Administrator

[Email address]

Abstract

Some note for coding with Qt5 – C++

Qt5 – C++

For practice

# Contents

[Contents 1](#_Toc84171092)

[I. Signal and slot 2](#_Toc84171093)

[II. Multi-windows 3](#_Toc84171094)

[III. Multi-thread 4](#_Toc84171095)

# Signal and slot

2 ways for connecting 2 objects:

* Using connect function:

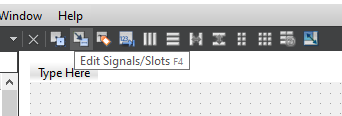
|  |
| --- |
| MainWindow::**MainWindow**(QWidget \*parent)  : QMainWindow(parent)  , ui(new Ui::MainWindow)  {  ui->setupUi(this);  connect(ui->horizontalScrollBar, SIGNAL(valueChanged(int)),  ui->progressBar, SLOT(setValue(int)));  } |

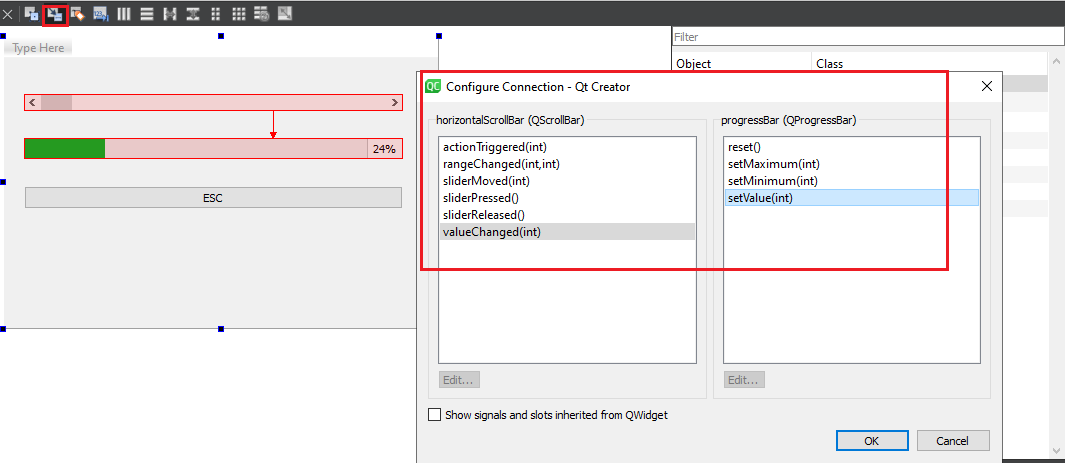
+ Using disconnect function to break that connection:

|  |
| --- |
| disconnect(ui->horizontalScrollBar, SIGNAL(valueChanged(int)),  ui->progressBar, SLOT(setValue(int))); |

* Using Edit Signal/Slots to clicking and dragging, after that select Signal and slot.

We can tick to “Show signals and slots ….” To get more.





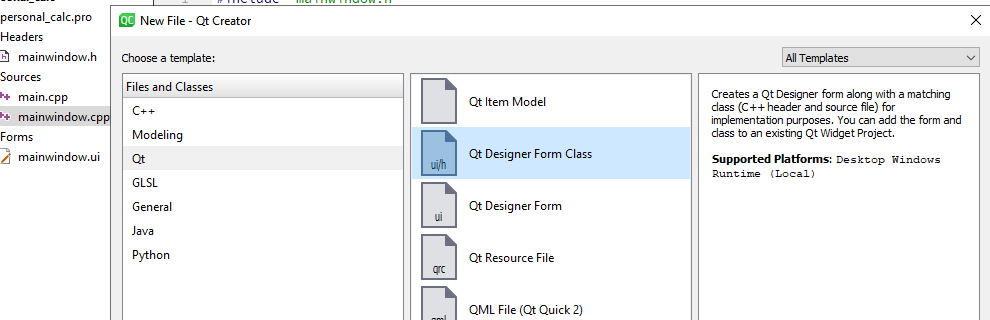
* 1 Signal can control multi-obj

# Multi-windows

* Can set pos/name obj in **mainwindow.cpp** :

|  |
| --- |
| MainWindow::**MainWindow**(QWidget \*parent)  : QMainWindow(parent)  , ui(new Ui::MainWindow) {  ui->setupUi(this);  **ui->bntEsc->setText("Thoat");**  } |

* Create new windows: Forms -> Add new…



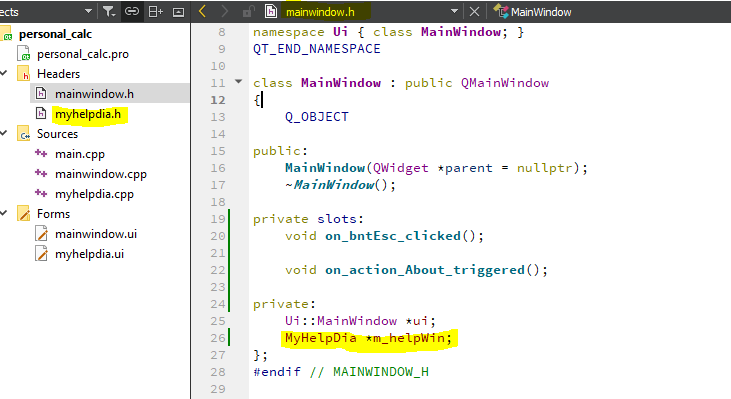
Callback of menu item:

**Method 1:**

|  |
| --- |
| void MainWindow::**on\_action\_About\_triggered**() {  MyHelpDia helpWindows;  helpWindows.setModal(true);  helpWindows.*exec*();  } |

**Method 2:**

Using show method:



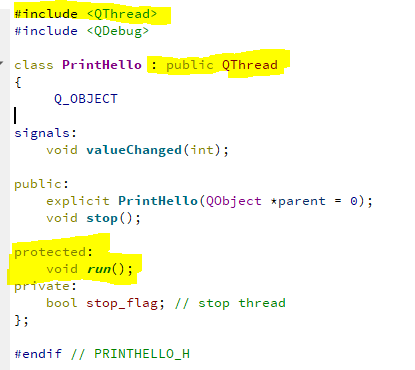
1. Add obj to mainwindow
2. Trigger

|  |
| --- |
| void MainWindow::**on\_action\_About\_triggered**(){  m\_helpWin = new MyHelpDia();  m\_helpWin->show();  } |

# Multi-thread

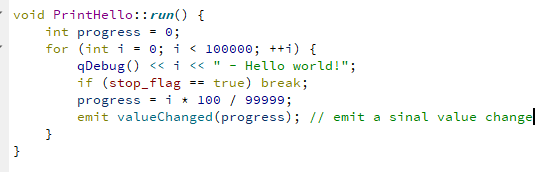
1. Create thread in Qt

* Create class inherits **QThread** class.
* Override the **run** method and it must be “**protected**”. To start this thread, we just call **start** method.



*Note: key word “****Q\_OBJECT****” is very important.*

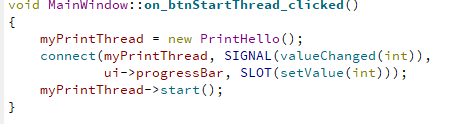
* An example for overriding run method:



Using key word “**signals**” and “**emit**” to declare and emit an signal inside a thread.

*Note: a declaration signal always “****void****”*

* Using thread:



Some outstanding methods of QThread:

Public Functions

|  |  |
| --- | --- |
| bool | [**isFinished**](qthread.html#isFinished)() const |
| bool | [**isRunning**](qthread.html#isRunning)() const |
| void | [**setPriority**](qthread.html#setPriority)(QThread::Priority *priority*) |

Public Slots

|  |  |
| --- | --- |
| void | [**quit**](qthread.html#quit)() |
| void | [**start**](qthread.html#start)(QThread::Priority *priority* = InheritPriority) |
| void | [**terminate**](qthread.html#terminate)() |

Signals

|  |  |
| --- | --- |
| void | [**finished**](qthread.html#finished)() |
| void | [**started**](qthread.html#started)() |

Static Public Members

|  |  |
| --- | --- |
| Qt::HANDLE | [**currentThreadId**](qthread.html#currentThreadId)() |
| int | [**idealThreadCount**](qthread.html#idealThreadCount)() |
| void | [**msleep**](qthread.html#msleep)(unsigned long *msecs*) |
| void | [**sleep**](qthread.html#sleep)(unsigned long *secs*) |
| void | [**usleep**](qthread.html#usleep)(unsigned long *usecs*) |
| void | [**yieldCurrentThread**](qthread.html#yieldCurrentThread)() |

Protected Functions

|  |  |
| --- | --- |
| int | [**exec**](qthread.html#exec)() |
| virtual void | [**run**](qthread.html#run)() |