Tổng hợp những kinh nghiệm code Qt

Nội dung

1. Thiết lập bố cục, kích cỡ các thuộc tính UI

2. Làm việc với mainwindow

3. Deploy output .exe

4. Signal and Slot

5. QDebug

6. QString

7. QMessageBox

8. QFile

9. Qwidget – Tạo các Tab

10. QCloseEvent – Hỏi xác nhận trước khi đóng Form

11. QTimer

12. QPixMap – Chèn hình vào label

13. Qdialog – Hiển thị cửa sổ thứ hai

14. Add resource file \*.qrc

15. QAction - tạo menu và các thanh công cụ (File, Edit, About)

16. QtSerialPort - Giao tiếp cổng COM

17. QSettings – Lưu các cài đặt của ứng dụng

**Thiết lập bố cục, kích cỡ các thuộc tính UI**

Thiết lập kích cỡ cho các thuộc tính button, lineEdit, textEdit...

Ui->pushbutton->setFixedHeight(30);

Ui->lineEdit->setFixedWidth(100);

Ui->textEdit->setFixedSize(100, 30);

**Làm việc với Mainwindow**

**Thiết lập title cho Mainwindow**

Ở file mainwindow.cpp

This->setWindowTitle(“Hello World”);

**Thiết lập icon cho Mainwindow**

Copy file icon.ico vào cùng đường dẫn với source file .pro

Ở cửa số code Qt, mở file example.pro

RC\_ICONS += icon.ico // Bắt buộc phải là .ico

Clean All > Run qmake > Build again.

**Deploy output .exe**

Cách output .exe file có thể chạy ở trên máy tính khác không cần cài Qt

Nguồn: <https://wiki.qt.io/Deploy_an_Application_on_Windows>

Bước 1: Thiết lập build Release

Cửa số QT, chọn Projects (Ctrl + 4 hoặc 5 tùy phiên bản).

Ở mục Build & Run > Build Settings > Chọn Release

Build và chạy project lại, lúc này file output .exe sẽ nằm ở thư mục mới, ví dụ:

C:\Users\ADMIN\Documents\QT\_Projects\build-03\_Login\_App-Desktop\_Qt\_5\_7\_0\_MSVC2015\_32bit-Release

Bước 2: Copy các thư viện cần thiết để chạy file .exe

Tùy trình biên dịch của bạn mà lựa chọn các file .dll cần thiết, ví dụ:

Visual C 2015 - C:\Qt\5.7\msvc2015\

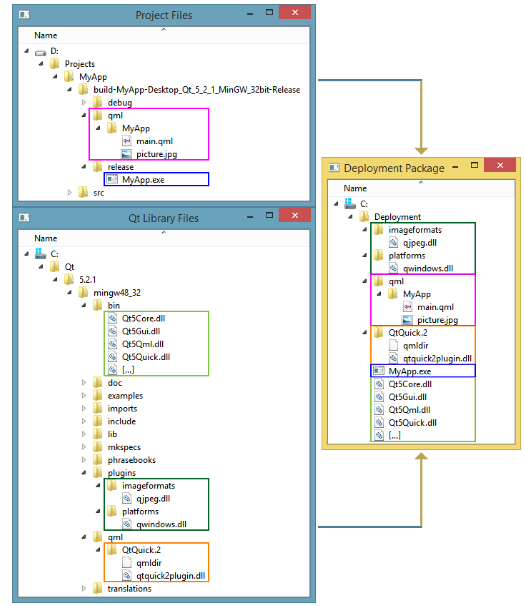
Mingw - C:\Qt\5.7\mingw\_48\_32\

Copy các file ở 3 thư mục bin, plugin, qml vào địa chỉ release output. (để ý các đường dẫn như hình bên dưới)

Bước 3: Chạy file .exe

Ctrl + A và chọn delete hết. Skip các file không xóa được.

Lúc này, chỉ còn lại các file cần thiết để chạy chương trình.



**Signal and Slot**

Cách 1: Sử dụng chức năng Edit Signals/Slots (F4) trên toolbar

Cách 2: Sử dụng hàm có sẵn:

Connect(ui->btnExit, SIGNAL(clicked()), this, SLOT(close()));

Cách 3: Viết thêm hàm handle:

Connect(ui->btnHello, SIGNAL(clicked()), this, SLOT(displayText()));

Mở header mainwindow.h, thêm trường:

Private slots:

Void displayText();

Mở mainwindow.cpp:

Void MainWindow::displayText() {

Ui->label\_Hello->setText(“Hello World”);

}

**Debug với thư viện QDebug**

#include <QDebug>

qDebug << “Hello”; // Tự động xuống dòng sau khi in chuỗi

Kết quả được in ra ở cửa sổ Application Output

**Xử lý Qstring**

**Khai báo**

Qstring str1 = “Hello”;

Qstring str2 = Qstring (3, ‘X’); // str2 = “XXX”

**Tính số ký tự của Qstring**

Qstring str = “World”;

int n = str.size(); // n == 5;

int l = str.length(); // l == 5;

**Kiểm tra Qstring**

Hàm isEmpty() trả về true nếu Qstring không chứa kí tự nào

Qstring().isEmpty(); // true

Qstring(“”).isEmpty(); // true

Qstring(“x”).isEmpty(); // false

Hàm isNull() trả về true nếu chuỗi rỗng

Qstring().isNull(); // true

Qstring(“”).isNull(); // false

Qstring(“x”).isNull(); // false

Hàm startsWith Kiểm tra Qstring có bắt đầu bằng một chuỗi con hay không

Qstring str = “Hello World”;

Str.startsWith(“Hello”); // true

Str.startsWith(“Car”); // false

Hàm endsWith kiểm tra chuỗi kết thúc bằng một chuỗi con hay không

Qstring str = “Hello World”;

Str.endsWith(“World”); // true

Str.endsWith(“Car”); // false

Hàm contains kiểm tra Qstring có chứa Qstring con hay không

Qstring str = “Hello World”;

bool isExist = str.contains(“or”); // true

**Xóa Qstring**

Hàm chop(int n) xóa đi n kí tự cuối của Qstring

Qstring str = “LOGOUT\r\n”;

Str.chop(2); // str == “LOGOUT”

Hàm clear() xóa toàn bộ nội dung của Qstring, Qstring sẽ trở thành rỗng

Qstring str = “LOGOUT\r\n”;

Str.clear();

qDebug() << str.isNull(); // true

Hàm truncate(int n) xóa các kí tự từ vị trí n đến cuối Qstring

Qstring str = “LOGOUT\r\n”;

Str.truncate(5); // str = “LOGOU”;

**Ghép Qstring**

Dùng toán tử (+) để ghép các Qstring lại với nhau

Qstring str1 = “Hello”;

String str2 = “World”;

Qstring str = str1 + “ “ + Str2; // str == “Hello World”

Hàm append() ghép thêm một Qstring vào sau một Qstring cho trước.

Qstring str = “Hello”;

Str.append(“ World”); // str == “Hello World”

Hàm prepend() ghép thêm một Qstring vào phía trước Qstring đã có

Qstring str = “World”;

Str.prepend(“Hello “); // str == “Hello World”

Hàm insert() ghép thêm một Qstring vào vị trí bất kì của một Qstring cho trước

Qstring str = “I Tung”;

Str.insert(2, “am “); // str == “I am Tung”

**Cắt Qstring**

Lấy ra Qstring con của từ một Qstring cho trước

Qstring str = “Hello World”;

Qstring str1 = str.left(5); // str1 = “Hello”

Qstring str2 = text.right(5); // str2 = “World”

Qstring str3 = text.mid(6, 3); // str3 = “Wor”

Lấy ký tự tại một vị trí bất kì (at)

Qstring str = “Hello World”;

Qchar chr = str.at(1); // chr = ‘e’

**Đổi sang chữ thường hoặc chữ hoa**

Qstring str = “Hello World”;

Qstring str1 = str.toLower(); // str1 = “hello world”

Qstring str2 = str.toUpper(); // str2 = “HELLO WORLD”

**Đổi int sang Qstring**

Cách 1: Dùng phương thức Qstring::number, áp dụng cho tất cả các kiểu số

Int i = 10;

Qstring str = Qstring::number(i); // str = “10”

Riêng đối với số nguyên, có thể chuyển sang Qstring ở các hệ số khác nhau

Str = Qstring::number(i, 16); // str = “a”, hệ Hexa

Cách 2: Dùng phương thức setNum() của lớp Qstring, áp dụng cho tất cả các kiểu số

Double pi = 3.14;

Qstring str;

Str.setNum(pi); // str = “3.14”

Cách 3: Áp dụng khi cần ghép số vào Qstring

Int year = 2016

Qstring str = Qstring(“This year is %1”).arg(year);

// “This year is 2016”

Int day = 16;

Int month = 10;

Qstring date = Qstring(“This date is %1/%2/%3).arg(day).arg(month).arg(year);

// This date is 16/10/2016

**Đổi QString sang int**

Sử dụng phương thức toInt (toFloat, toDouble,...) của lớp Qstring

Qstring str = “2016”;

Int number = str.toInt(); // number = 2016

**Class QmessageBox**

Yêu cầu

#include <QMessageBox>

Information

QmessageBox::information(this, “Title”, “Message”);

Warning

QmessageBox::warning(this, “Title”, “Warning”);

Critical

QmessageBox::critical(this, “Title”, “Critical”);

Question

If (QmessageBox::question(this, “Title”, “Question”) == QmessageBox::Yes) {

// do Sth if click YES

} else {

// do Str if click NO

}

Trường hợp muốn đổi tên 2 button thành “Ok” và “Cancel”

QmessageBox::question(this, “Title”, “Question”, QmessageBox::Ok | QmessageBox::Cancel)

**Xử lý Qfile**

Yêu cầu:

#include <QFile>

#include <QFileDialog>

#include <QTextStream>

Khai báo biến private ở thư viện mainwindow.h

QString file\_path;

Nhập file có sẵn và Đọc thông tin của nó

QString file\_name = QFileDialog::getOpenFileName(this, "Open the file");

QFile file(file\_name);

file\_path = file\_name;

if (!file.*open*(QFile::ReadOnly | QFile::Text)) {

QMessageBox::warning(this, "...", "File not open");

return;

}

QTextStream in(&file);

QString text = in.readAll();

ui->textEdit->setText(text);

file.close();

Tạo mới một file và ghi thông tin vào nó

file\_path = "C:\new\_file.txt";

QFile file(file\_path);

if (!file.*open*(QFile::WriteOnly | QFile::Text)) {

QMessageBox::warning(this, "...", "File not open");

return;

}

QTextStream out(&file);

QString text = ui->textEdit->toPlainText();

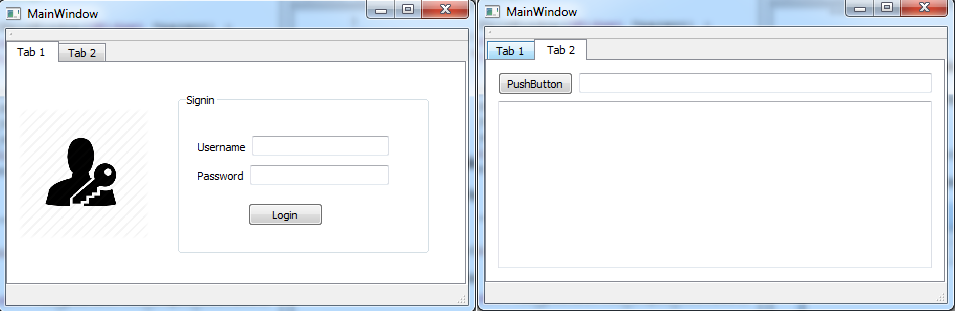
out << text;

file.flush();

file.*close*();

**Qwidget – Tạo các tab**

Sử dụng công cụ Tab Widget để tạo các tab làm việc khác nhau



Chỉnh Widget chiếm toàn bộ cửa sổ Mainwindow:

this->setCentralWidget(ui->tabWidget);

**QCloseEvent – Hỏi xác nhận trước khi đóng Form**

Khai báo hàm ở header mainwindow.h

Protected:

Void closeEnvent(QCloseEvent \*event);

Viết lại hàm closeEvent() ở mainwindow.cpp

#include <QCloseEvent>

Void MainWindow::closeEvent(QCloseEvent \*event) {

If (QMessageBox::question(this, “Confirm”, “Do you want to close?”)==QMessageBox::No) {

event->ignore();

}

}

**QTimer**

Sử dụng khi cần 1 tác vụ lặp lại theo một chu kỳ nào đó.

#include <QTimer>

QTimer \*timer;

Timer = new QTimer(this);

Timer->setInterval(1000); // 1s

Connect(timer, SIGNAL(timeout()), this, SLOT(timer\_timeout()));

Timer->start();

Tạo hàm timer\_timeout() xem phần signal and slot.

**QPixMap – Chèn hình vào label**

Sử dụng label để hiển thị hình ảnh trên UI

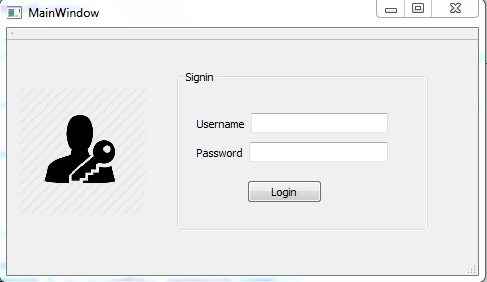
#include <QPixMap>

QPixMap pix(“C:/Users/ADMIN/Documents/QT\_Projects/03\_Login\_App/img/Login\_icon.png");

Int pix\_h = ui->label\_pic->height();

Int pix\_w = ui->label\_pic->width();

Ui->label\_pic->setPixmap(pix.scaled(pix\_w, pix\_h, Qt::KeepAspectRatio));



**Qdialog – Hiển thị cửa sổ thứ hai**

Thêm cửa số secdialog

Add New... > Qt > Qt Designer Form Class > Choose Form.

Khai báo secdialog ở thư viện mainwindow.h

#include “secdialog.h”

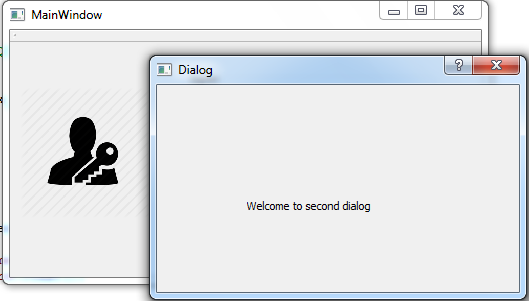
Private:

SecDialog \*secDialog;

Mainwindow.cpp, xử lý hàm slot on\_pushButton\_clicked():

secDialog = new SecDialog(this);

secDialog->show();



Cách 2: Khi cửa sổ secDialog xuất hiện, không thể thao tác với MainWindow cho tới khi đóng secDialog

#include ”secdialog.h”

SecDialog secDialog(this);

secDialog.setModal(true); // Chỉ có thể thao tác với SecDialog

dialog.exec();

**Add resource file \*.qrc**

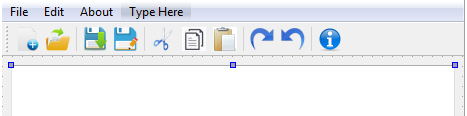
Thêm Resources file (hình ảnh, icon, ...)

Add New... > Qt > Qt Resource File > Resource name

Ở cửa sổ resource\_name.qrc,

Add Prefix

Add Files > Lựa chọn add các icon, img cần thiết như hình dưới

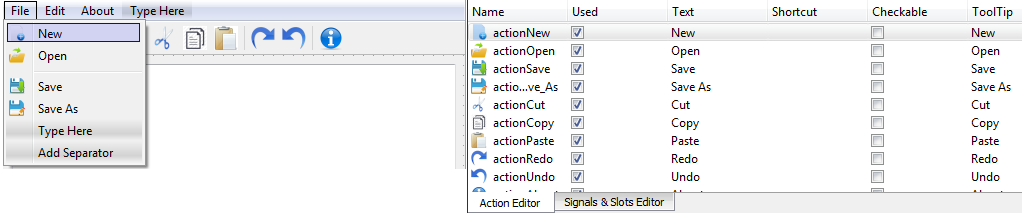


**QAction - tạo menu và các thanh công cụ (File, Edit, About)**

Tạo các thanh công cụ File, Edit, About

Mỗi action được add vào sẽ tương ứng với 1 thuộc tính trong cửa sổ Action Editor.

Sử dụng cửa sổ Action Editor để thiết lập icon, slot,.... cho action.



**QtSerialPort - Giao tiếp cổng COM**

QT 5.1 trở lên thì default. Còn các phiên bản thấp hơn phải add thêm thư viện.

Nếu máy không có cổng COM, có thể cài phần mềm Virtial Serial Port

Add vào project\_name.pro: (runqmake)

QT += core gui serialport

Mainwindow.h

Private\_slot:

Void serialport\_read(); // Hàm đọc dữ liệu từ cổng COM

Mainwindow.cpp

#include <QtSerialPort>

QSerialPort \*serialPort;

// Constructor

serialPort = new QSerialPort(this);

//Cau hinh cong COM

serialPort->setPortName("com1");

serialPort->setBaudRate(QSerialPort::Baud9600);

serialPort->setDataBits(QSerialPort::Data8);

serialPort->setParity(QSerialPort::NoParity);

serialPort->setStopBits(QSerialPort::OneStop);

serialPort->*open*(QIODevice::ReadWrite);

connect(serialPort, SIGNAL(readyRead(), this, SLOT(serialport\_read()));

// Destructor

serialPort->*close*();

// serialport\_read

serialPort->write(ui->lineEdit->text().toStdString().data());

Liệt kê danh sách cổng COM

Ở hàm constructor, sử dụng comboBox hiển thị danh sách các cổng COM.

serialPort = new QSerialPort(this);

// List all COM Port

Q\_FOREACH(QSerialPortInfo port, QSerialPortInfo::availablePorts()) {

ui->comboBox\_comport->addItem(port.portName());

}

connect(ui->pushButton\_comport, SIGNAL(clicked()), this, SLOT(serialport\_openorclose()));

Kết nối sự kiện clicked() của button để xử lý mở/đóng COM port.

void MainWindow::serialport\_openorclose() {

if (serialPort->isOpen()) {

// COM PORT CLOSE

ui->pushButton\_comport->setText("Open Port");

serialPort->*close*();

ui->label\_comport->setText(serialPort->portName() + " is disconnected");

} else {

//Cau hinh cong COM

QString portName = ui->comboBox\_comport->currentText();

serialPort->setPortName(portName);

serialPort->setBaudRate(QSerialPort::Baud9600);

serialPort->setDataBits(QSerialPort::Data8);

serialPort->setParity(QSerialPort::NoParity);

serialPort->setStopBits(QSerialPort::OneStop);

serialPort->*open*(QIODevice::ReadWrite);

connect(serialPort, SIGNAL(readyRead()), this, SLOT(serialport\_read()));

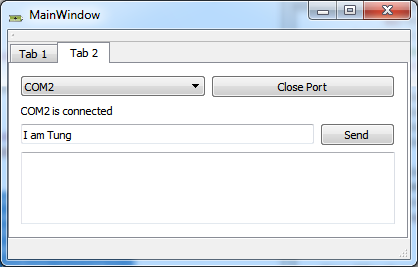
ui->label\_comport->setText(portName + " is connected");

ui->pushButton\_comport->setText("Close Port");

}

}

Giao diện như sau:



**Qsettings – Lưu các cài đặt của ứng dụng**

Ví dụ: lưu trạng thái của một CheckBox. Sau khi tắt app và bật lại, trạng thái CheckBox vẫn giữ nguyên.

Mainwindow.h

Private:

Void SaveSettings();

Void LoadSettings();

Mainwindow.cpp

#include <QSettings>

MainWindow::SaveSettings()

Qsettings settings(“MyName”, “MyApp”);

settings.beginGroup(“SaveState”);

settings.setValue(“CheckBoxState”, ui->checkBox->isChecked());

settings.endGroup();

MainWindow::LoadSettings()

Qsettings settings(“MyName”, “MyApp”);

settings.beginGroup(“SaveState”);

ui->checkBox->setChecked(settings.value(“CheckBoxState”).toBool());

settings.endGroup();

Vấn đề là đặt 2 hàm này ở đâu?

Xem chương QCloseEvent. Tạo hàm Mainwindow::closeEvent().

MainWindow::closeEvent(QCloseEvent \*event)

SaveSettings();

Còn hàm LoadSettings() được đặt ở Constructor.