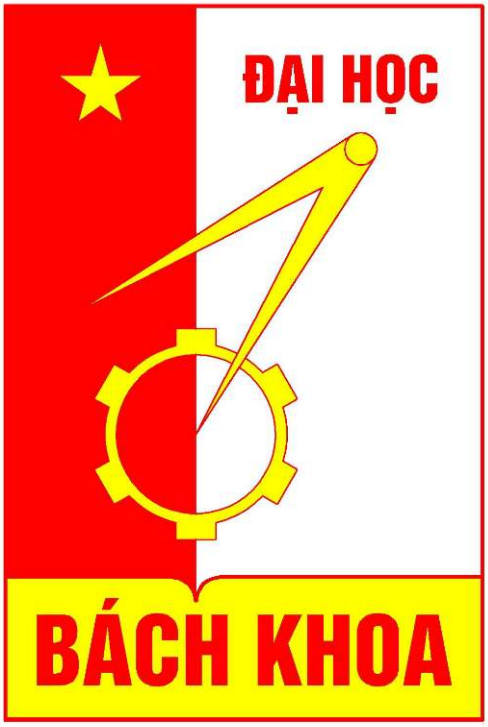
ĐẠI HỌC BÁCH KHOA HÀ NỘI



BÀI TẬP LỚN KĨ THUẬT LẬP TRÌNH

Bài 14. Tìm hiểu và sử dụng Qt để viết chương trình tạo một cửa sổ chương trình điển hình.

Giáo viên hướng dẫn: Nguyễn Việt Tùng
Sinh viên thực hiện: Nguyễn Đình Cương – MSSV: 20111202
Nguyễn Thế Tùng – MSSV: 20102493

Hà Nội, tháng 5 năm 2013

I. Giới thiệu:

Qt là nền tảng xây dựng các ứng dụng chạy được trên nhiều hệ điều hành. Phần lớn các ứng dụng xây dựng bằng Qt đều có giao diện đồ họa, do vậy Qt còn được coi như là một bộ công cụ (widget toolkit). Ban đầu Qt ra đời như một sản phẩm thương mại và cũng được dùng để viết môi trường KDE, nhưng về sau được bổ sung giấy phép LGPL, theo đó có thể được sử dụng tự do để phát triển các phần mềm nguồn mở hay đúng hơn là có thể sử dụng trong các phần mềm thương mại nếu muốn. Bản quyền thương mại của Qt hiện nay đã được chuyển qua hình thức thu phí hỗ trợ. Chúng ta có thể dùng Qt như một thư viện để viết phần mềm thương mại, nếu có sửa đổi nào trong bộ nguồn chính của Qt thì chúng ta chỉ phải cung cấp mã nguồn đã sửa ra chứ không yêu cầu phải mở toàn bộ mã nguồn hay phải mua giấy phép thương mại như trước kia.

Qt Creator là một IDE rất được các lập trình viên ngày nay ưa chuộng vì từ ngày Microsoft hết mặn mà với MFC và chuyển sang .NET thì các lập trình viên đang dần chuyển sang Qt. Qt đã được Nokia mua lại từ năm 2008 vì vậy hiện giờ cộng động Qt ngày càng lớn mạnh và gia tăng rất nhanh. Qt hỗ trợ rất mạnh trong lập trình giao diện, tương tác với Database, Graphics…

Điểm nổi bật của Qt Creator bao gồm:

- Cross-platform, nó có thể chạy trên mọi hệ điều hành: Mac, Linux, Windows.
- Có text-editor đẹp.
- Hỗ trợ vim editor (vim editor là một advance editor và có thể lập trình được các key, và xài hotkey rất nhanh).
- Miễn phí! Cũng giống như VC++ và CB chúng ta không phải trả bất cứ một khoản phí nào để download và sử dụng nó.

II. Tải về:

Click vào link sau đây http://qt-project.org/downloads và tải về bản thích hợp với hệ điều hành và chương trình dịch.

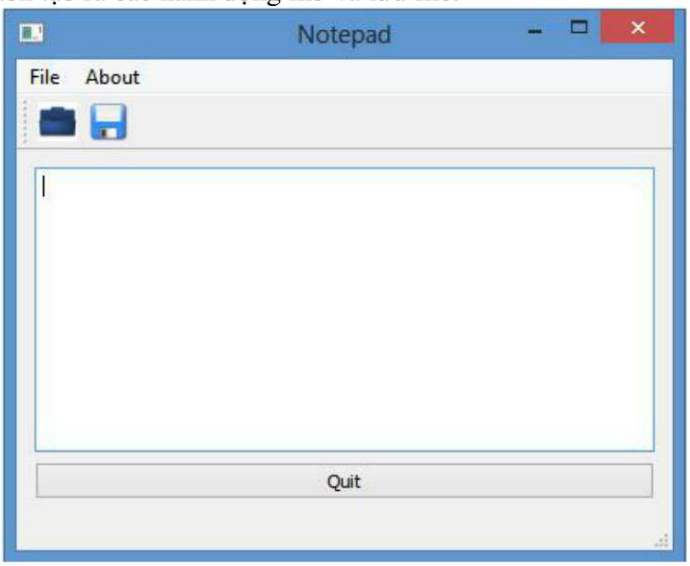
III. Cài đặt:

Cài đặt Qt Creator rất dễ dàng. Chúng ta chỉ cần chọn “Next” và chờ đợi cho tới khi nó chạy xong.

IV. Chạy Qt Creator và tạo một ứng dụng đơn giản.

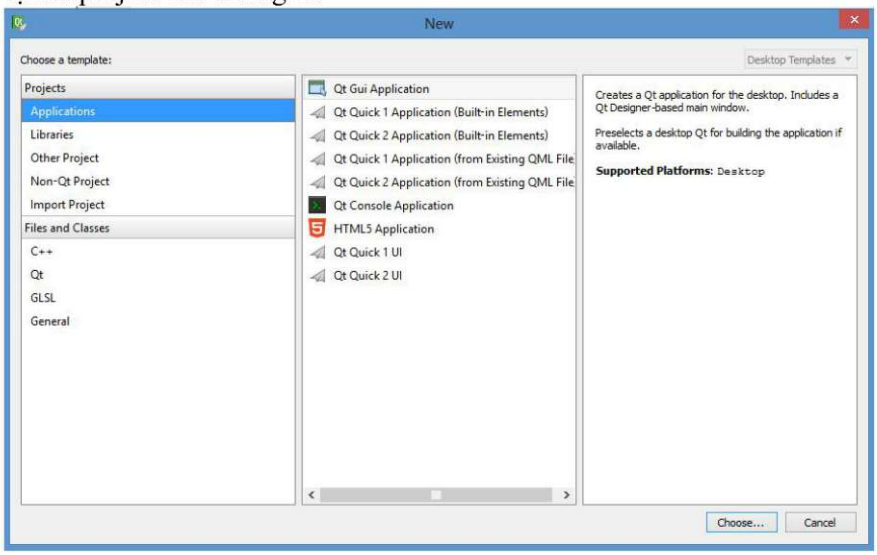
Chúng ta sẽ tìm hiểu những kiến thức cơ bản về Qt bằng cách tạo một ứng dụng Notepad đơn giản sử dụng C ++ và các mô-đun Widget Qt. Chúng ta sử dụng Qt Creator IDE và Qt Designer để tạo ra những dòng lệnh.

Đầu tiên chúng ta sử dụng Qt Creator để tạo ra một project với các file cần thiết. Sau đó, sử dụng Qt Designer để chỉnh sửa các file giao diện người dùng để tạo một ứng dụng chỉnh sửa văn bản và một nút nhấn trong một cửa sổ trên màn hình, một thanh công cụ, một widget dock. Đó là một ứng dụng Qt đơn giản. Cuối cùng, chúng ta thêm tương tác người dùng cho ứng dụng bằng cách tạo ra các hành động mở và lưu file.

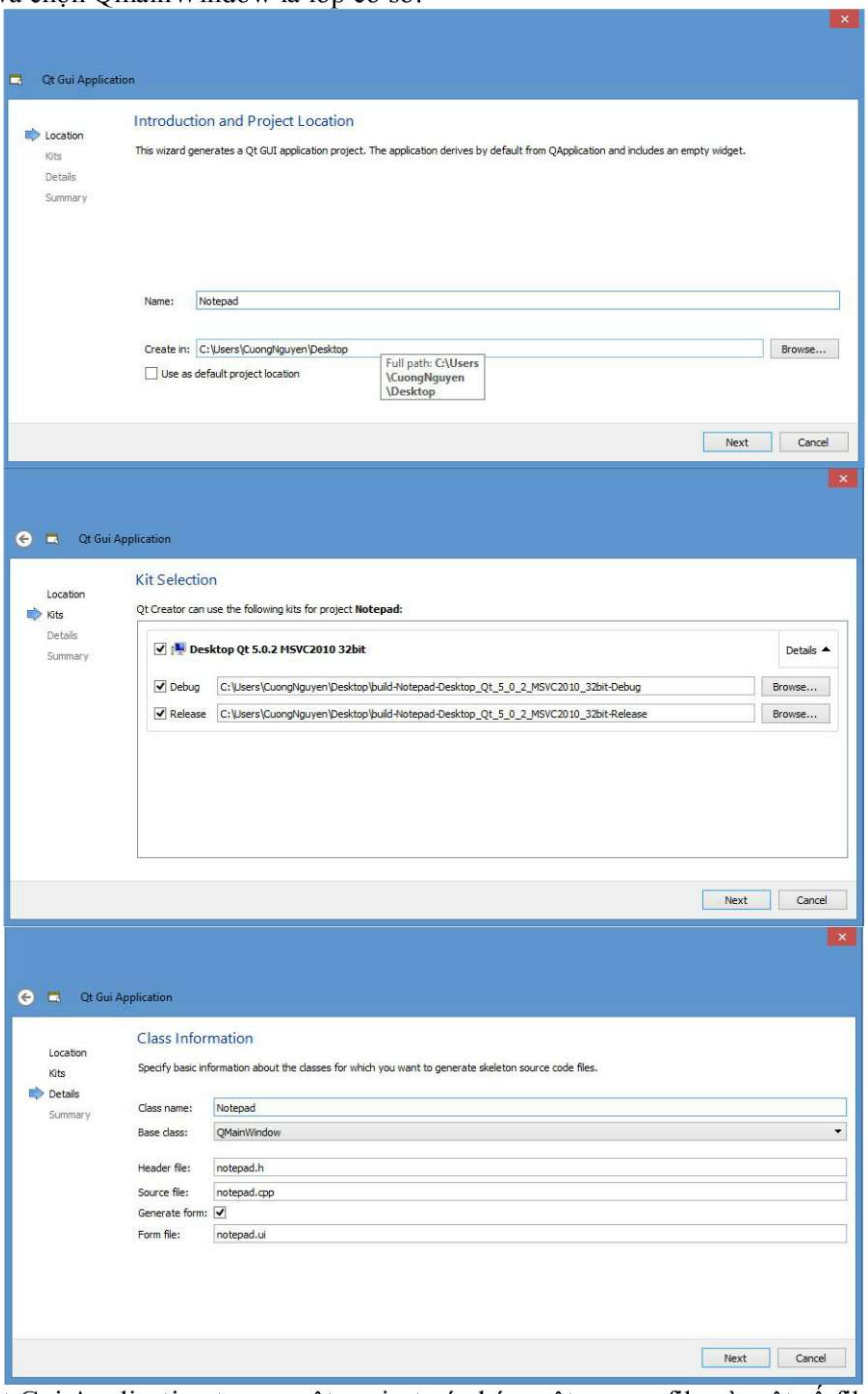


1. Tạo project Notepad

Quá trình tạo một project mới trong Qt Creator được hỗ trợ bởi một wizard (trình thuật sĩ) hướng dẫn bạn từng bước. Wizard sẽ nhắc ta nhập các cài đặt cần thiết cho từng loại project cụ thể và tạo ra project cho chúng ta.



Để tạo project Notepad, chọn File> New File or Project> Applications> Qt Gui Application> Choose, và làm theo hướng dẫn của wizard. Trong hộp thoại Class Information, gõ tên lớp là Notepad và chọn QmainWindow là lớp cơ sở.



Wizard Qt Gui Application tạo ra một project có chứa một source file và một số file xác định một giao diện người dùng (Notepad widget):

notepad.pro – project file.
main.cpp - file nguồn chính cho ứng dụng.
notepad.cpp - file nguồn của lớp notepad của Notepad widget.
notepad.h - file header của lớp notepad cho Notepad widget.
notepad.ui - giao diện người dùng cho Notepad widget.

2. Source File

Wizard tạo ra đoạn mã sau vào file main.cpp:

```
#include "notepad.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    Notepad w;
    w.show();

    return a.exec();
}
```

Hai dòng đầu tiên khai báo file header cho Notepad widget và QApplication.
Dòng 4 định nghĩa hàm main theo C + + cho ứng dụng.
Dòng 6 tạo ra một đối tượng QApplication. Đối tượng này quản lý ứng dụng và cần thiết để chạy bất kỳ chương trình Qt nào sử dụng Qt Widgets. Nó khởi tạo một đối tượng ứng dụng với argc tham số chạy trong argv. (Đối với các ứng dụng giao diện mà không sử dụng Qt Widget, ta có thể sử dụng QGuiApplication thay thế.)
Dòng 7 tạo ra đối tượng Notepad. Đây là đối tượng mà wizard tạo ra các lớp và các tập tin giao diện. Giao diện người dùng chứa các yếu tố thị giác được gọi là các widget (vật dụng) trong Qt. Ví dụ là các widget edit text, thanh cuộn, nhãn, và các nút radio. Một widget cũng có thể chứa các widget khác, một hộp thoại hoặc một cửa sổ ứng dụng chính chẳng hạn.
Dòng 8 hiển thị widget Notepad trên màn hình trong một cửa sổ riêng. Vì các widget cũng có chức năng chứa (ví dụ một QMainWindow, trong đó có các thanh công cụ, menu, thanh trạng thái, và một vài widget khác), có thể hiển thị một widget duy nhất trong một cửa sổ riêng. Widget được mặc định là không nhìn thấy; hàm show() làm cho các widget hiện ra.
Dòng 10 giúp cho khi một ứng dụng Qt đang chạy, các sự kiện được tạo ra và gửi đến các widget của ứng dụng. Ví dụ của sự kiện là nháy chuột và nhấn phím.

3. Thiết kế một giao diện người dùng

Wizard tạo ra một định nghĩa giao diện người dùng trong định dạng XML, notepad.ui. Khi chúng ta mở file notepad.ui trong Qt Creator, nó sẽ tự động mở trong Qt Designer.
Khi build ứng dụng, Qt Creator chạy trình biên dịch giao diện người dùng (Qt User Interface Compiler - uic), đọc các file .ui và tạo ra một file header C++ tương ứng ui_notepad.h.

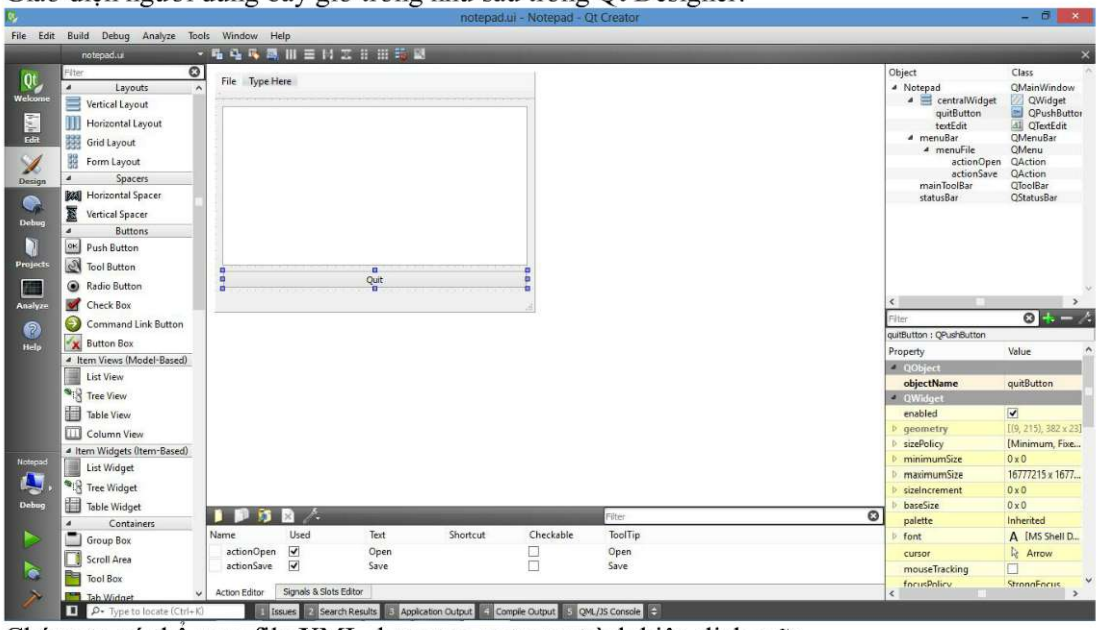
4.Sử dụng Qt Designer

Wizard tạo ra một ứng dụng sử dụng một QMainWindow . Nó có sự bố trí riêng mà chúng ta có thể thêm một thanh menu, widget dock, thanh công cụ, và một thanh trạng thái. Khu vực trung tâm có thể là bất kỳ loại widget nào.
Chúng ta hãy sử dụng Qt Designer để thêm một đối tượng QTextEditvà một đối tượng QPushButton vào cửa sổ chính. Khi chúng ta gõ văn bản trong widget text edit, nó nhận được sự kiện nhấn phím và phản ứng bằng cách đánh máy văn bản. Nút sẽ thoát khỏi ứng dụng Notepad khi nhấn. (có nghĩa là, nhấp chuột).

Để thêm các widget trong Qt Designer:
Trong chế độ Qt Creator Editor, bấm đúp vào file notepad.ui trong Projects để khởi động các file trong Qt Designer.
Kéo và thả các widget sau đây vào màn hình:

- Text Edit (QTextEdit)
- Push Button (QPushButton)

Kích đúp vào widget Push Button và gõ Quit.
Trong cửa sổ thuộc tính, thay đổi giá trị của objectName thành quitButton.
Nhấn Ctrl + A (hoặc Cmd + A) để chọn các widget và bấm vào Lay out Vertically (hoặc nhấn Ctrl + L) để bố trí theo chiều dọc (QVBoxLayout).
Nhấn Ctrl + S (hoặc Ctrl + S) để lưu thay đổi.
Giao diện người dùng bây giờ trông như sau trong Qt Designer:



Chúng ta có thể xem file XML được tạo ra trong trình biên dịch mã:

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Notepad</class>
 <widget class="QMainWindow" name="Notepad">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>400</width>
    <height>300</height>
   </rect>
  </property>
```

```
  <property name="windowTitle">
   <string>Notepad</string>
  </property>
  <widget class="QWidget" name="centralWidget">
   <widget class="QWidget" name="">
    <property name="geometry">
     <rect>
      <x>70</x>
      <y>0</y>
      <width>268</width>
      <height>235</height>
     </rect>
    </property>
    <layout class="QVBoxLayout" name="verticalLayout">
     <item>
      <widget class="QTextEdit" name="textEdit"/>
     </item>
     <item>
      <widget class="QPushButton" name="quitButton">
       <property name="text">
        <string>Quit</string>
       </property>
      </widget>
     </item>
    </layout>
   </widget>
  </widget>
  <widget class="QMenuBar" name="menuBar">
    ...
```

5. File Notepad Header

Wizard tạo ra một file header cho lớp Notepad có các #include, một constructor, một destructor, và các đối tượng Ui. File như sau:

```
#ifndef NOTEPAD_H
#define NOTEPAD_H

#include <QMainWindow>

namespace Ui {
class Notepad;
}

class Notepad : public QMainWindow
```

```
{
    Q_OBJECT

public:
    explicit Notepad(QWidget *parent = 0);
    ~Notepad();

private:
    Ui::Notepad *ui;
};

#endif // NOTEPAD_H
```

Dòng 4 include QMainWindow cung cấp một cửa sổ ứng dụng chính.
Dòng 6 định nghĩa lớp Notepad trong namespace Ui, đó là namespace chuẩn cho các lớp UI tạo ra từ các file .ui bằng uic tool.
Dòng 10 chứa macro Q_OBJECT. Nó xuất hiện đầu tiên trong định nghĩa lớp, và tuyên bố lớp của ta như một QObject . Đương nhiên, nó cũng phải kế thừa từ QObject . Một QObject có thêm một số khả năng so với một lớp C++ tiêu chuẩn. Đáng chú ý là, tên lớp và tên slot có thể được truy vấn tại run-time. Cũng có thể truy vấn các loại tham số của một slot và gọi nó.
Dòng 15 định nghĩa một constructor có một tham số mặc định được gọi là parent. Giá trị 0 chỉ ra rằng các widget không có parent (nó là một widget cấp cao nhất).
Dòng 16 định nghĩa một destructor ảo để giải phóng bộ nhớ đã được sử dụng bởi các đối tượng trong vòng đời của nó. Theo quy ước đặt tên của C++, hàm hủy có cùng tên với lớp mà nó được khai báo, bắt đầu bằng một dấu ngã (~). Trong QObject , hàm hủy ảo là để đảm bảo rằng hàm hủy của các lớp dẫn xuất được gọi đúng khi một đối tượng được xóa thông qua một con trỏ đến lớp cơ sở.
Dòng 19 khai báo một biến thành viên là một con trỏ đến lớp Notepad UI. Một biến thành viên gắn với một lớp cụ thể, và có thể tiếp cận tất cả các phương thức của nó.

6. Notepad Source File

Source file mà wizard tạo ra cho lớp Notepad như sau:

```
#include "notepad.h"
#include "ui_notepad.h"

Notepad::Notepad(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::Notepad)
{
    ui->setupUi(this);
}

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

Hai dòng đầu tiên include các file header lớp Notepad đã được tạo ra bởi wizard và các file header UI đã được tạo ra bởi uic tool.
Dòng 4 định nghĩa constructor Notepad và thiết lập file UI.
Dòng 5 gọi constructor QMainWindow là lớp cơ sở cho lớp Notepad.
Dòng 6 tạo ra một đối tượng lớp UI và gán nó cho thành viên ui.
Dòng 8 thiết lập UI.

Dòng 11 hủy ui.

7. Project File

Wizard tạo ra project file notepad.pro cho chúng ta như sau:

```
#-------------------------------------------------
#
# Project created by QtCreator 2013-04-21T21:08:50
#
#-------------------------------------------------

QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = Notepad
TEMPLATE = app


SOURCES += main.cpp\
        notepad.cpp

HEADERS  += notepad.h

FORMS    += notepad.ui
```

Project file chỉ định tên ứng dụng và mẫu qmake để sử dụng nhằm tạo ra project, cũng như source file, header file, và các UI file chứa trong project.

Chúng ta cũng có thể sử dụng tùy chọn qmake's –project để tạo file .pro. Mặc dù, trong trường hợp đó, phải nhớ thêm dòng QT += widgets để file được tạo ra liên kết lại Qt Widgets Module.

8. Thêm Tương tác người dùng

Bây giờ chúng ta có một giao diện người dùng, nhưng nó không thực sự làm được điều gì hữu ích, vì nó chỉ chứa một text edit và một nút nhấn, cũng như một số hàm tiêu chuẩn để thoát, thu nhỏ và mở rộng ứng dụng. Để làm ứng dụng trở nên hữu ích, chúng ta sẽ bổ sung thêm tương tác với người dùng vào nó. Đầu tiên, chúng ta sẽ thêm chức năng vào nút nhấn. Tiếp theo, chúng ta sẽ bổ sung thêm các hàm để tải một file vào text edit và để lưu nội dung của text edit thành một file.

Thêm Nút nhấn

Hầu hết các hệ thống điều khiển màn hình có cách tiêu chuẩn cho phép người dùng thoát khỏi ứng dụng. Tuy nhiên, trong ví dụ này chúng ta sử dụng hàm cơ bản này để minh họa làm thế nào để có thể thêm tương tác người dùng vào ứng dụng. Để làm điều này, chúng ta thêm một slot kết nối với Quit button.

Để thoát khỏi ứng dụng khi nút Quit được nhấn, ta sử dụng các tín hiệu Qt và cơ chế slot. Một tín hiệu được phát ra khi một sự kiện đặc biệt xảy ra và một slot là một hàm được gọi là để đáp ứng với một tín hiệu cụ thể. Widget Qt đã xác định trước tín hiệu và slot mà chúng ta có thể sử dụng trực tiếp từ Qt Designer.

Sử dụng Qt Designer để thêm một slot cho hàm thoát, kích chuột phải vào nút Quit để mở một trình đơn ngữ cảnh và sau đó chọn Go to slot> clicked().

Một slot private, on_quitButton_clicked(), được thêm vào file header của lớp Notepad widget, notepad.h và một hàm private, Notepad::on_quitButton_clicked(), được thêm vào

source file của lớp Notepad widget, notepad.cpp. Chúng ta chỉ cần viết mã để thực hiện hàm thoát trong source file.
Chúng ta hãy nhìn vào mã đã được sửa đổi trong header file, notepad.h:

```
#ifndef NOTEPAD_H
#define NOTEPAD_H

#include <QMainWindow>

namespace Ui {
class Notepad;
}

class Notepad : public QMainWindow
{
    Q_OBJECT

public:
    explicit Notepad(QWidget *parent = 0);
    ~Notepad();

private slots:
    void on_quitButton_clicked();

private:
    Ui::Notepad *ui;
};

#endif // NOTEPAD_H
```

Dòng 14 sử dụng tín hiệu của QT và cơ chế slot để ứng dụng thoát khi nút Quit được nhấn. Qt Designer sử dụng QMetaObject auto-connection facilities để kết nối để kết nối tín hiệu button's clicked() đến một slot trong lớp Notepad. uic tool tự động tạo ra mã trong hàm SetupUi() của hộp thoại để làm điều này, vì vậy Qt Designer chỉ cần khai báo, thực hiện một slot với một cái tên tiêu chuẩn.
Các mã tương ứng trong source file, notepad.cpp, như sau:

```
void Notepad::on_quitButton_clicked()
{

}
```

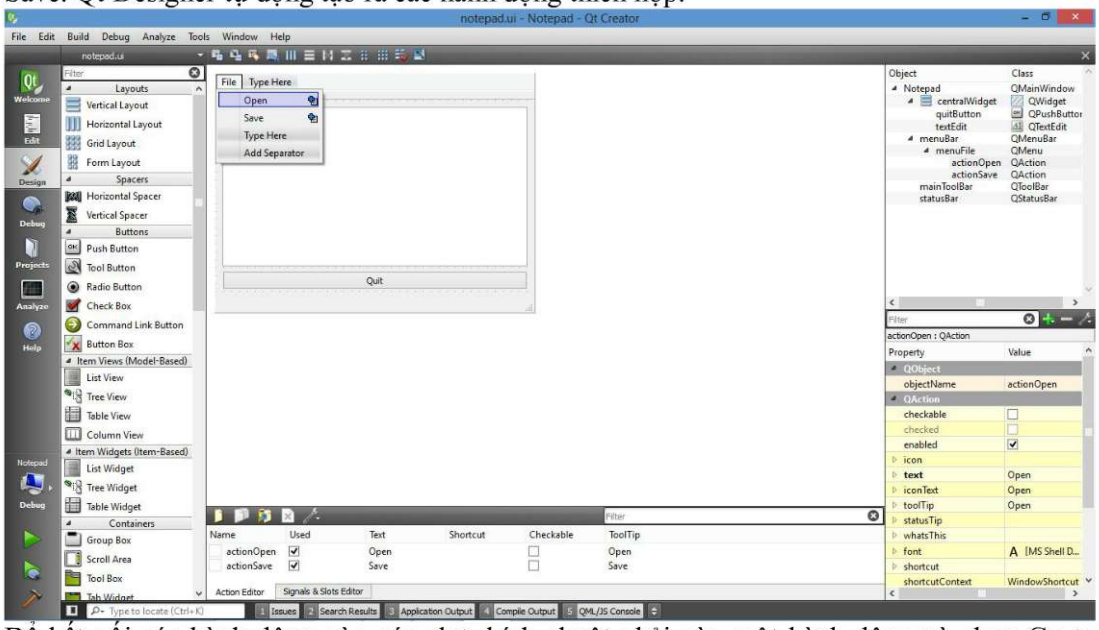
Mã xác định hàm private được thực hiện khi QPushButton phát tín hiệu clicked().
Bây giờ chúng ta bổ sung code để slot quit() của QApplication thoát Notepad:

```
void Notepad::on_quitButton_clicked()
{
    qApp->quit();
}
```

9. Thêm các item menu

Thông thường, trong một cửa sổ chính, cùng một slot nên được gọi bằng nhiều widget. Ví dụ như các item menu và các nút trên thanh công cụ. Để làm điều này dễ dàng hơn, Qt cung cấp QAction, có thể được trao cho một số widget, và đã được kết nối với một slot. Ví dụ, cả QMenu và QToolBar có thể tạo ra các item menu và các nút công cụ từ QAction .
Để tìm hiểu cách sử dụng nó với các tín hiệu và slot, chúng ta sẽ thêm các item menu để mở và lưu tài liệu và kết nối chúng vào các slot.

Như trước đây, chúng ta sử dụng Qt Designer để thêm các widget vào giao diện người dùng. Wizard tạo ra một ứng dụng với một QMenu widget, với chữ Type Here để giữ chỗ cho menu và các item menu. Kích đúp vào text để nhập tên cho menu File, item Open và item Save. Qt Designer tự động tạo ra các hành động thích hợp.



Để kết nối các hành động vào các slot, kích chuột phải vào một hành động và chọn Go to slot> triggered().

Các trường hợp của QAction được tạo ra với văn bản sẽ xuất hiện trên các widget mà chúng ta thêm chúng vào (trong trường hợp này là các item menu). Nếu chúng ta muốn thêm các hành động vào một thanh công cụ, chúng ta có thể xác định các biểu tượng cho chúng.

Mã sửa đổi trong notepad.ui bây giờ như sau:

```
<widget class="QMenuBar" name="menuBar">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>400</width>
    <height>22</height>
   </rect>
  </property>
  <widget class="QMenu" name="menuFile">
   <property name="title">
    <string>File</string>
   </property>
   <addaction name="actionOpen"/>
   <addaction name="actionSave"/>
```

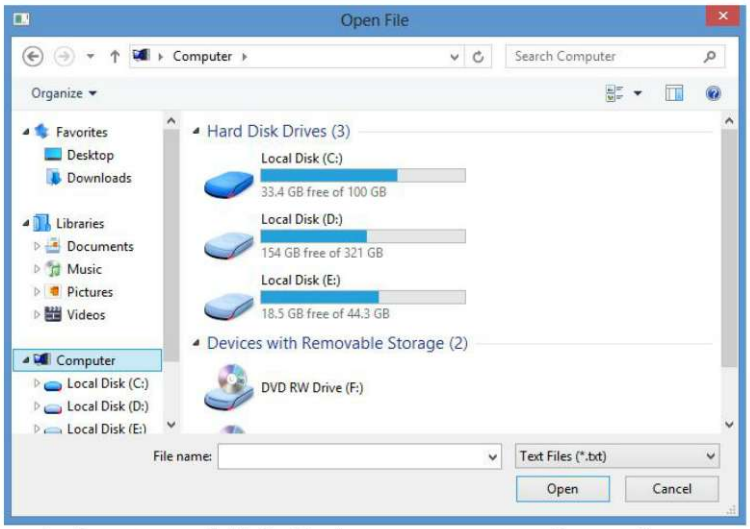
```
   </widget>
   <addaction name="menuFile"/>
  </widget>
  <widget class="QToolBar" name="mainToolBar">
   <attribute name="toolBarArea">
    <enum>TopToolBarArea</enum>
   </attribute>
   <attribute name="toolBarBreak">
    <bool>false</bool>
   </attribute>
  </widget>
  <widget class="QStatusBar" name="statusBar"/>
  <action name="actionOpen">
   <property name="text">
    <string>Open</string>
   </property>
  </action>
  <action name="actionSave">
   <property name="text">
    <string>Save</string>
   </property>
  </action>
 </widget>
```

Qt Designer thêm các private slot on_actionOpen_triggered() và on_actionSave_triggered() vào notepad.h và các hàm private Notepad::on_actionOpen_triggered() và Notepad::on_actionSave_triggered() vào notepad.cpp.

Trong các phần sau, chúng ta bổ sung code để tải và lưu các file. Khi một item menu được nhấp, item kích hoạt hành động, và slot tương ứng được gọi.

Mở file

Trong phần này, chúng ta thực hiện chức năng của slot on_actionOpen_triggered(). Bước đầu tiên là yêu cầu người sử dụng cho tên của tập tin để mở. Qt đi kèm với QfileDialog, là một hộp thoại từ đó người dùng có thể chọn một tập tin. Sự xuất hiện của hộp thoại phụ thuộc vào nền tảng máy tính mà bạn chạy ứng dụng. Hình ảnh sau đây cho thấy hộp thoại trên windows:



Chúng ta thêm code được tạo ra bởi Qt Designer trong notepad.cpp, như sau:

```
void Notepad::on_actionOpen_triggered()
{
    QString fileName = QFileDialog::getOpenFileName(this, tr("Open File"),
QString(),
                tr("Text Files (*.txt);;C++ Files (*.cpp *.h)"));

            if (!fileName.isEmpty()) {
                QFile file(fileName);
                if (!file.open(QIODevice::ReadOnly)) {
                    QMessageBox::critical(this, tr("Error"), tr("Could not
open file"));
                    return;
                }
                QTextStream in(&file);
                ui->textEdit->setText(in.readAll());
                file.close();
            }
}
```

Hàm static getOpenFileName() hiển thị một hộp thoại phương thức file. Nó trả về đường dẫn của file được chọn, hoặc một chuỗi rỗng nếu người dùng hủy bỏ hộp thoại.

Nếu chúng ta có một tên file, chúng ta mở file với open(), trả về đúng nếu file có thể mở được. Nếu file không thể mở được, chúng ta sử dụng QMessageBox để hiển thị một hộp thoại với một thông báo lỗi.

Việc đọc dữ liệu sử dụng lớp QTextStream, mà chứa object QFile. Hàm readAll() trả về nội dung của file như một QString. Nội dung sau đó có thể được hiển thị trong text edit. Sau đó chúng ta close() file để trả lại file cho hệ điều hành.

Bây giờ chúng ta sử dụng hàm tr(). Hàm này là cần thiết khi muốn viết ứng dụng trên nhiều ngôn ngữ.

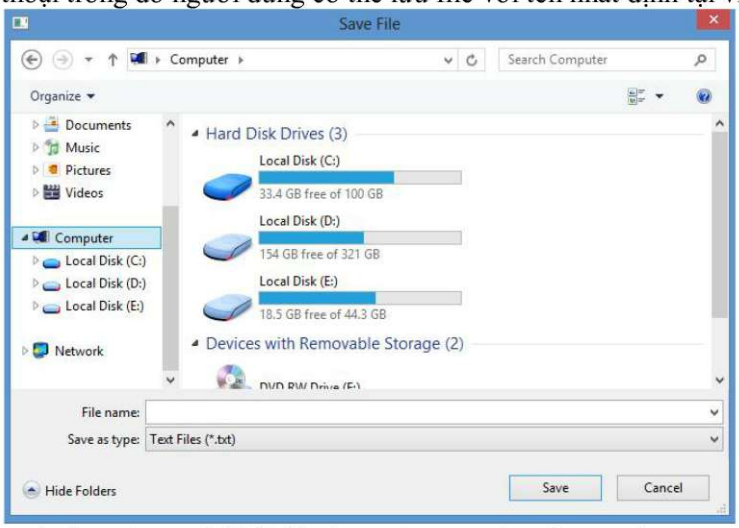
Để sử dụng QFileDialog, QFile, QmessageBox và QTextStream, thêm các dòng sau vào notepad.cpp:

```
#include <QFileDialog>
#include <QFile>
#include <QMessageBox>
```

```
#include <QTextStream>
```

Lưu file

Bây giờ, chúng ta sẽ chuyển sang slot on_actionSave_triggered(), sử dụng QFileDialog để tạo ra một hộp thoại trong đó người dùng có thể lưu file với tên nhất định tại vị trí nhất định.



Chúng ta thêm code được tạo ra bởi Qt Designer trong notepad.cpp, như sau:

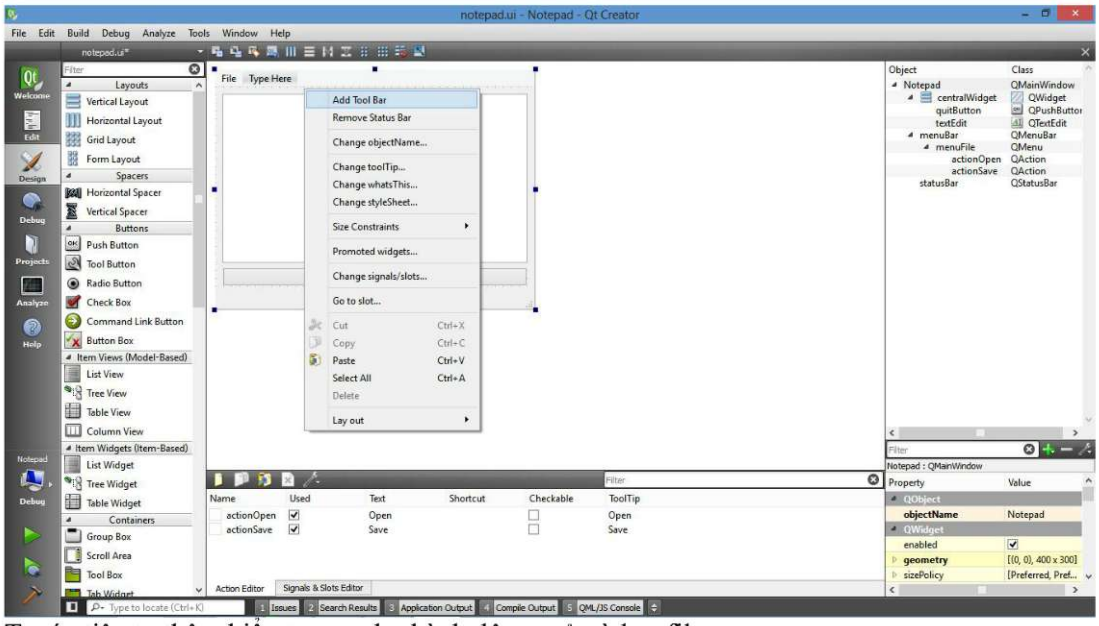
```
void Notepad::on_actionSave_triggered()
{
    QString fileName = QFileDialog::getSaveFileName(this, tr("Save File"),
QString(),
                tr("Text Files (*.txt);;C++ Files (*.cpp *.h)"));

                if (!fileName.isEmpty()) {
                    QFile file(fileName);
                    if (!file.open(QIODevice::WriteOnly)) {
                        // error message
                    } else {
                        QTextStream stream(&file);
                        stream << ui->textEdit->toPlainText();
                        stream.flush();
                        file.close();
                    }
                }
}
```

Khi chúng ta viết nội dung của text edit vào file, chúng ta lại sử dụng lớp QtextStream một lần nữa. QTextStream cũng có thể viết các QString vào file với toán tử <<.

10. Thêm thanh công cụ

Thanh công cụ được thêm vào một cửa sổ chính tương tự như thanh menu: Chọn Add Tool Bar từ menu ngữ cảnh. Ngoài ra, nếu có một thanh công cụ hiện có trong cửa sổ chính, chúng ta có thể nhấn vào mũi tên trên đầu bên phải của nó để tạo ra một thanh công cụ mới.



Trước tiên ta thêm biểu tượng cho hành động mở và lưu file:

Kích đúp chuột trái vào hành động ở Action Editor, cửa sổ Edit action hiện ra, ở mục Icon chúng ta chọn icon cho hành động. Ở đây chúng ta chọn từ file resources.

File resources .qrc được tạo ra bởi trình notepad của window bởi đoạn code sau:

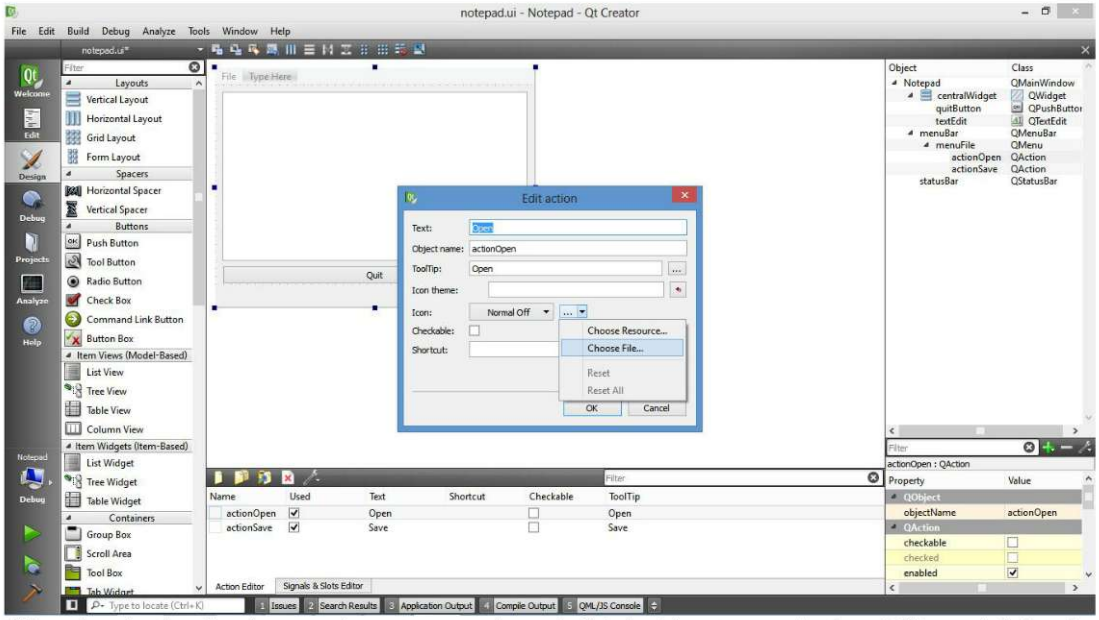
```
<!DOCTYPE RCC><RCC version="1.0">
<qresource>
    <file>images/open.png</file>
    <file>images/save.png</file>
</qresource>
</RCC>
```

Ta lưu lại file với tên notepad.qrc và tạo ra thư mục images rồi copy các icon cần thiết vào, ở đây là open và save.

Tiếp theo ta thêm dòng sau vào Notepad.pro để thêm file resources vào project.

```
RESOURCES     += notepad.qrc
```

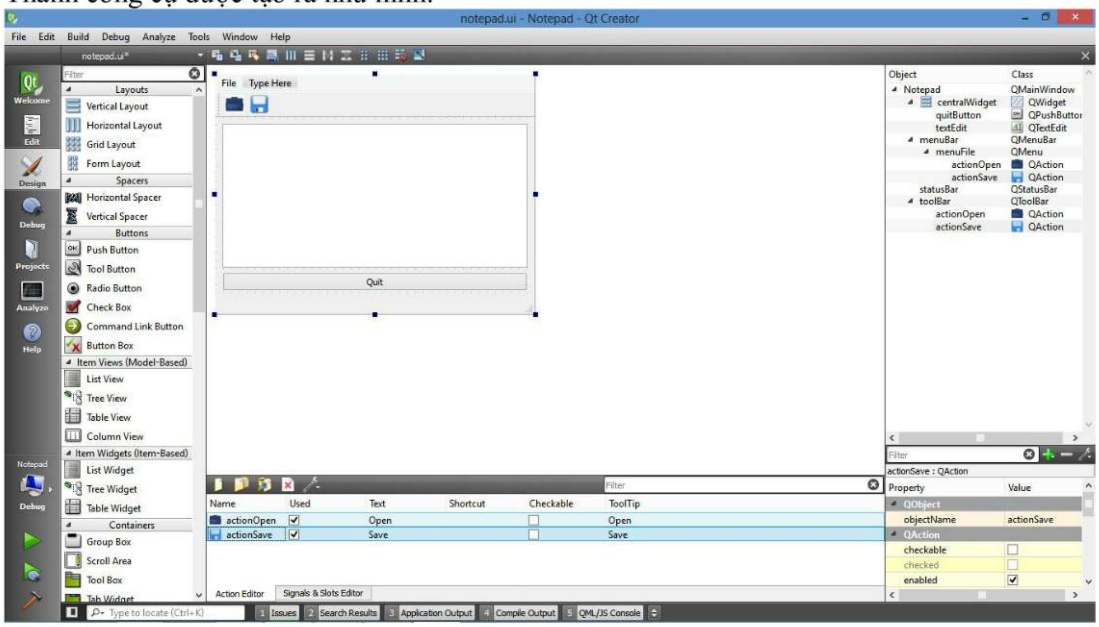
Và việc còn lại là chọn các icon thích hợp cho thanh công cụ.



Các nút của thanh công cụ được tạo ra như các hành động trong Action Editor và kéo vào thanh công cụ. Do các hành động có thể được đại diện bởi các menu và các nút trên thanh công cụ, chúng ta có thể di chuyển giữa các menu và thanh công cụ.

Để chia sẻ một hành động giữa một menu và thanh công cụ, kéo biểu tượng của nó từ action editor vào thanh công cụ.

Thanh công cụ được tạo ra như hình.



11. Thêm widget dock

Ta sẽ thêm vào một widget chứa thông tin về chương trình.

Ta bắt đầu bằng việc thêm một item About ở thanh menu tương tự như Open và Save rồi thêm hành động Notepad.

Để kết nối hành động Notepad vào slot, kích chuột phải vào actionNotepad trong Action Editor và chọn Go to slot> triggered().
Chúng ta thêm code được tạo ra bởi Qt Designer trong notepad.cpp, như sau:

```
void Notepad::on_actionNotepad_triggered()
{
    QMessageBox::about(this, tr("About"),
                       tr("<b>Notepad 1.0.0</b>"));
}
```

12. Build và chạy Notepad

Bây giờ chúng ta có tất cả các tập tin cần thiết, chọn Build> Build Project Notepad để build và chạy ứng dụng. Qt Creator sử dụng qmake và make để tạo ra một chương trình thực thi trong thư mục xác định trong thiết lập build của dự án và chạy nó.

13. Build và chạy từ dòng lệnh

Để build ứng dụng từ dòng lệnh, chuyển sang thư mục mà chúng ta có file .cpp của ứng dụng và thêm file project .pro được mô tả trước đây. Các lệnh sau giúp build ứng dụng:

```
qmake
make (or nmake on Windows)
```

Các lệnh tạo ra một chương trình thực thi trong thư mục project. Công cụ qmake đọc project file và tạo ra một Makefile với các hướng dẫn về cách build ứng dụng. Công cụ make (hoặc công cụ nmake) sau đó đọc Makefile và tạo ra chương trình thực thi nhị phân.