Interfaces homme-machine

Marc Silanus marc.silanus@univ-avignon.fr

Interface homme-machine avec PyQt

Introduction à PyQt

PyQt est un module libre qui permet de lier le langage Python avec la bibliothèque Qt distribué sous deux licences : une commerciale et la GNU GPL. Il permet ainsi de créer des interfaces graphiques en Python. Une extension de QtDesigner (utilitaire graphique de création d'interfaces Qt) permet de générer le code Python d'interfaces graphiques.

Wikipedia - https://fr.wikipedia.org/wiki/PyQt

- Multi-Plateforme
  - Windows
  - Mac
  - Linux
- Modules dédiés à l'embarqué (Android, IOS, WinRT, BlackBerry, ...)

Pourquoi Qt?

- Multi-Plateformes
- Performance
- Relativement Simple
- Gratuit (GPL) et code source Nombreux
- outils
  - Générateur d'interface : Qt Designer
  - Internationalisation : Qt Linguist
  - Documentation : Qt Assistant
  - Examples : Qt Examples

Qui utilise Qt ?

ESA, Nokia, Nasa, Adobe, Motorola, Google, ...

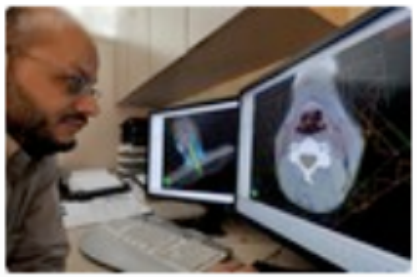
Domaines d'utilisation

Qt in Automotive Infotainment

Qt in Aerospace

Qt in Home Media

Qt in IP Communication



Qt in Medical

Qt in Oil & Gas

Qt in Visual Effects

PyQt

- Bindings Python2 et Python3 pour Qt
- Développé par Riverbank Computing Limited
- Binding les plus populaires :
  - PyQt5 pour Qt5 (+ récent)
  - PyQt4 pour Qt4
- PyQt peut générer du code python depuis Qt Designer
- Possibilité d'ajouter des widgets écrits en PyQt à Qt Designer
- L'interface graphique générée est une classe à importée

Les principaux modules

- **QtCore**
- **QtWidgets**
- **QtGUI**
- QtBluetooth
- QtOpenGL
- QtSript/QtScriptTools
- QtSql
- QtSvg
- QtWebKit
- QtXml/QtXmlPatterns
- QtMultimedia
- QtSensors

Les principaux modules

QtCore

Il gère :

- les types de base
- les containers
- le systèmes de fichiers
- les objets graphiques
- les threads
- les timers

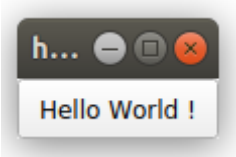
Les principaux modules

QtWidets

Il gère les widgets qui constituent l'interface graphique

- QPushButton
- QLabel
- QEdit
- QRadioButton
- QCheckBox
- QSlider
- QProgressBar
- ...

Hello Word !




```
from PyQt5.QtCore import *
fromPyQt5.QtWidgets import *
import sys

def main(args):
    app = QApplication(args)
    button = QPushButton("Hello World !", None)
    button.resize(100,30)
    button.show()
    app.exec_()

if __name__ =="__main__":
    main(sys.argv)
```

interactivité : Signal / Slot

Signal

Lorsqu'une action est effectuée sur un widget (click, survol, ...) un évènement est généré et un signal est envoyé à l'IHM.

Slot

Un Slot est une méthode de l'IHM. Si un signal est connecté à un Slot, il sera exécuté.
Les widgets possèdent déjà des signaux/slots mais le programmeur peut définir les siens

```
button.connect(button, QtCore.SIGNAL("clicked()"),\
        app, QtCore.SLOT("quit()"))
```

QtDesigner

Si vous avez installé Anaconda et/ou pycharm, vous avez déjà PyQt et QtDesigner

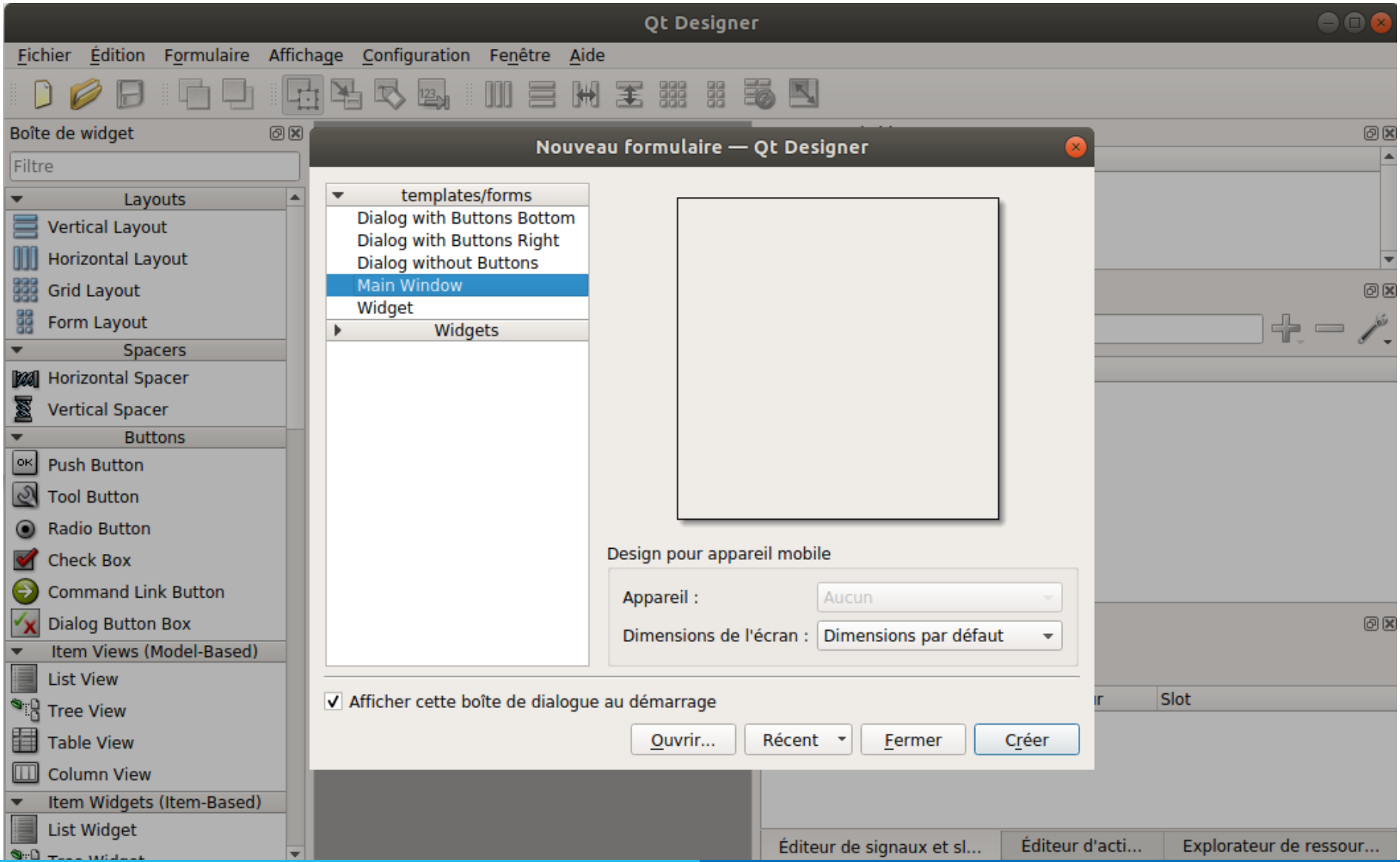
sinon :

```
$ sudo apt-get install python3-pyqt5
$ sudo apt-get install qttools5-dev-tools
```

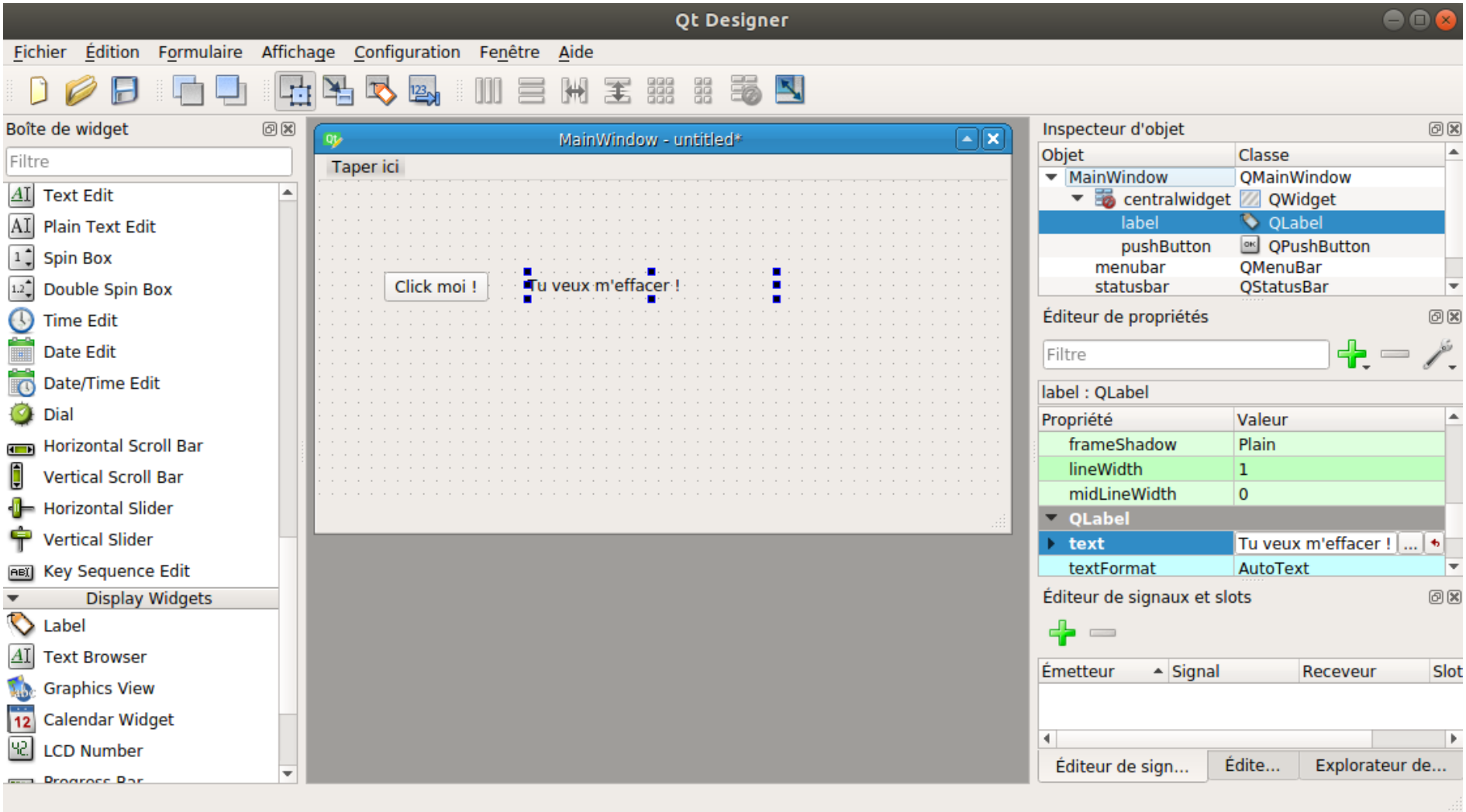
Lancement :

```
$ designer
```

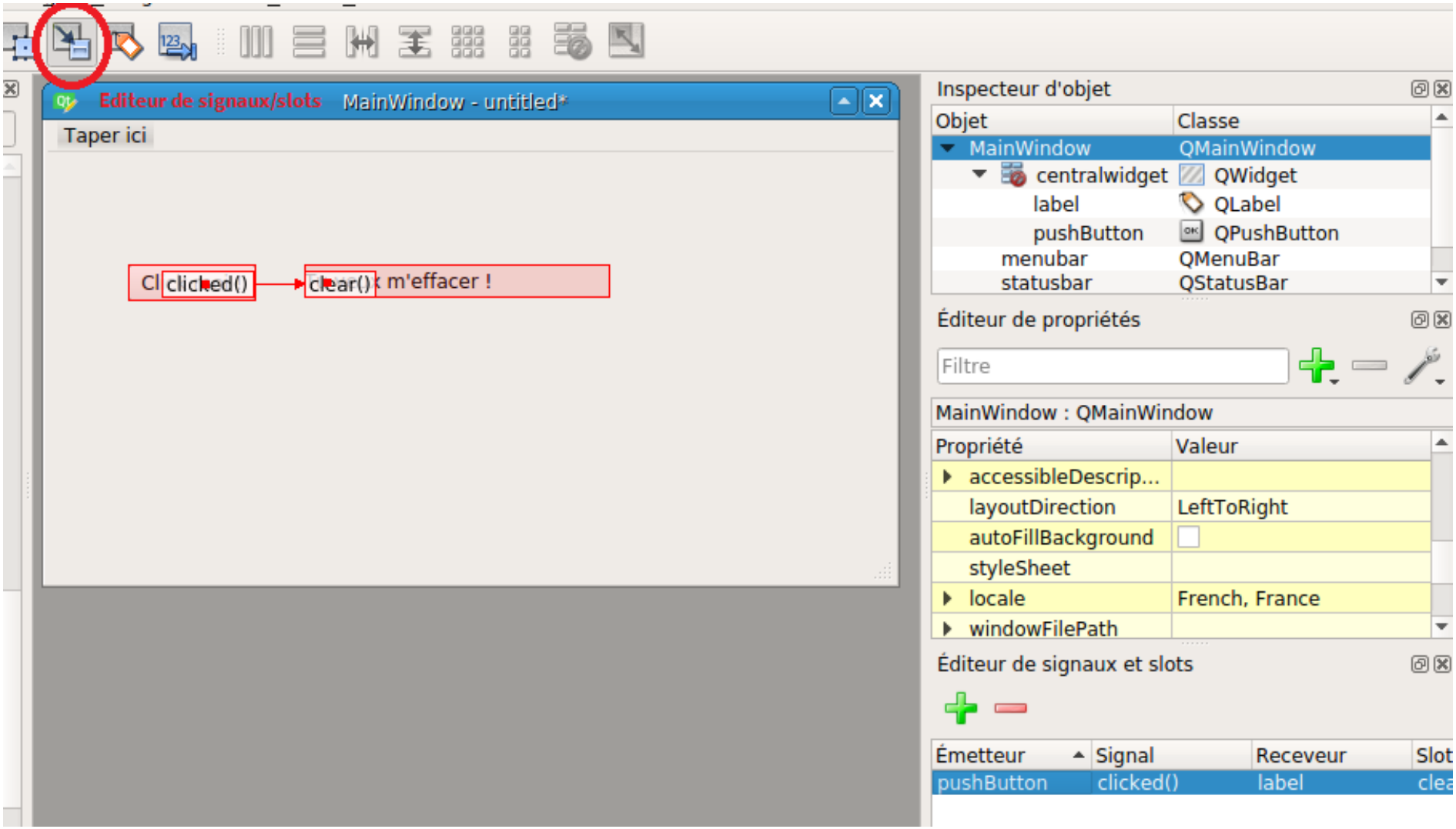
Pour une IHM : Créer Main Window



Glisser/Déplasser les widgets



Connecter Signal/Slot



QtDesigner

Enregistrer au format .ui et transformer en .py

Utilitaire pyuic5

Générer la classe UI_MainWindow :

```
$ pyuic5 clickmoi.ui -o clickmoi.py
```

Avec le main pour l'exécuter :

```
$ pyuic5 -x clickmoi.ui -o clickmoi.py
$ python3 clickmoi.py
```

QtDesigner - code généré

```
from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(528, 288)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.pushButton = QtWidgets.QPushButton(self.centralwi
        ...
```

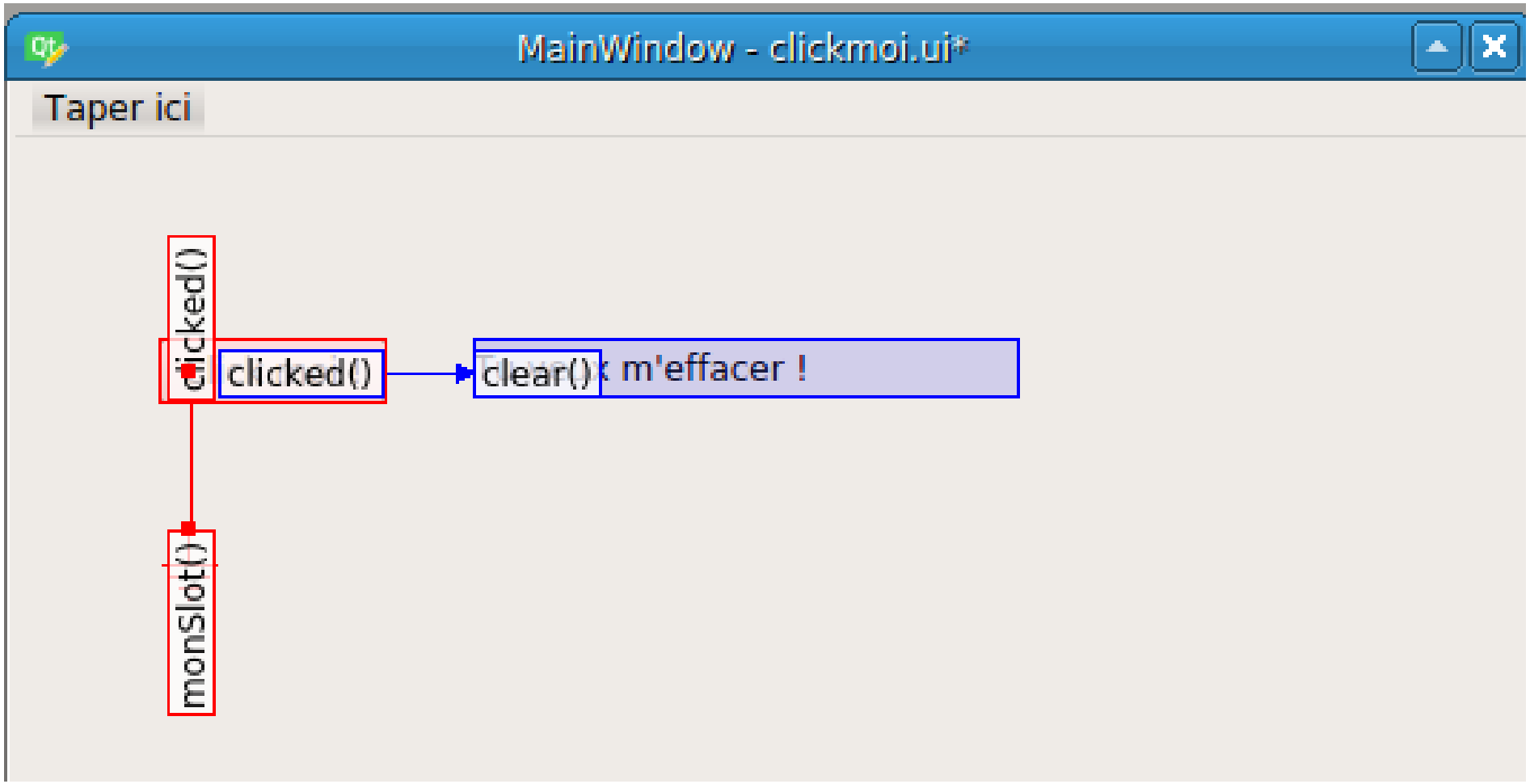
QtDesigner - code généré

avec l'option -x :

```
if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

Mes propres slots

Ne pas oublier de régénerer le .py avec `pyuic5`



Importer l'IHM

Nouveau script python

mon_clickmoi.py

```
from PyQt5 import QtWidgets
import sys

# importer du  module correspondant à l'IHM
# la classe Ui_MainWindow
# généré avec pyuic5
from clickmoi import Ui_MainWindow
```

Générer l'application :

Classe applicationIHM

```
# le nom de la classe n'a pas d'importance
class applicationIHM(QtWidgets.QMainWindow):
    # son constructeur
    def __init__(self):
        super(applicationIHM, self).__init__()

        # on instancie un objet de classe Ui_MainWindow
        self.ui = Ui_MainWindow()
        # on invoque son constructeur
        self.ui.setupUi(self)

    # Definition de mes slots personnels
    def monSlot(self):
        self.ui.label_2.setText("Me revoilà...")
```

Générer l'application :

Fonction main()

```
# exécute l'application
def main():
    app = QtWidgets.QApplication(sys.argv)
    application = applicationIHM()
    application.show()
    sys.exit(app.exec_())

if __name__ == "__main__":
    main()
```

Documentation officielle

https://www.riverbankcomputing.com/static/Docs/PyQt5/index.html