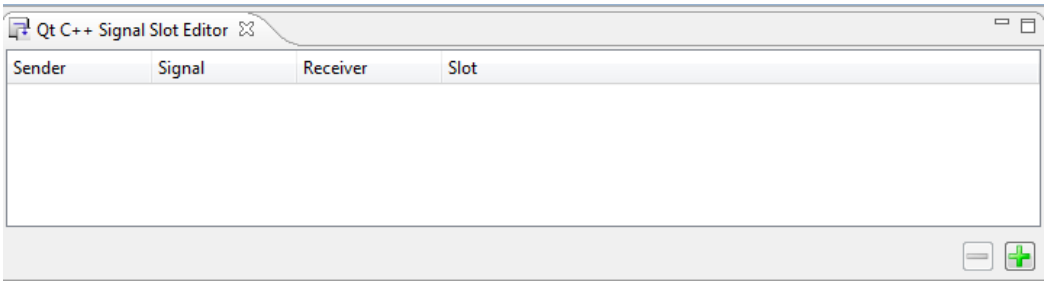
Exit push button



Implementing Signal and Slot mechanism for Exit push button

- Open `Hello.ui`.
- Open the **Qt C++ Signal Slot Editor** view.

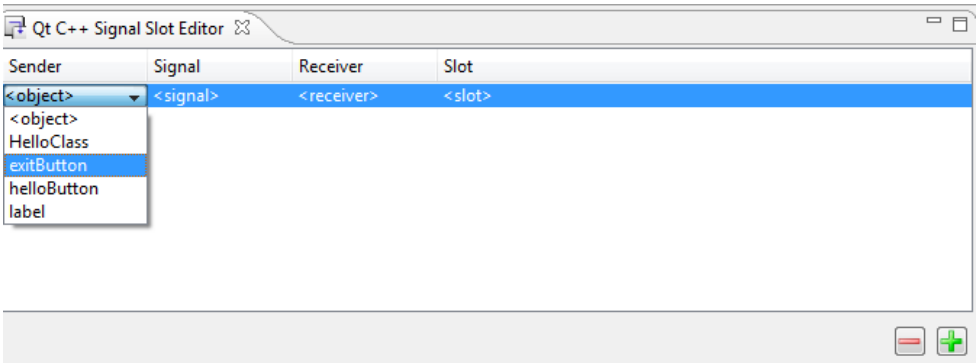
(C) Copyright Nokia 2013. All rights reserved.



- Click to add an entry for handling events on control.
- In this example, we will handle events on the Exit push button
- The application will terminate when the user clicks the Exit push button

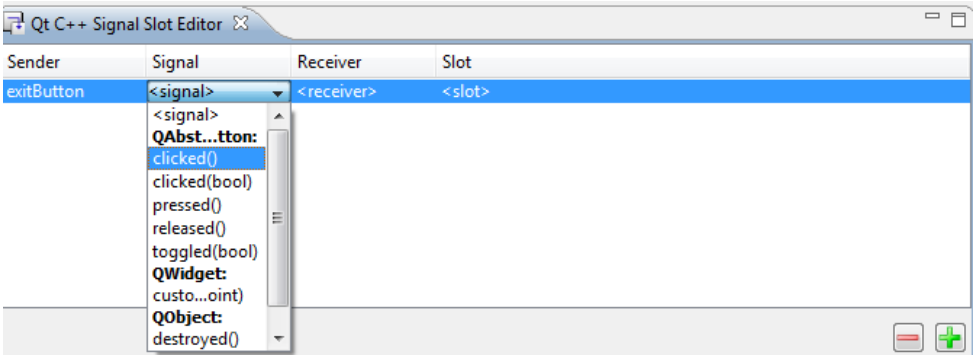
Sender

- exitButton is the sender of the signal.



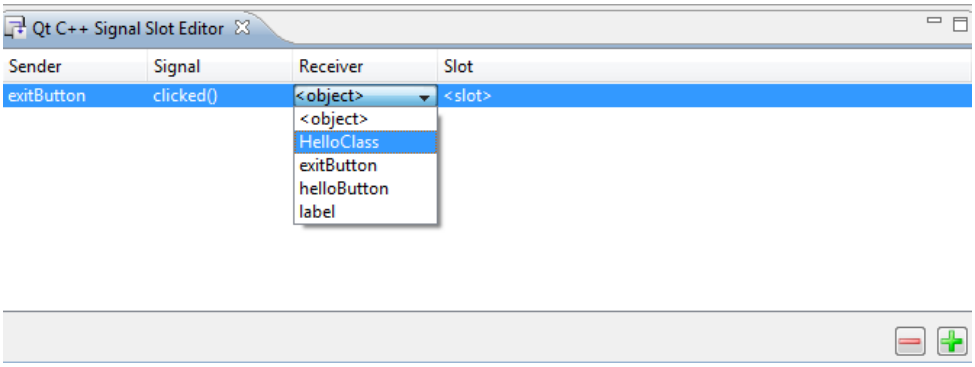
Signal

- Select the signal clicked() to be sent by exitButton.



Receiver

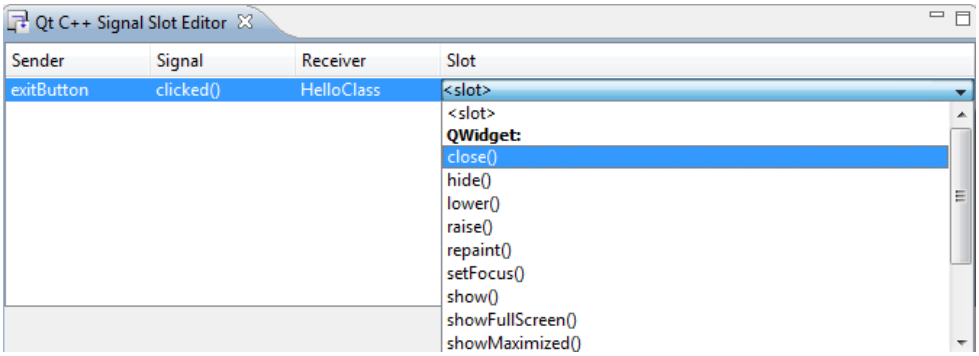
- `HelloClass` is the receiver of this event handling mechanism.



Slot

- Slot is the action performed when exitButton is clicked(). To exit from the application, select close().

(C) Copyright Nokia 2013. All rights reserved.



Implementing the Hello push button

- In this example, we implement the Hello button event by coding in the files Hello.h and Hello.cpp.

Header file

```
#ifndef HELLO_H
#define HELLO_H

#include <QtGui/QWidget>
#include "ui_Hello.h"

class Hello : public QWidget
{
    Q_OBJECT

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

public slots: //We have added this section to handle "clicked()" event on "helloButton"
    void ShowHelloText(); // Fill the label text on clicking "Hello" button

private:
    Ui::HelloClass ui;
};

#endif // HELLO_H
```

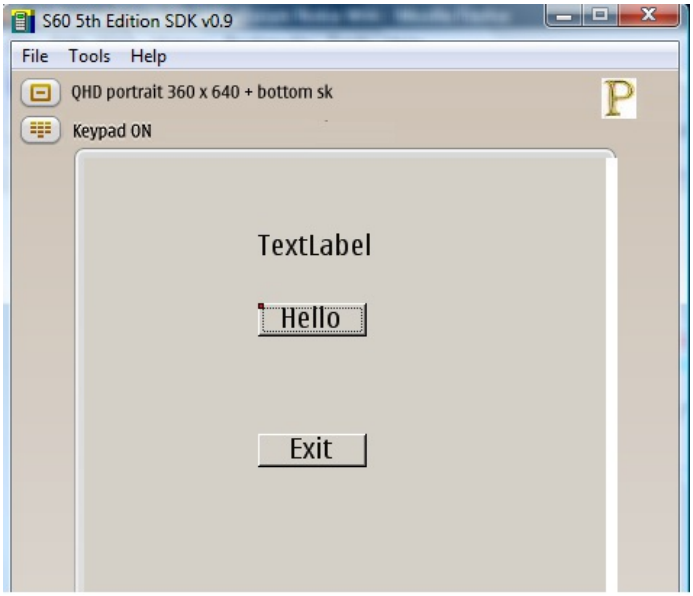
Source file

```
#include "Hello.h"

Hello::Hello(QWidget *parent)
    : QWidget(parent)
{
 ui.setupUi(this);

        // Signal and slot mechanism for "helloButton"
 QObject::connect(ui.helloButton, SIGNAL(clicked()), this, SLOT(ShowHelloText()));
}

Hello::~Hello()
{

}

//We have implemented the following function to display "HelloWorld!!!" text in TextLabel
void Hello::ShowHelloText()
{
 ui.label->setText("HelloWorld!!!");
}
```

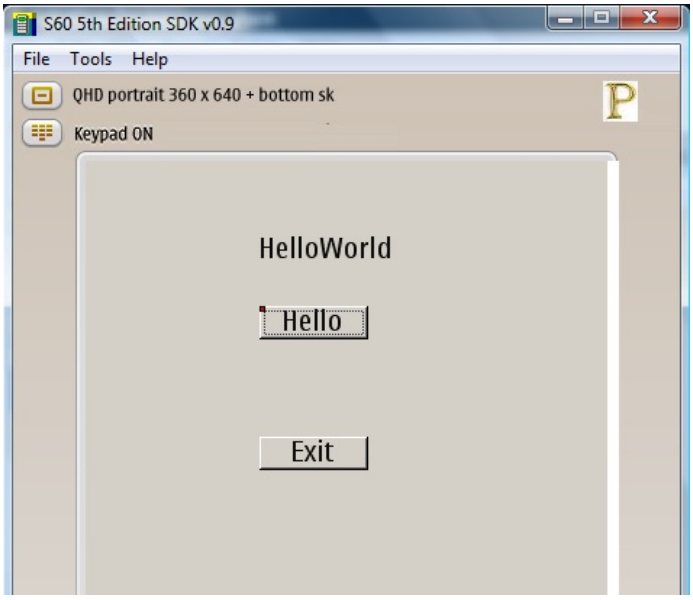
Output

Initial state

(C) Copyright Nokia 2013. All rights reserved.



Hello button pressed state



Example application

- Extract this example application: Hello.zip
- Import the .pro file from the previously extracted application in your Carbide.c++ IDE.

```
Carbide.c++ >> Import >> Qt >> Qt Project >> Hello.pro
```

Feedback

- If you encounter any problems in creating your HelloWorld application in Qt, please enter your feedback in the Comment page of this article.
- Raise any other Qt related questions on the Qt Discussion Boards.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.