IHM avec Qt

Connexion des signaux-slots

DAKKAR Borhen-eddine

Lycée le Corbusier

BTS SN-IR

Table des matières

1 Table des matières

2 Connexion basique des signaux-slots

3 Utilisation de la méthode sender

4 Utilisation de classe dérivée

Connexion basique des signaux-slots

- Qt utilise un mécanisme de communication d'objets appelé signal/slot.
- L'idée du signal-slot est de créer un «lien» entre deux fonctions membres de classes indépendantes.
- Pour établir la connexion signal/slot la commande suivante est nécessaire:

```
connect(objet1,SIGNAL(signal1()),objet2,SLOT(slot1()))
connect(objet1,SIGNAL(signal1()),objet2,SLOT(slot2()))
```

Exemple d'un clavier numérique

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QPushButton>
#include <QGridLayout>
#include <QLabel>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE
class MainWindow : public QMainWindow
{
 Q_OBJECT
public:
MainWindow(QWidget *parent = nullptr);
~MainWindow();
QPushButton *buttons[10];
QGridLayout *Grille_boutons;
QLabel *label = new QLabel(this);
private:
Ui::MainWindow *ui;
signals:
void digitClicked(int digit);
private slots:
void slot_button0();
...
void slot_button9();
};
#endif // MAINWINDOW_H
```

mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
MainWindow::MainWindow(QWidget *parent)
: QMainWindow(parent)
, ui(new Ui::MainWindow)
{
ui->setupUi(this);
setWindowTitle("Clavier numérique");
resize(300,300);
for (int i = 0; i < 10; ++i)
{
    QString text = QString::number(i);
    buttons[i] = new QPushButton(text, this);
}
connect(buttons[0], SIGNAL(clicked()), this, SLOT(slot_button0()));
...
connect(buttons[9], SIGNAL(clicked()), this, SLOT(slot_button9()));
QGridLayout *layout = new QGridLayout();
layout->setMargin(6);
layout->setSpacing(6);
for (int i = 0; i<9; ++i)
{
    layout->addWidget(buttons[0], 3, 1);
    layout->addWidget(buttons[i+1], i / 3, i % 3);
}
```

mainwindow.cpp

```
label->setFrameStyle(QFrame::Panel | QFrame::Sunken);
label->setAlignment(Qt::AlignCenter);
layout->addWidget(label, 4, 0, 1, 3);

QWidget *Widget = new QWidget;
Widget->setLayout(layout);
setCentralWidget(Widget);
}
//--- Définition des slots ---//
void MainWindow ::slot_button0()
{
 emit digitClicked(0);
label->setText("buttons_0");
}
...
void MainWindow::slot_button9()
{
emit digitClicked(9);
label->setText("buttons_9");
}

MainWindow::~MainWindow()
{
delete ui;
}
```

main.cpp

```cpp
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

- Dans l'éxemple précedent chaque signal clicked() a été connecté à un slot button0Clicked().

- Les slots émettant le signal digitClicked(int) avec comme paramètre le numéro du bouton correspondant (0 à 9).

- Ils font appel aussi à un Qlabel qui affiche dans une zone de text le bouton appuyer label->setText("buttons_0").

- Nous pouvons constater que la flexibilité de cette méthode est discutable. Elle présente une source d'erreurs dans le cas de plusieurs connexions.

Utilisation de la méthode sender

QObject::sender()

- Renvoie un pointeur vers l'objet qui a envoyé le signal, s'il est appelé dans un slot activé par un signal; sinon elle retourne un pointeur null (nullptr).
- Le pointeur renvoyé par cette fonction devient invalide si l'expéditeur est détruit ou si le slot est déconnecté du signal de l'expéditeur.
- Comme c'est une fonction protégée, elle viole le principe de modularité orienté objet. Cependant, accéder à l'expéditeur peut être utile lorsque de nombreux signaux sont connectés à un seul slot.

Constructeur de la classe mainwindow

```
MainWindow::MainWindow(QWidget *parent)
: QMainWindow(parent)
, ui(new Ui::MainWindow)
{
ui->setupUi(this);
setWindowTitle("Clavier numérique");
resize(300,300);

for (int i = 0; i < 10; ++i) {
        QString text = QString::number(i);
        buttons[i] = new QPushButton(text, this);
        connect(buttons[i], SIGNAL(clicked()), this, SLOT(buttonClicked()));
}

QGridLayout *layout = new QGridLayout();
layout->setMargin(6);
layout->setSpacing(6);
for (int i = 0; i<9; ++i)
{
    layout->addWidget(buttons[0], 3, 1);
    layout->addWidget(buttons[i+1], i / 3, i % 3);
}
```

Constructeur de la classe mainwindow

```
label->setFrameStyle(QFrame::Panel | QFrame::Sunken);
label->setAlignment(Qt::AlignCenter);
layout->addWidget(label, 4, 0, 1, 3);

QWidget *Widget = new QWidget;
Widget->setLayout(layout);
setCentralWidget(Widget);
}
```

Le slot buttonClicked devient:

Le slot buttonClicked

```
void MainWindow :: buttonClicked()
{
QPushButton *button = (QPushButton *)sender();
emit digitClicked(button->text()[0].digitValue());
int N_bouton;
N_bouton = button->text()[0].digitValue();
label->setText("Bouton "+ QString::number(N_bouton));
}
```

- Le code est moins long. L'utilisation de la boucle "for" pour la définition des boutons et leurs connexions était très utile.

- Pour le slot buttonClicked nous avons commencer par appeler la méthode sender() pour récupérer un pointeur vers le QObject qui a émis le signal.

- Nous savons que l'émetteur du signal est un QPushButton, donc la valeur de retour de sender() est un QPushButton *.

- Nous avons utilisé la méthode text() pour récupérer le text écrit sur le bouton cliquer. Ensuite, nous l'avons affiché sur un Qlabel.

Inconvénients de la méthode sender

- Le premier inconvénient de cette approche est que nous avons besoin d'un slot privé pour effectuer le démultiplexage. Le code de buttonClicked() n'est pas très élégant; si vous remplacez soudainement les QPushButtons par un autre type de widget et oubliez de le changer dans la définition, vous obtiendrez un crash.

- De même, si vous modifiez le texte sur les boutons (par exemple, "NIL" au lieu de "0"), le signal digitClicked(int) sera émis avec une valeur incorrecte.

- Enfin, l'utilisation de sender() conduit à des composants étroitement couplés, que de nombreuses programmeurs considèrent comme un mauvais style de programmation.

Utilisation de classe dérivée

- Cette approche ne nécessite aucun slot privé.

- A la place des slots privés, nous allons s'assurer que les boutons eux-mêmes émettent un signal clicked(int) qui peut être directement connecté au signal digitClicked(int) du MainWindow.

- Cela nécessite d'utiliser un héritage de la classe QPushButton.

Utilisation de classe dérivée

La classe KeypadButton

```
 class KeypadButton : public QPushButton
{
    Q_OBJECT
public:
    KeypadButton(int digit, QWidget *parent);

signals:
    void clicked(int digit);
private slots:
    void reemitClicked();

private:
    int myDigit;
};
KeypadButton::KeypadButton(int digit, QWidget *parent)
    : QPushButton(parent)
{
    myDigit = digit;
    setText(QString::number(myDigit));
    connect(this, SIGNAL(clicked()), this, SLOT(reemitClicked()));
}
void KeypadButton::reemitClicked()
{
    emit clicked(myDigit);
}
```

Utilisation de classe dérivée

Constructeur de la MainWindow

```
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    setWindowTitle("Clavier numérique");
    resize(300,300);

for (int i = 0; i < 10; ++i) {
    buttons[i]=new KeypadButton(i,this);
    connect(buttons[i], SIGNAL(clicked(int)),this, SIGNAL(digitClicked(int)));

    }
}
```