IHM avec Qt

Programmation avacée des widgets

DAKKAR Borhen-eddine

Lycée le Corbusier

BTS SN-IR

Table des matières

La classe QSettings

- La classe QSettings fournit des paramètres d'application persistants.
- Les utilisateurs s'attendent généralement à ce qu'une application mémorise ses paramètres (tailles et positions des fenêtres, options, etc.) à travers les sessions.
- Lors de la création d'un objet QSettings, vous devez transmettre le nom de votre société ou organisation ainsi que le nom de votre application. Par exemple, si votre produit s'appelle "Star Runner" et que votre entreprise s'appelle "MySoft", vous construirez l'objet QSettings comme suit:

Création d'un objet QSettings

```
QSettings settings("MySoft", "Star Runner");
```

- Le premier paramètre est le nom de votre organisation. Le deuxième paramètre est le nom du programme qui utilise l'annuaire [1].

Exemple d'utilisation de la classe QSettings

QSettings

```
QLabel *label = new QLabel;
QSettings settings("Le Corbusier", "BTS SN2");
int Donnees_persistantes  = settings.value("Test").toInt();


if (Donnees_persistantes==NULL){
    label->setText("La variable \"Test\" n'a pas encore été créée.");
    settings.setValue("Test", 1);}
else {
        settings.setValue("Test", Donnees_persistantes+1);
        label->setText("La valeur de \"Test\" est:"+
        QString::number(Donnees_persistantes));
    }
//--- Un label pour afficher la valeur de QSettings ---//

label->setAlignment(Qt::AlignCenter);
setCentralWidget(label);
```

La classe QStackedLayout

- La classe QStackedLayout fournit une pile de widgets où un seul widget est visible à la fois.
- La classe QStackedLayout peut être utilisée pour gérer des onglets d'une application graphique.
- L'indice du widget affiché à l'écran est donné par currentIndex() et peut être modifié à l'aide de setCurrentIndex() [2].

Exemple QStackedLayout

Color.h

```
#ifndef COLOR_H
#define COLOR_H

#include <QWidget>

QT_BEGIN_NAMESPACE
namespace Ui { class Widget; }
QT_END_NAMESPACE

class Color : public QWidget
{
    Q_OBJECT

public:
    Color(QWidget *parent = nullptr);
    Color(QString couleur);
    ~Color();

private:
    //Ui::Widget *ui;
};
#endif // COLOR_H
```

Color.cpp

```
#include <Color.h>
#include <QPalette>
#include <QString>
Color::Color(QWidget *parent)
    : QWidget(parent)
{
}

Color::Color(QString couleur)
{
    //setGeometry(0, 0, 300, 300);
    this->setAutoFillBackground(true); //# couleur d'arrière plan
     QPalette myPalette = palette();// # définir une palette
    // On ajoute la couleur passée en paramètres
    //QString couleur = "red";
    myPalette.setColor(QPalette::Window, QColor(couleur));
    setPalette(myPalette);// # Palette ajoutée au widg
}

Color::~Color()
{
}
```

MainWindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QVBoxLayout>
#include <QHBoxLayout>
#include <QStackedLayout>
#include <QPushButton>
#include <Color.h>
#include <QStatusBar>
#include <QLabel>
QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    QVBoxLayout *pageLayout = new QVBoxLayout();
    QHBoxLayout *buttonLayout = new QHBoxLayout();
    QStackedLayout *stackedLayout = new QStackedLayout();
    QPushButton *Bouton[3];
    QString List_color[3];
    Color *Couleur[3];
private slots:
    void changer_couleur();
private:
    Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

MainWindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <Color.h>
#include <QPushButton>
#include <QStatusBar>
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    setWindowTitle("Utilisation de QStackedLayout");
    pageLayout->addLayout(buttonLayout);// Les boutons horizontaux
    pageLayout->addLayout(stackedLayout);// Le stacked layout
    List_color[0] = "red";
    List_color[1] = "green";
    List_color[2] = "blue";
    for (int i = 0; i<3; ++i) {
            QString text = QString::number(i);
            Bouton[i] = new QPushButton(List_color[i], this);
}
```

MainWindow.cpp

```
for (int i = 0; i<3; ++i)
    {
        buttonLayout->addWidget(Bouton[i]);
        connect(Bouton[i],SIGNAL(clicked()),SLOT(changer_couleur()));
        Couleur[i] = new Color(List_color[i]);
        stackedLayout->addWidget(Couleur[i]);
    }
    QWidget *widget = new QWidget();
    widget->setLayout(pageLayout);
    setCentralWidget(widget);
    }
void MainWindow::changer_couleur(){
    QPushButton *button = (QPushButton *)sender();
    QString color = button->text();
    for (int i=0; i<3; i++)
    {
        if(color == List_color[i])
        {
            stackedLayout->setCurrentIndex(i);
        }
    }
}
MainWindow::~MainWindow()
{
    delete ui;
```

main.cpp

```
#include "mainwindow.h"

#include <QApplication>

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

La classe QTabWidget

- Un widget onglet (tab widget) fournit une barre d'onglets et une zone de page qui est utilisée pour afficher les pages liées à chaque onglet.
- Par défaut, la barre d'onglets est affichée au-dessus de la zone de page, mais différentes configurations sont disponibles:
  1. QTabWidget::North: Les onglets sont dessinés au-dessus des pages.
  2. QTabWidget::South: Les onglets sont dessinés en dessous les pages.
  3. QTabWidget::West: Les onglets sont dessinés à gauche des pages.
  4. QTabWidget::East: Les onglets sont dessinés à droite des pages.
- Chaque onglet est associé à un widget différent (appelé une page). Seule la page actuelle est affichée dans la zone de page.
- La position des onglets est définie par tabPosition, leur forme est définie par tabShape [3].

GeneralTab.h

```
#ifndef GENERALTAB_H
#define GENERALTAB_H
#include <QWidget>
class GeneralTab : public QWidget
{
  Q_OBJECT

public:
  GeneralTab(QWidget *parent = nullptr);
};
#endif // GENERALTAB_H
```

GeneralTab.cpp

```
#include <GeneralTab.h>
#include <QLabel>
#include <QLineEdit>
#include <QVBoxLayout>
GeneralTab::GeneralTab(QWidget *parent)
    : QWidget(parent)
{
QLabel *fileNameLabel = new QLabel(tr("Nom du ficheir:"));
QLineEdit *fileNameEdit = new QLineEdit("QTabWidget.h");

QLabel *pathLabel = new QLabel(tr("Chemin:"));
QLabel *pathValueLabel = new QLabel("C:/Users");
pathValueLabel->setFrameStyle(QFrame::Panel | QFrame::Sunken);

QLabel *sizeLabel = new QLabel(tr("Taille:"));
QLabel *sizeValueLabel = new QLabel("24 ko");
sizeValueLabel->setFrameStyle(QFrame::Panel | QFrame::Sunken);

QVBoxLayout *mainLayout = new QVBoxLayout;
mainLayout->addWidget(fileNameLabel);
mainLayout->addWidget(fileNameEdit);
mainLayout->addWidget(pathLabel);
mainLayout->addWidget(pathValueLabel);
mainLayout->addWidget(sizeLabel);
mainLayout->addWidget(sizeValueLabel);
mainLayout->addStretch(1);
setLayout(mainLayout);
}
```

Parametres.h

```
#ifndef PARAMETRES_H
#define PARAMETRES_H
#include <QWidget>
class Parametres : public QWidget
{
    Q_OBJECT

public:
    Parametres(QWidget *parent = nullptr);
};
#endif // PARAMETRES_H
```

Parametres.cpp

```
#include <Parametres.h>
#include <QLabel>
#include <QLineEdit>
#include <QPushButton>
#include <QVBoxLayout>
Parametres::Parametres(QWidget *parent)
    : QWidget(parent)
{
QPushButton *B1 = new QPushButton("Performances");
QPushButton *B2 = new QPushButton("Démarrage");

QLabel *Performances = new QLabel(tr("Effets visuels,
                          planification du processeur."));
QLabel *Demarrage = new QLabel(tr("Informations de démarrage
                                  "du système..."));

QVBoxLayout *V_layout = new QVBoxLayout;

V_layout->addWidget(Performances);
V_layout->addWidget(B1);
V_layout->addWidget(Demarrage);
V_layout->addWidget(B2);

setLayout(V_layout);
}
```

MainWindow.h

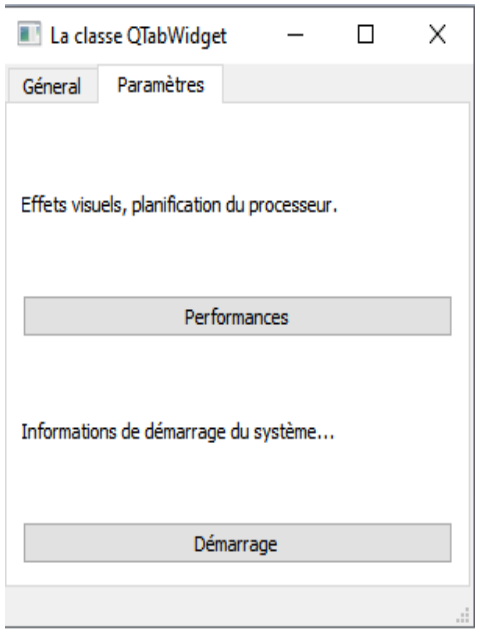
```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QTabWidget>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    QTabWidget *widget_onglets = new QTabWidget;

private:
    Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

MainWindow.cpp

```cpp
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <GeneralTab.h>
#include <Parametres.h>
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    setWindowTitle("La classe QTabWidget");
    this->resize(QSize(300,300));
    widget_onglets->setMovable(true); //Permet de dépm=lacer les onglets
    widget_onglets->setTabPosition(QTabWidget::North);

    GeneralTab *General = new GeneralTab;
    Parametres *Param = new Parametres;
    widget_onglets->addTab(General,"Géneral");
    widget_onglets->addTab(Param,"Paramètres");
    setCentralWidget(widget_onglets);
}

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

main.cpp

```
#include "mainwindow.h"

#include <QApplication>

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



Références


QSettings. https://doc.qt.io/qt-5/qmenu.htmldetails. 2020.


QStackedLayout. https://doc.qt.io/qt-5/qstackedlayout.htmldetails. 2020.


QTabWidget. https://doc.qt.io/qt-5/qtabwidget.htmlTabPosition-enum. 2020.