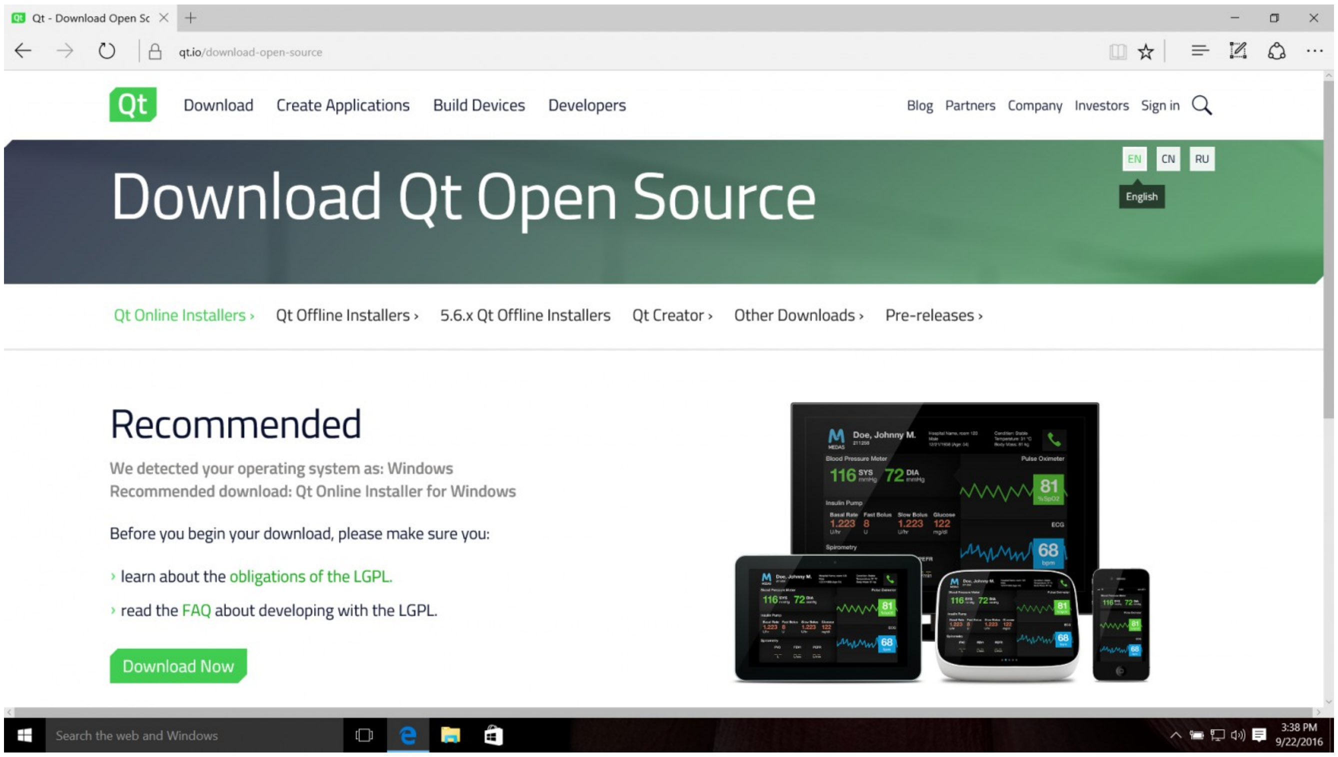
Getting Started With Qt (Desktop)

Qt is much more than just a cross-platform SDK - it's a technology strategy that lets you quickly and cost-effectively design, develop, deploy, and maintain software while delivering a seamless user experience across all devices. Let's get started with Qt.

How to setup Qt in Windows?



Qt - Download Open Sc

qt.io/download-open-source

Download | Create Applications | Build Devices | Developers | Blog | Partners | Company | Investors | Sign in

Download Qt Open Source

EN CN RU English

Qt Online Installers › | Qt Offline Installers › | 5.6.x Qt Offline Installers | Qt Creator › | Other Downloads › | Pre-releases ›

Recommended

We detected your operating system as: Windows
Recommended download: Qt Online Installer for Windows

Before you begin your download, please make sure you:

› learn about the obligations of the LGPL.

› read the FAQ about developing with the LGPL.

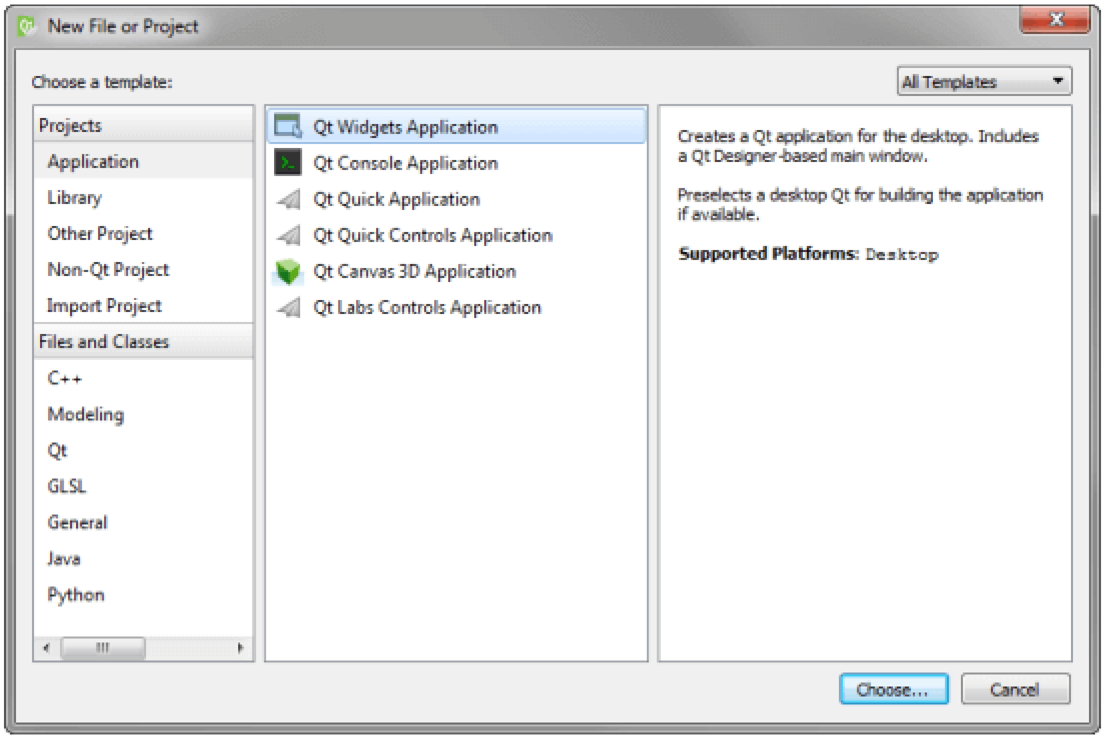
Download Now

Search the web and Windows

3:38 PM 9/22/2016

1. Download the open source Qt online installer from here.
2. Open the installer and select the package you want to select according to your visual studio edition along with Qt Creator.
3. Click next and start the installation.
4. After the installation completes, Qt is all setup.

Lets us create a basic "Text Finder" application to understand Qt :

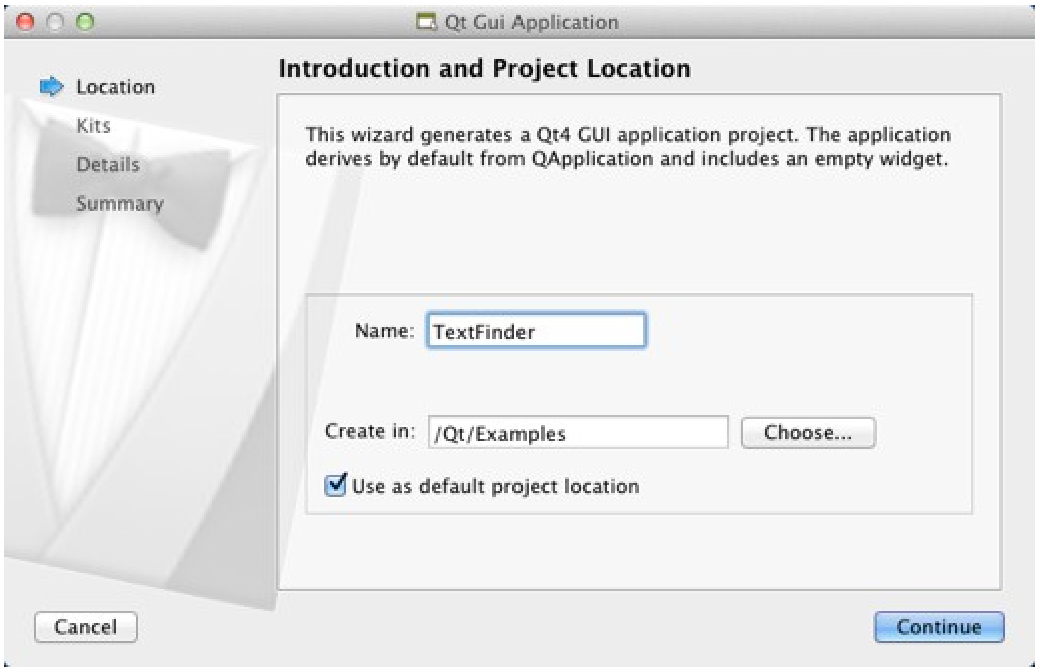


Creates a Qt application for the desktop. Includes a Qt Designer-based main window.

Preselects a desktop Qt for building the application if available.

Supported Platforms: `Desktop`

1. Select **File > New File or Project > Application > Qt Widgets Application > Choose**. The **Introduction and Project Location** dialog opens.



This wizard generates a Qt4 GUI application project. The application derives by default from QApplication and includes an empty widget.

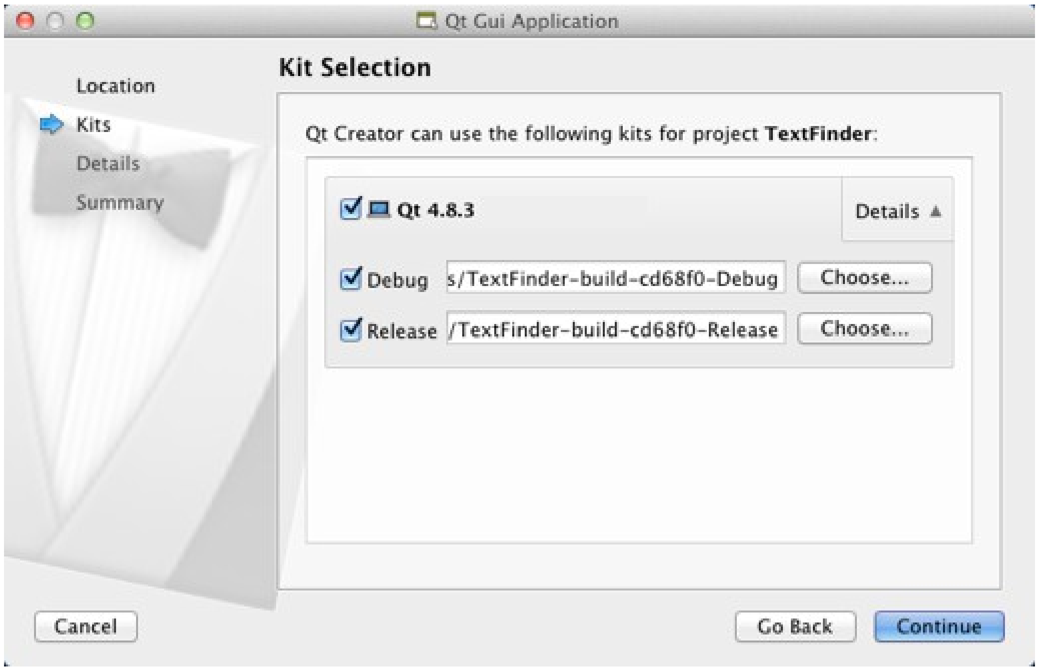
Name: TextFinder

Create in: /Qt/Examples

☑ Use as default project location

2.In the **Name** field, type **TextFinder**.

3.In the **Create in** field, enter the path for the project files. For example, C:\Qt\examples, and then click **Next** (on Windows and Linux). The **Kit Selection** dialog opens.



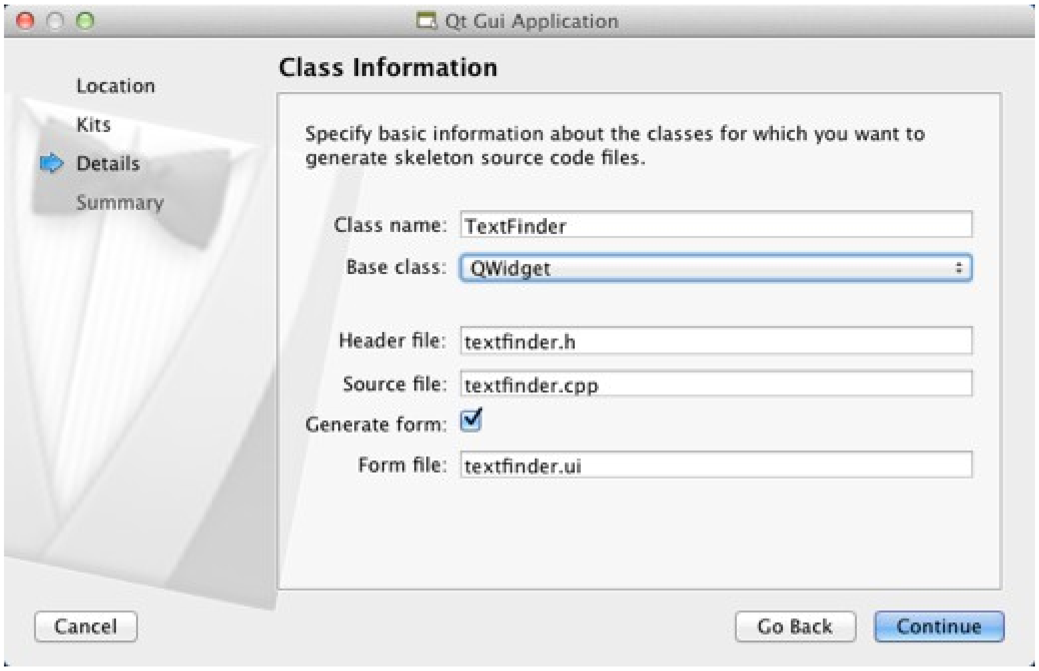
Qt Creator can use the following kits for project **TextFinder**:

☑ Qt 4.8.3

☑ Debug s/TextFinder-build-cd68f0-Debug

☑ Release /TextFinder-build-cd68f0-Release

4.Select build and run kits for your project, and click **Next** or **Continue**.
Note: If only one kit is specified in **Tools** > **Options** > **Build & Run** > **Kits** (on Windows and Linux), this dialog is skipped.
The **Class Information** dialog opens.



Specify basic information about the classes for which you want to generate skeleton source code files.

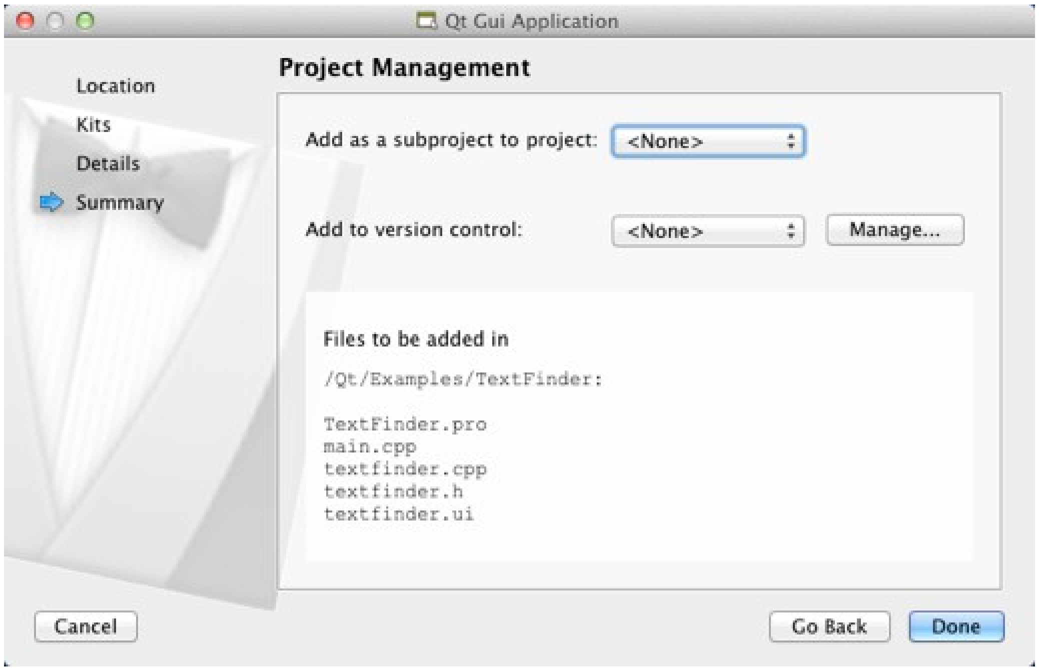
Class name: TextFinder
Base class: QWidget
Header file: textfinder.h
Source file: textfinder.cpp
Generate form: ☑
Form file: textfinder.ui

5.In the **Class name** field, type **TextFinder** as the class name.

6.In the **Base class** list, select **Qwidget** as the base class type.

Note: The **Header file**, **Source file** and **Form file** fields are automatically updated to match the name of the class.

7.Click **Next** or **Continue**.
The **Project Management** dialog opens.



Add as a subproject to project: <None>

Add to version control: <None>

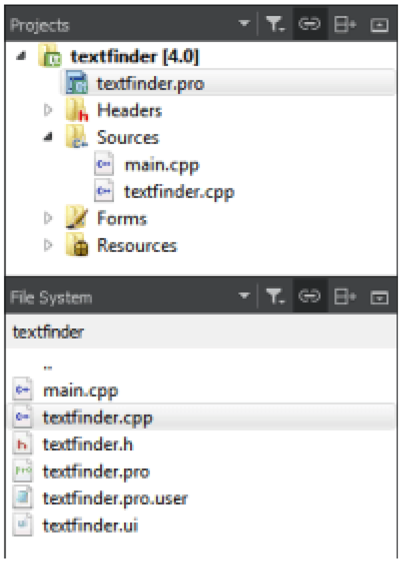
Files to be added in

```
/Qt/Examples/TextFinder:

TextFinder.pro
main.cpp
textfinder.cpp
textfinder.h
textfinder.ui
```

8. Review the project settings, and click **Finish** (on Windows and Linux) .

Note: The project opens in the **Edit** mode, and these instructions are hidden. To return to these I nstructions, open the **Help** mode.

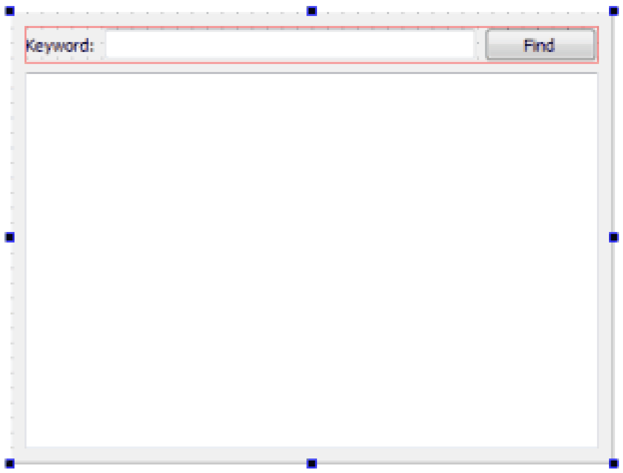


The .h and .cpp files come with the necessary boiler plate code. The .pro file is complete.

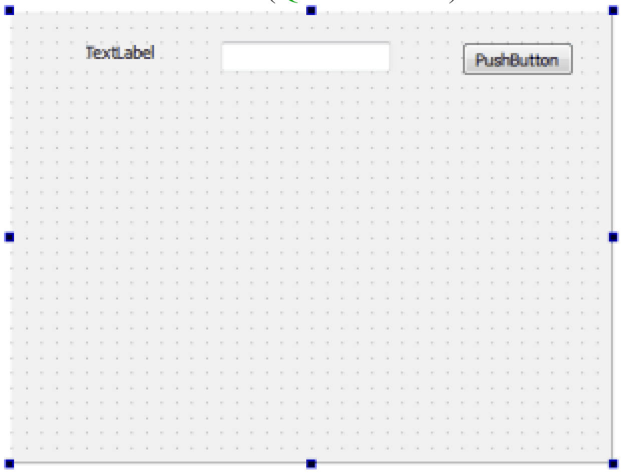
Filling in the Missing Pieces:

Begin by designing the user interface and then move on to filling in the missing code. Finally, add the find functionality.

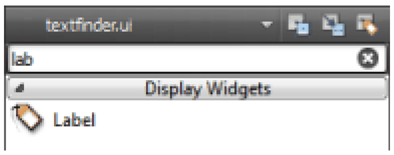
Designing the User Interface:



1. In the **Editor** mode, double-click the textfinder.ui file in the **Projects** view to launch the integrated Qt Designer.
2. Drag and drop the following widgets to the form:
   - **Label** (QLabel)
   - **Line Edit** (QLineEdit)
   - **Push Button** (QPushButton)



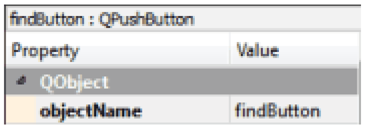
Note: To easily locate the widgets, use the search box at the top of the **Sidebar**. For example, to find the **Label** widget, start typing the word **label**.



3. Double-click the **Label** widget and enter the text **Keyword**.

4. Double-click the **Push Button** widget and enter the text **Find**.

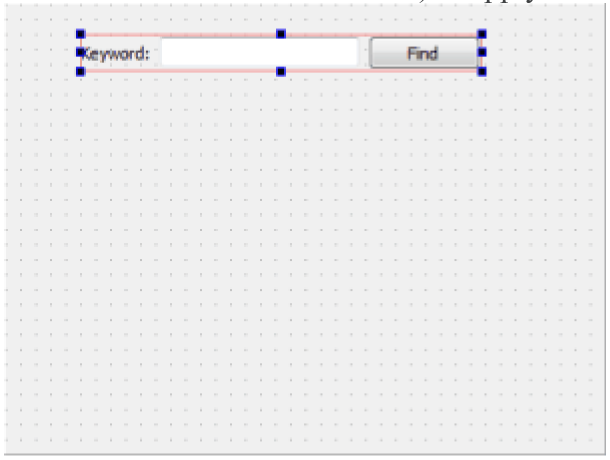
5. In the **Properties** pane, change the **objectName** to **findButton**.



findButton : QPushButton

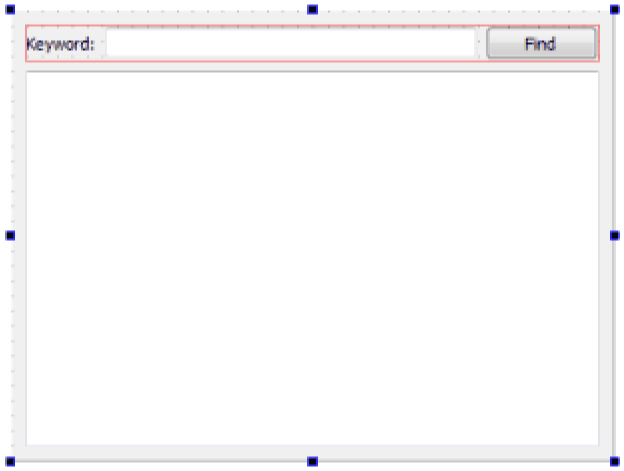
Property	Value
QObject	
objectName	findButton

6. Press **Ctrl+A** (or **Cmd+A**) to select the widgets and click **Lay out Horizontally** (or press **Ctrl+H** on Linux or Windows) to apply a horizontal layout (QHBoxLayout).



7. Drag and drop a **Text Edit** widget (QTextEdit) to the form.

8. Select the screen area and click **Lay out Vertically** (or press **Ctrl+L**) to apply a vertical layout (QVBoxLayout).



Applying the horizontal and vertical layouts ensures that the application UI scales to different screen sizes.

9. To call a find function when users press the **Find** button, you use the Qt signals and slots mechanism. A signal is emitted when a particular event occurs and a slot is a function that is

called in response to a particular signal. Qt widgets have predefined signals and slots that you can use directly from Qt Designer. To add a slot for the find function:

- Right-click the **Find** button to open a context-menu.
- Select **Go to Slot > clicked()**, and then select **OK**.

  A private slot, on_findButton_clicked(), is added to the header file, textfinder.h and a private function, TextFinder::on_findButton_clicked(), is added to the source file, textfinder.cpp.

10.Press **Ctrl+S** (or **Cmd+S**) to save your changes.

For more information about designing forms with Qt Designer, see the Qt Designer Manual.

Completing the Header File:

The textfinder.h file already has the necessary #includes, a constructor, a destructor, and theUiobject. You need to add a private function, loadTextFile(), to read and display the contents of the input text file in the QTextEdit.

1.In the **Projects** pane in the **Edit view**, double-click the textfinder.h file to open it for editing.

2.Add a private function to the private section, after the Ui::TextFinder pointer, as illustrated by the following code snippet:

```
private slots:
    void on_findButton_clicked();


private:
    Ui::TextFinder *ui;
    void loadTextFile();
```

Completing the Source File:

Now that the header file is complete, move on to the source file, textfinder.cpp.

1.In the **Projects** pane in the **Edit** view, double-click the textfinder.cpp file to open it for editing.

2.Add code to load a text file using QFile, read it with QTextStream, and then display it on textEdit with QTextEdit::setPlainText(). This is illustrated by the following code snippet:

```
void TextFinder::loadTextFile()
{
    QFile inputFile(":/input.txt");
    inputFile.open(QIODevice::ReadOnly);


    QTextStream in(&inputFile);
    QString line = in.readAll();
    inputFile.close();


    ui->textEdit->setPlainText(line);
    QTextCursor cursor = ui->textEdit->textCursor();
    cursor.movePosition(QTextCursor::Start, QTextCursor::MoveAnchor, 1);
}
```

3.To use QFile and QTextStream, add the following #includes to textfinder.cpp:

```
#include <QFile>
#include <QTextStream>
```

4.For the on_findButton_clicked() slot, add code to extract the search string and use the QTextEdit::find() function to look for the search string within the text file. This is illustrated by the following code snippet:

```
void TextFinder::on_findButton_clicked()
{
    QString searchString = ui->lineEdit->text();
    ui->textEdit->find(searchString, QTextDocument::FindWholeWords);
}
```

5.Once both of these functions are complete, add a line to call loadTextFile() in the constructor, as illustrated by the following code snippet:

```
TextFinder::TextFinder(QWidget *parent)
    : QWidget(parent), ui(new Ui::TextFinder)
{
    ui->setupUi(this);
    loadTextFile();
}
```

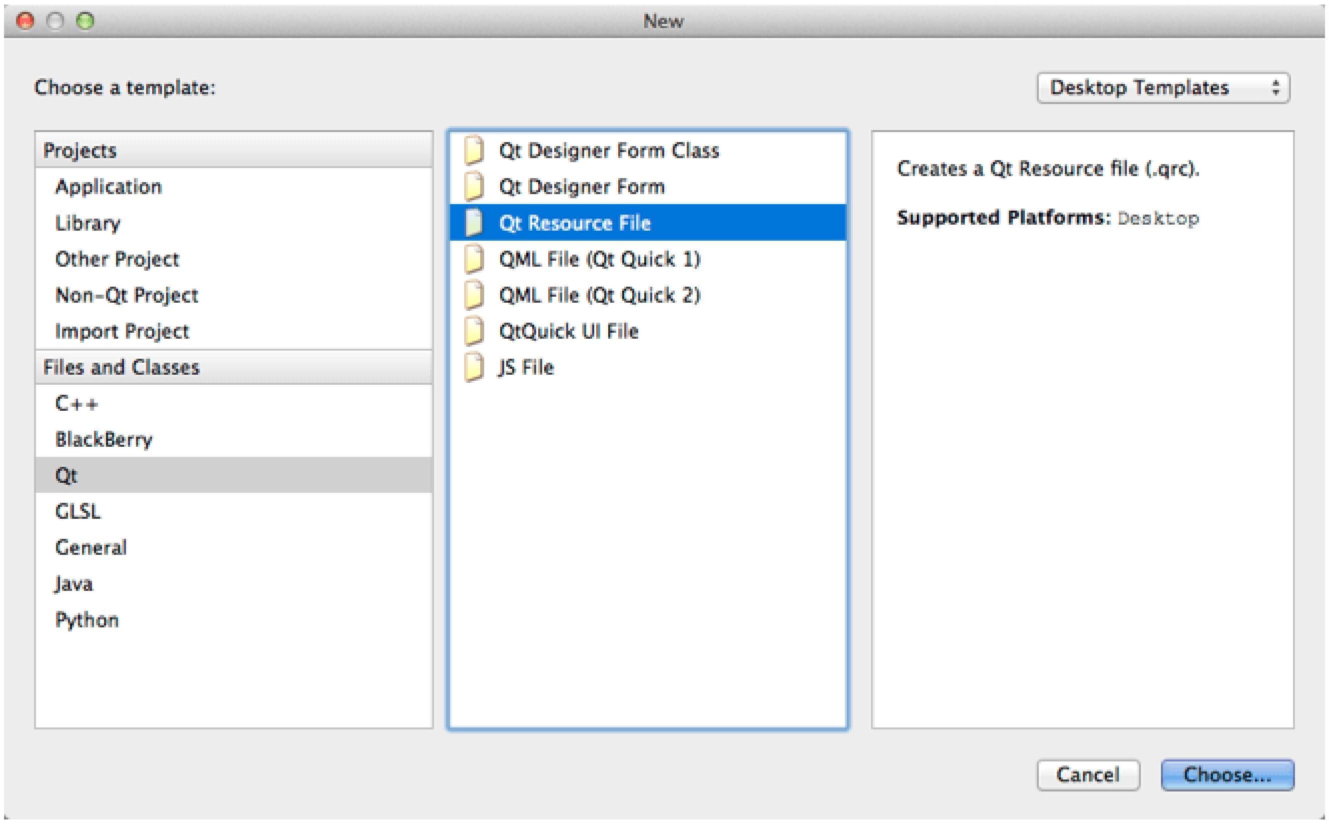
The on_findButton_clicked() slot is called automatically in the uic generated ui_textfinder.h file by this line of code:

```
QMetaObject::connectSlotsByName(TextFinder);
```

Creating a Resource File:

You need a resource file (.qrc) within which you embed the input text file. The input file can be any .txt file with a paragraph of text. Create a text file called input.txt and store it in the textfinder folder.
To add a resource file:

1.Select **File > New File or Project > Qt > Qt Resource File > Choose**.


New

Choose a template: Desktop Templates

Projects
- Application
- Library
- Other Project
- Non-Qt Project
- Import Project

Files and Classes
- C++
- BlackBerry
- Qt
- GLSL
- General
- Java
- Python

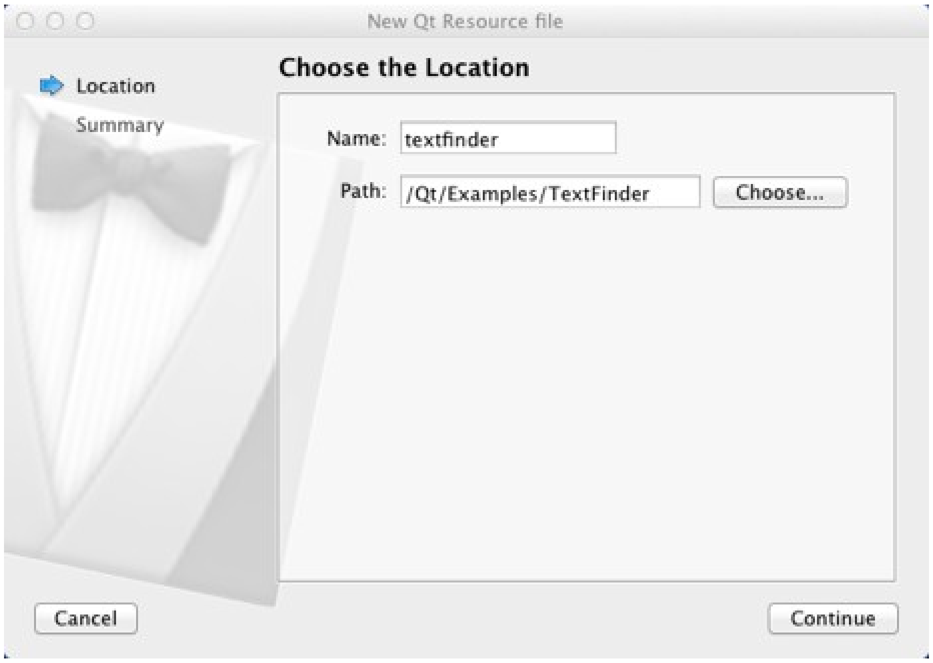
- Qt Designer Form Class
- Qt Designer Form
- Qt Resource File
- QML File (Qt Quick 1)
- QML File (Qt Quick 2)
- QtQuick UI File
- JS File

Creates a Qt Resource file (.qrc).

Supported Platforms: Desktop

Cancel | Choose...

The **Choose the Location** dialog opens.


New Qt Resource file

Location

Summary

Choose the Location

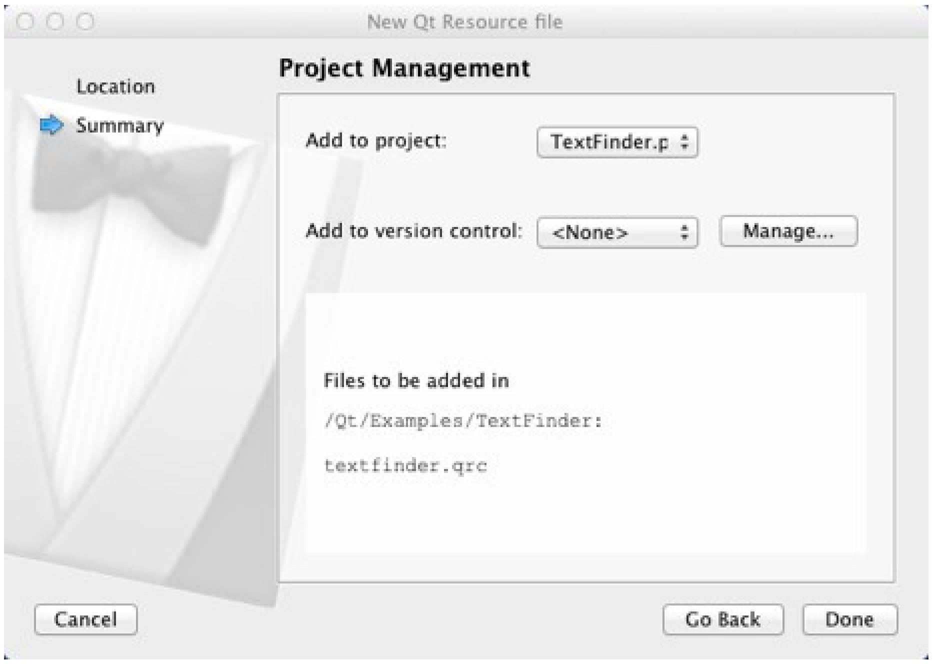
Name: textfinder

Path: /Qt/Examples/TextFinder | Choose...

Cancel | Continue

2.In the **Name** field, enter **textfinder**.

3.In the **Path** field, enter C:\Qt\examples\TextFinder, and click **Next** or **Continue**.
The **Project Management** dialog opens.


New Qt Resource file

Location
Summary

Project Management

Add to project: TextFinder.p

Add to version control: <None>

Files to be added in

/Qt/Examples/TextFinder:

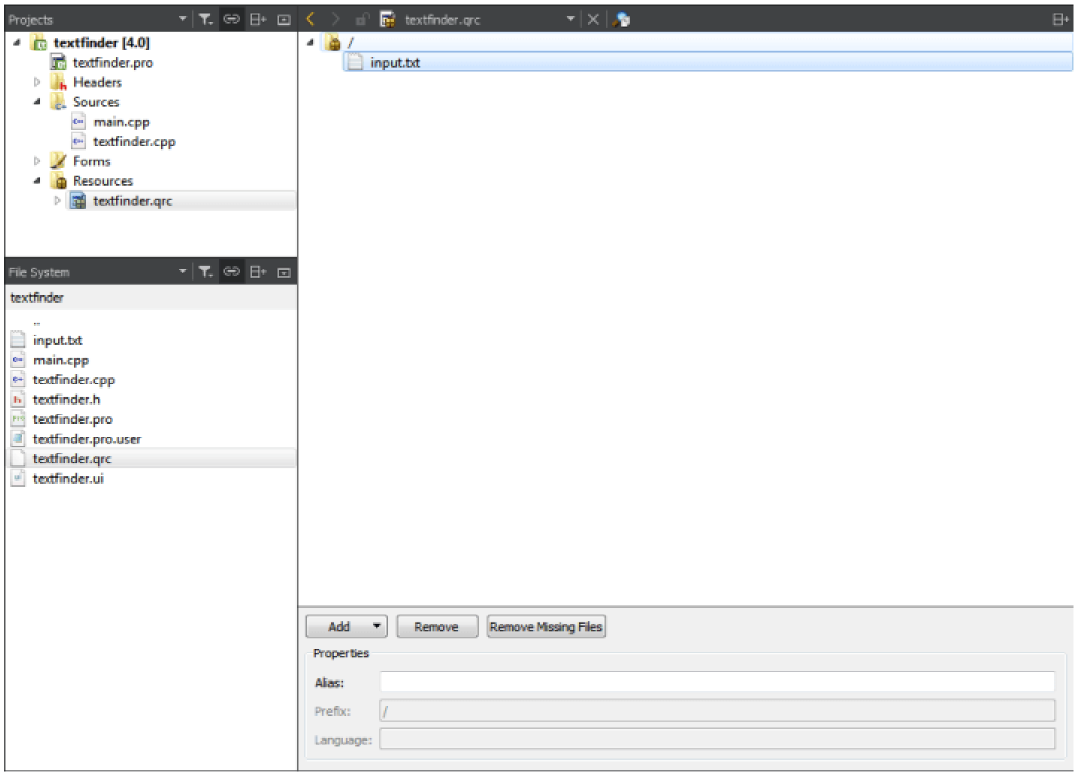
textfinder.qrc

4.In the **Add to project** field, select **TextFinder.pro** and click **Finish** or **Done** to open the file in the code editor.

5.Select **Add** > **Add Prefix**.

6.In the **Prefix** field, replace the default prefix with a slash (/).

7.Select **Add** > **Add Files**, to locate and add input.txt.



Alias:

Prefix: /

Language:

Compiling and Running Your Program:

Now that you have all the necessary files, run the qmake utility and then click the ▶ button to compile and run your program.
TextFinder is ready to be used.

What is the use of qmake in Qt ?

Qmake is a tool that helps simplify the build process for development project across different platforms. Qmake automates the generation of Makefiles so that only a few lines of information are needed to create each Makefile . Qmake generates a Makefile based on the information in a project file. In Qt , you just click on run qmake and it automatically generates the required makefiles for your project.

How does user interface files(.ui) work in Qt ?

Qt GUI toolkit has a *User Interface Compiler.* The uic reads an XML format user interface definition (.ui) file as generated by Qt Designer and creates a corresponding C++ header file. With the Qt's Integrated build tools , uic and qmake , code is generated for the corresponding ui elements.

These are the basics to get started with Qt. To know more go to http://doc.qt.io/.