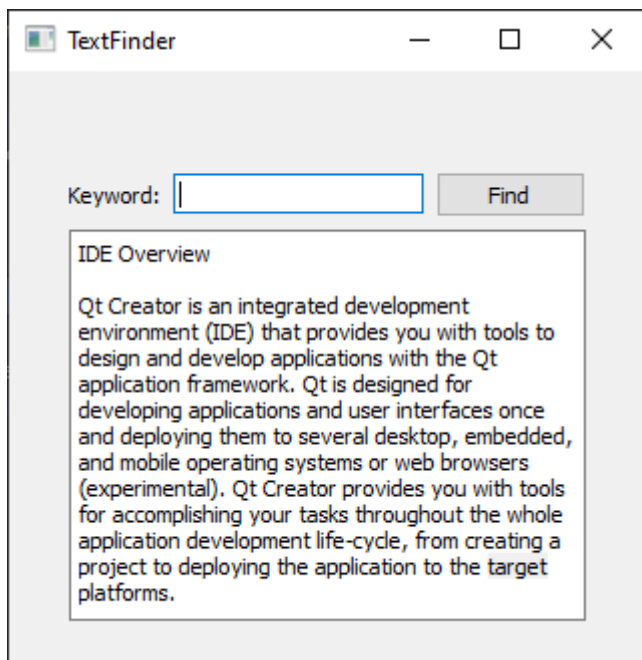


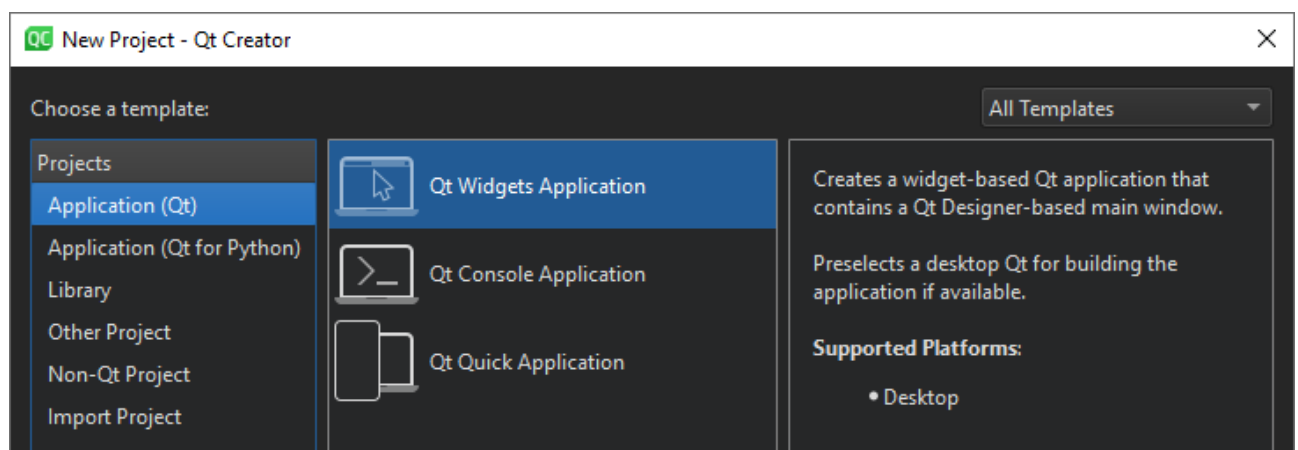
Creating a Qt Widget Based Application

This tutorial describes how to use Qt Creator to create a small Qt application, Text Finder. It is a simplified version of the Qt UI Tools [Text Finder Example](#). The application user interface is constructed from Qt widgets by using Qt Designer. The application logic is written in C++ by using the code editor.

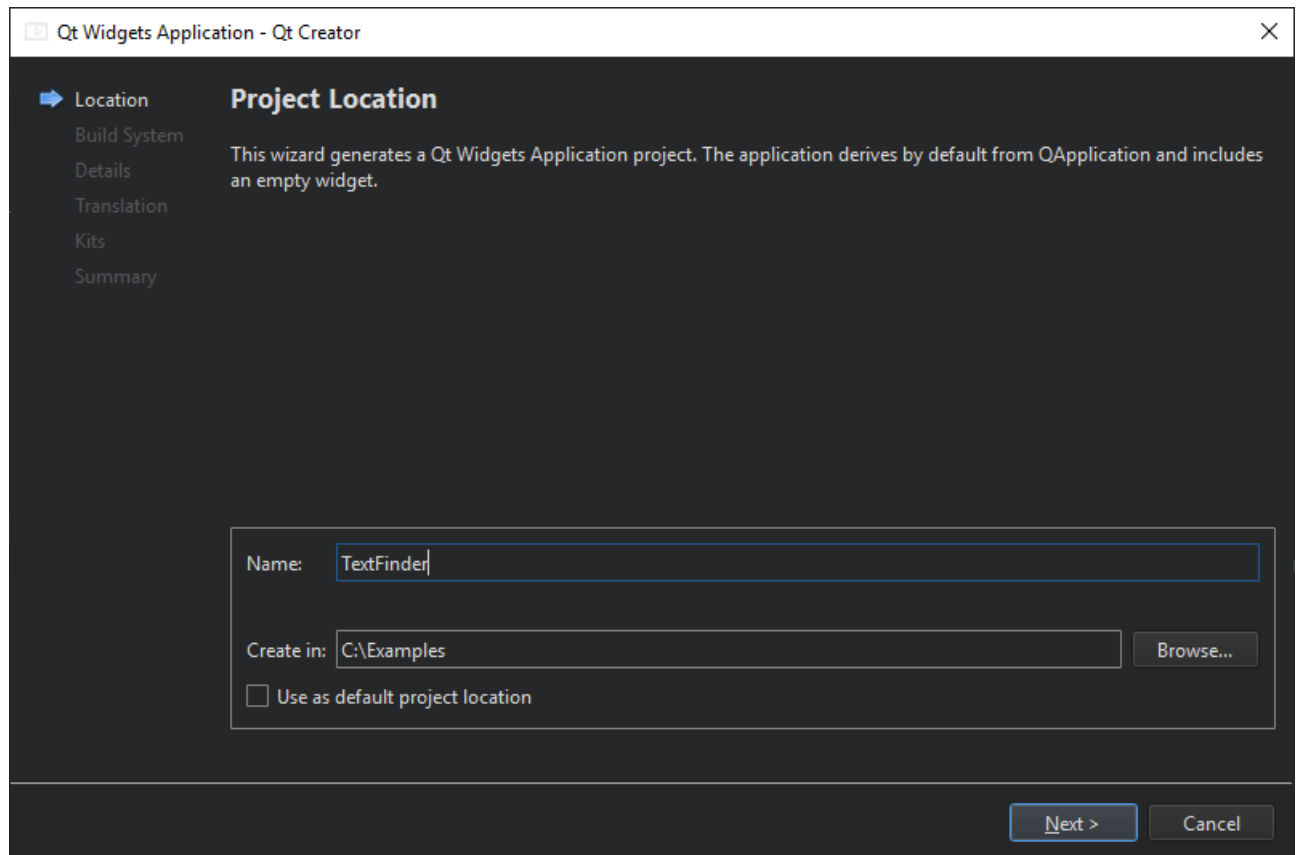


Creating the Text Finder Project

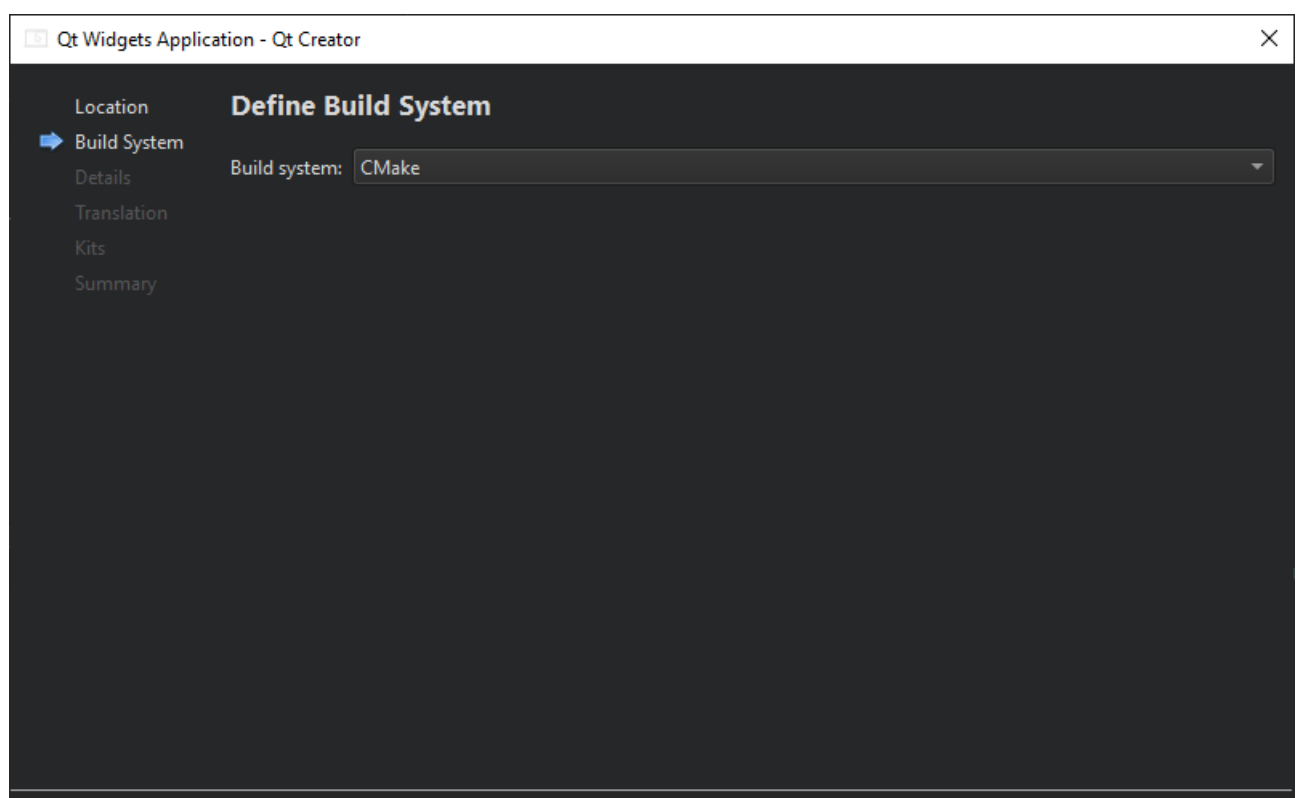
1. Select File > New Project > Application (Qt) > Qt Widgets Application > Choose.



The **Introduction and Project Location** dialog opens.

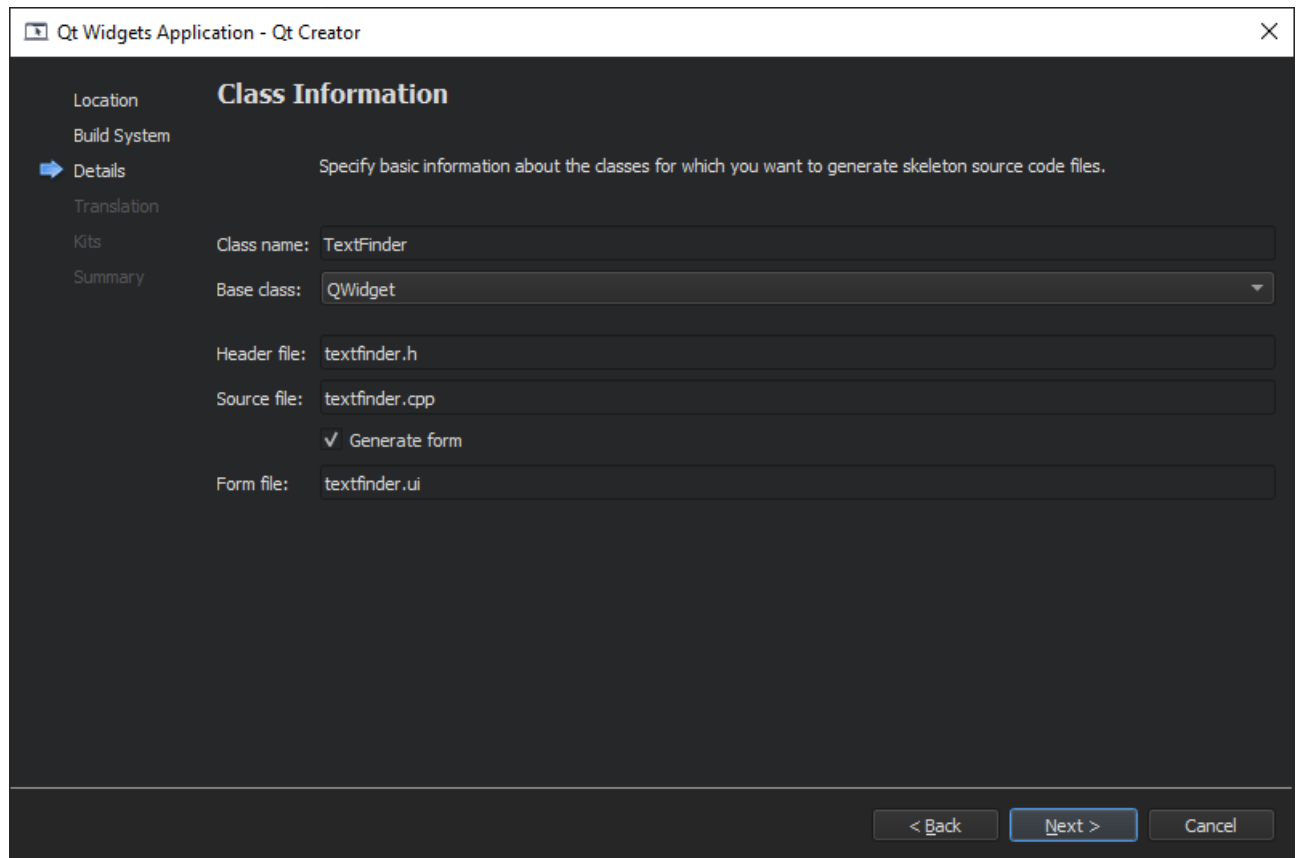


2. In the **Name** field, type **TextFinder**.
3. In the **Create in** field, enter the path for the project files. For example, **C : \Qt \examples**.
4. Select **Next** (on Windows and Linux) or **Continue** (on macOS) to open the **Define Build System** dialog.



5. In the **Build System** field, select **CMake** as the build system to use for building the project.

6. Select **Next** or **Continue** to open the **Class Information** dialog.

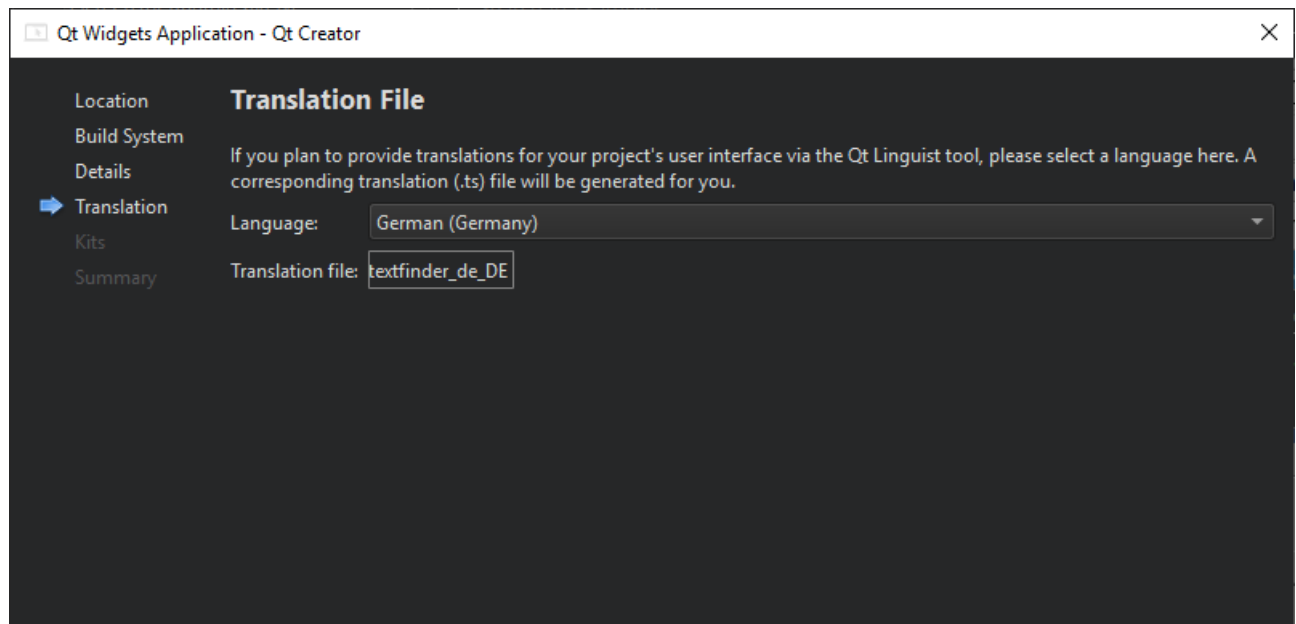


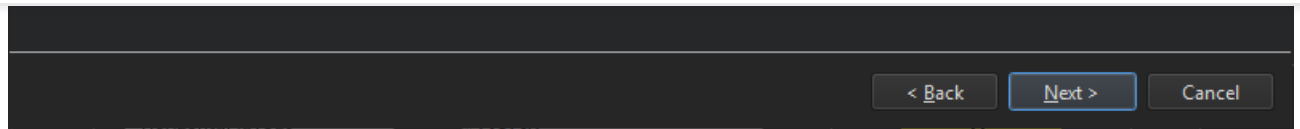
7. In the **Class name** field, type **TextFinder** as the class name.

8. In the **Base class** list, select **QWidget** as the base class type.

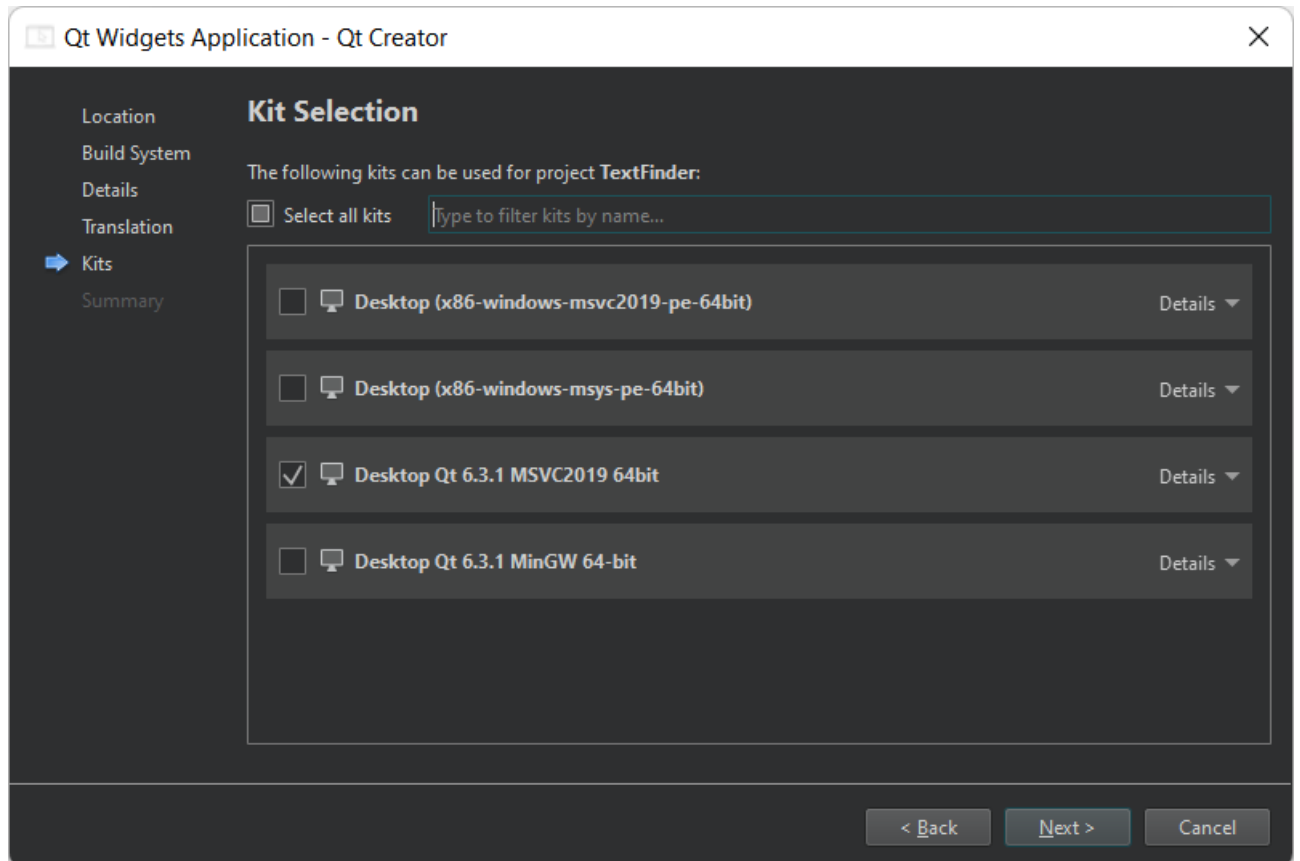
Note: The **Header file**, **Source file** and **Form file** fields are automatically updated to match the name of the class.

9. Select **Next** or **Continue** to open the **Translation File** dialog.

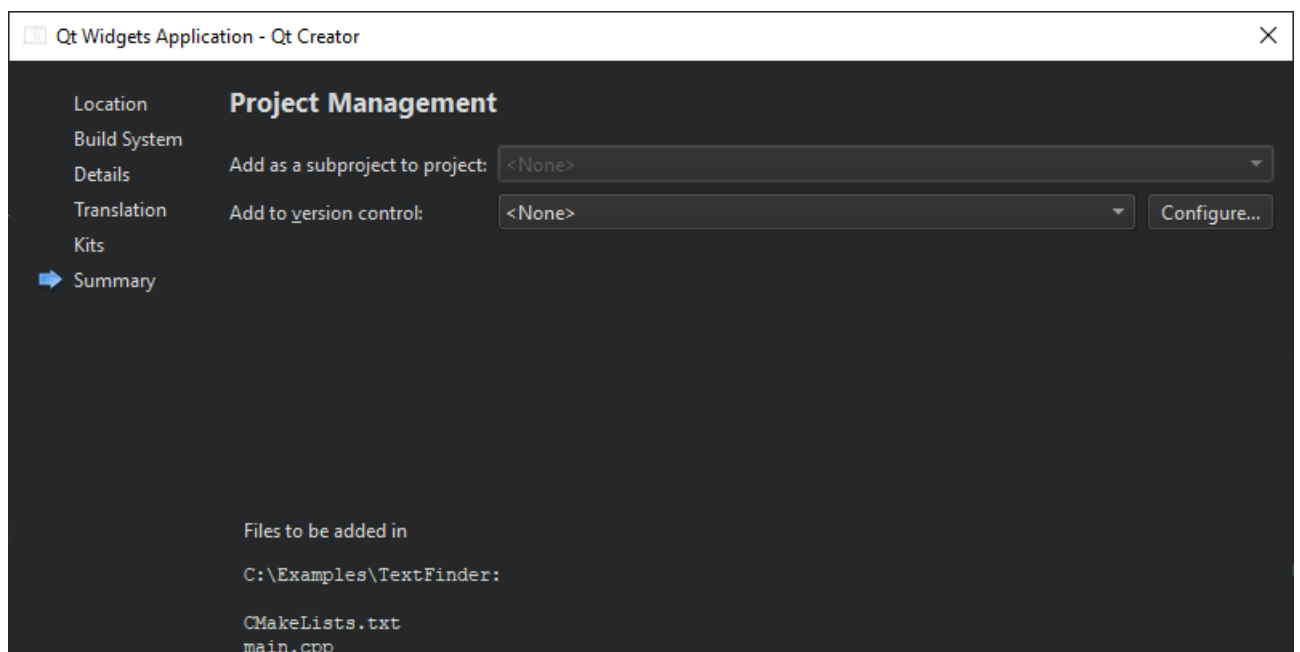




10. In the **Language** field, you can select a language that you plan to **translate** the application to. This sets up localization support for the application. You can add other languages later by editing the project file.
11. Select **Next** or **Continue** to open the **Kit Selection** dialog.



12. Select build and run **kits** for your project.
13. Select **Next** or **Continue** to open the **Project Management** dialog.



< Back

Finish

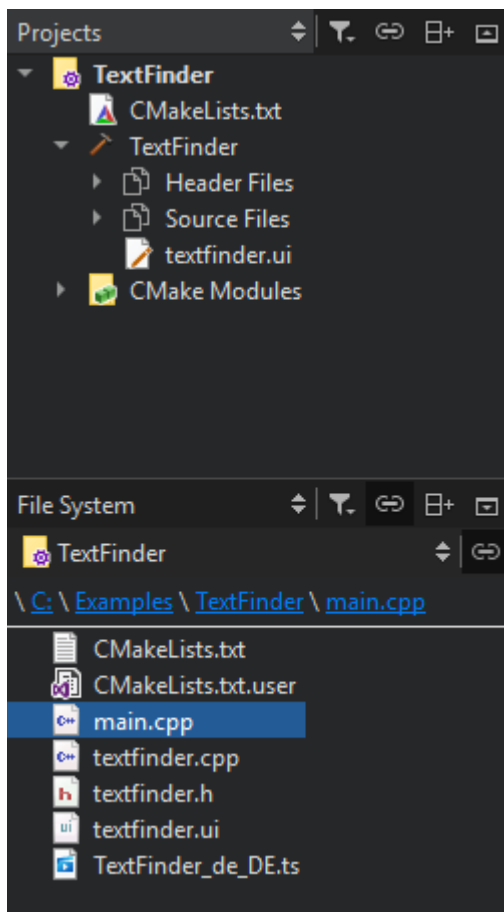
Cancel

14. Review the project settings, and select **Finish** (on Windows and Linux) or **Done** (on macOS) to create the project.

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

The TextFinder project now contains the following files:

- › main.cpp
- › textfinder.h
- › textfinder.cpp
- › textfinder.ui
- › CMakeLists.txt



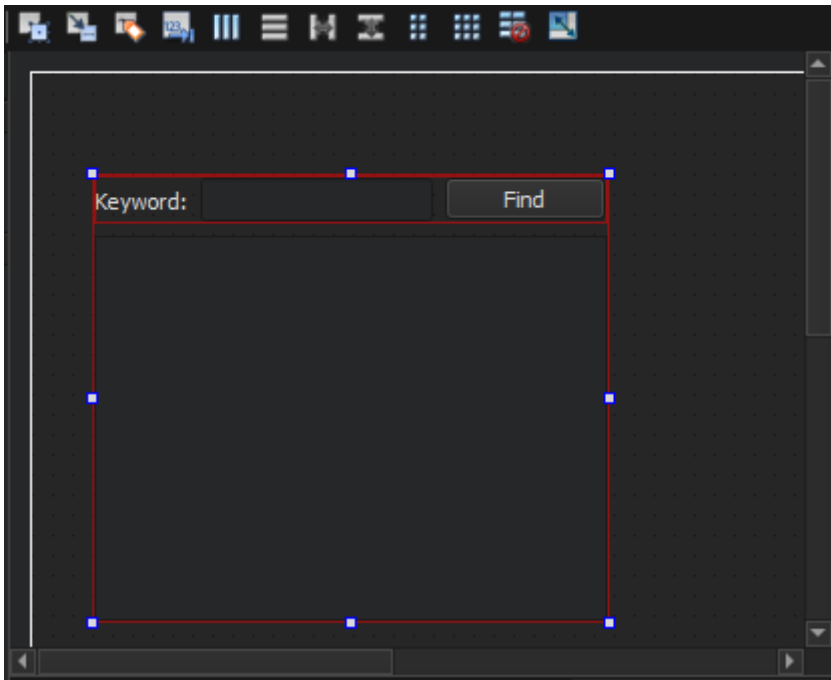
The .h and .cpp files come with the necessary boiler plate code.

If you selected CMake as the build system, Qt Creator created a CMakeLists.txt project file for you.

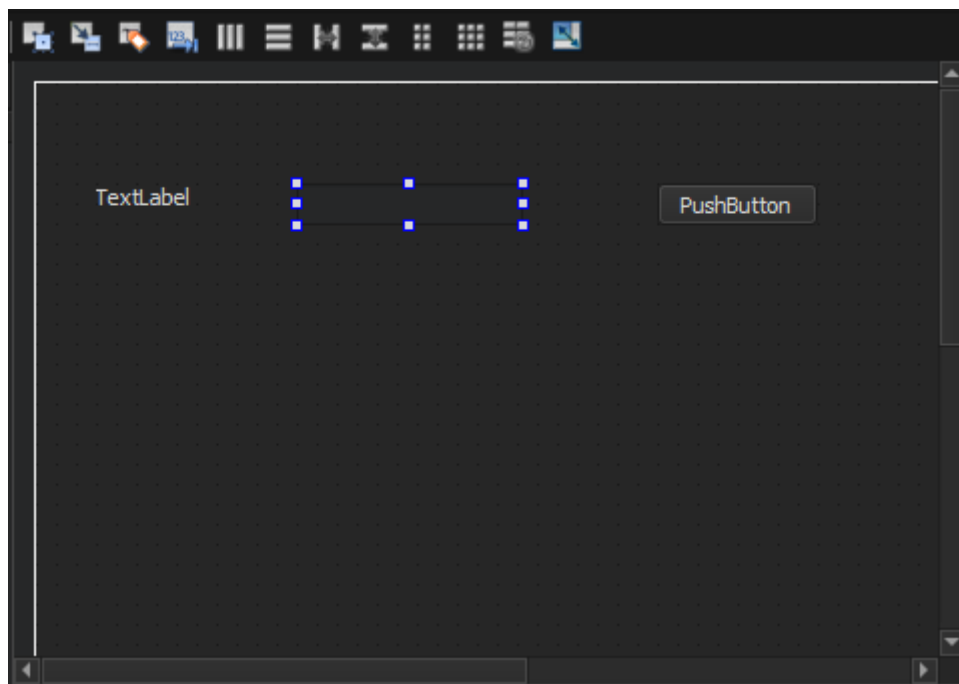
Filling in the Missing Pieces

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

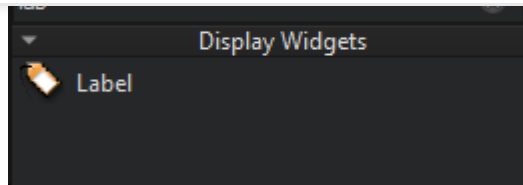
Designing the User Interface



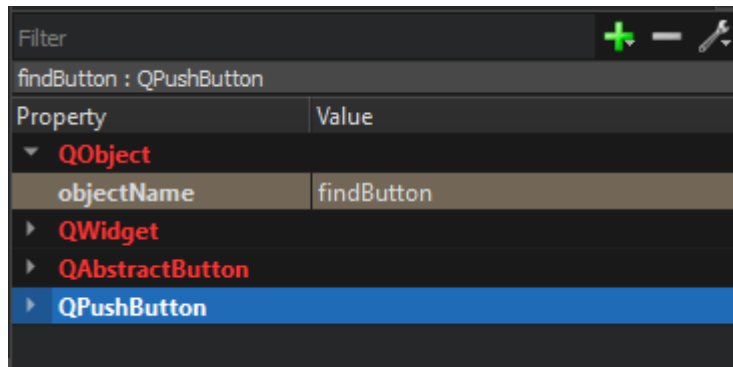
1. In the **Editor** mode, double-click the textfinder.ui file in the **Projects** view to launch the integrated Qt Designer.
2. Drag and drop the following widgets to the form:
 - › Label (QLabel)
 - › Line Edit (QLineEdit)
 - › Push Button (QPushButton)



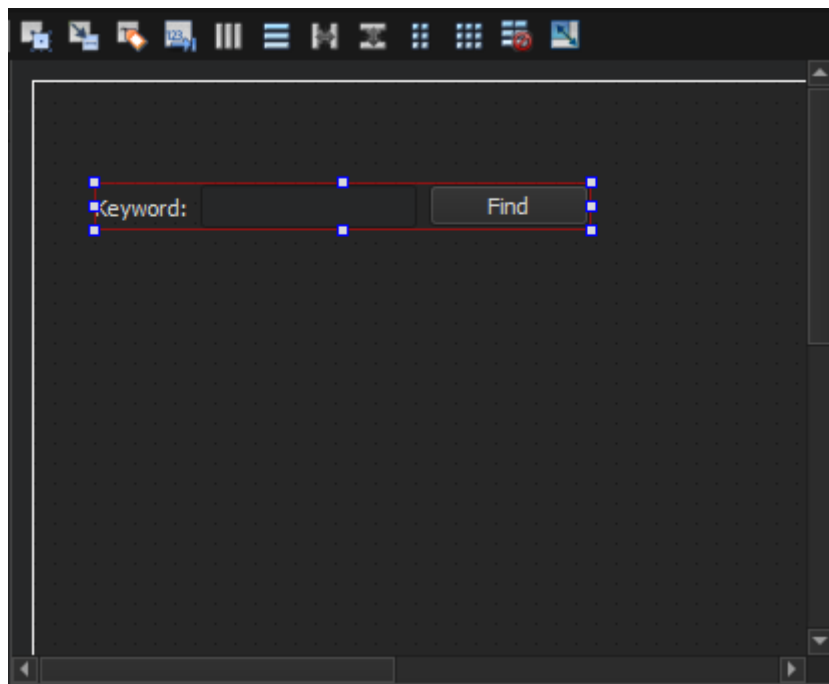
Note: To easily locate the widgets, use the search box at the top of the **Sidebar**. For example, to find the **Label** widget, start typing the word **label**.



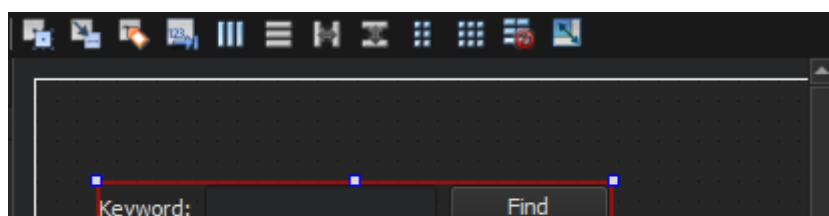
3. Double-click the **Label** widget and enter the text **Keyword**.
4. Double-click the **Push Button** widget and enter the text **Find**.
5. In the **Properties** view, change the **objectName** to **findButton**.

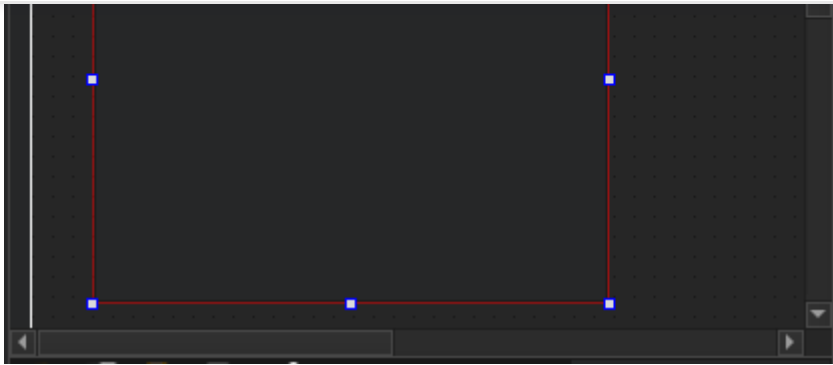


6. Press **Ctrl+A** (or **Cmd+A**) to select the widgets and select **Lay out Horizontally** (or press **Ctrl+H** on Linux or Windows or **Ctrl+Shift+H** on macOS) to apply a horizontal layout (**QHBoxLayout**).



7. Drag and drop a **Text Edit** widget (**QTextEdit**) to the form.
8. Select the screen area, and then select **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 select 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 the `Ui` object. 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** view 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** view 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:


```

#include <QIODevice>
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 "ui_textfinder.h"
#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:

```

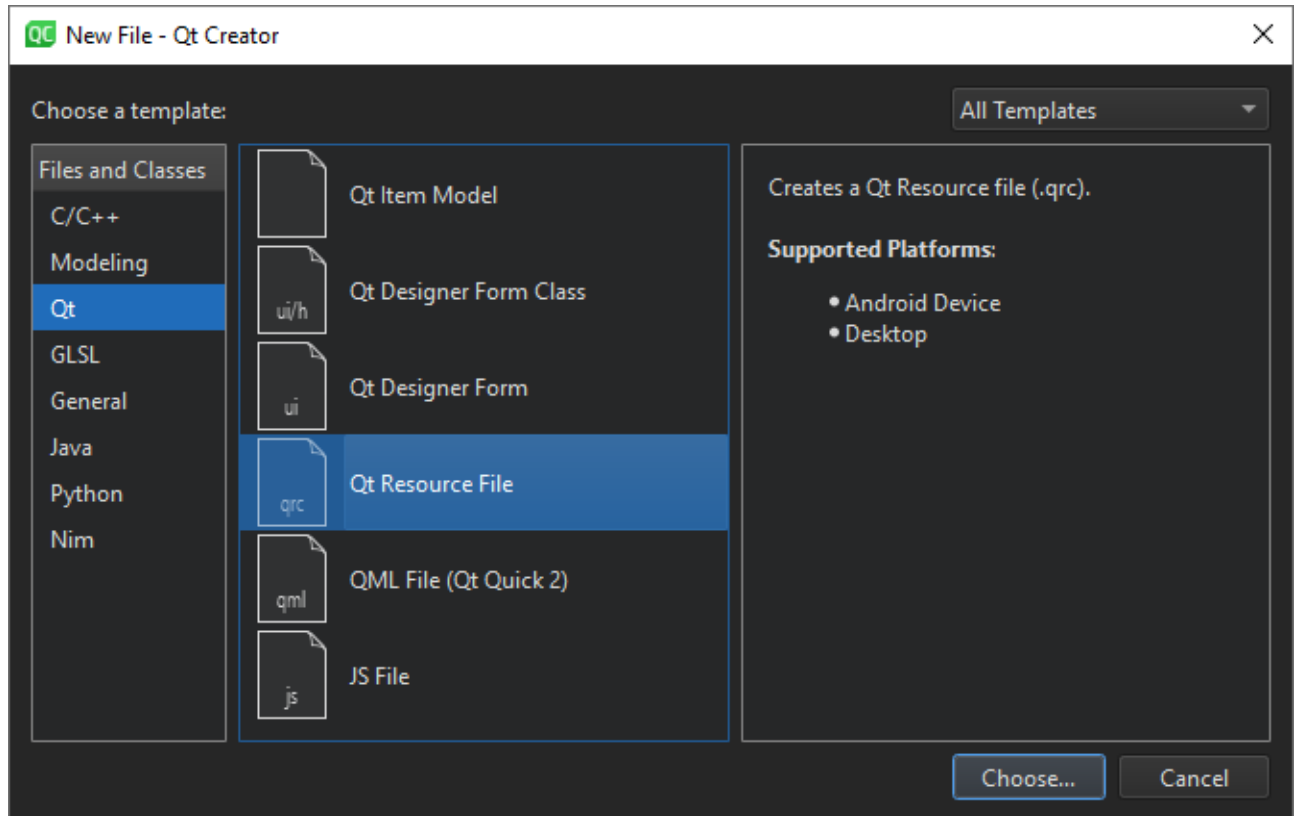
QMetaObject::connectSlotsByName(TextFinder);

```

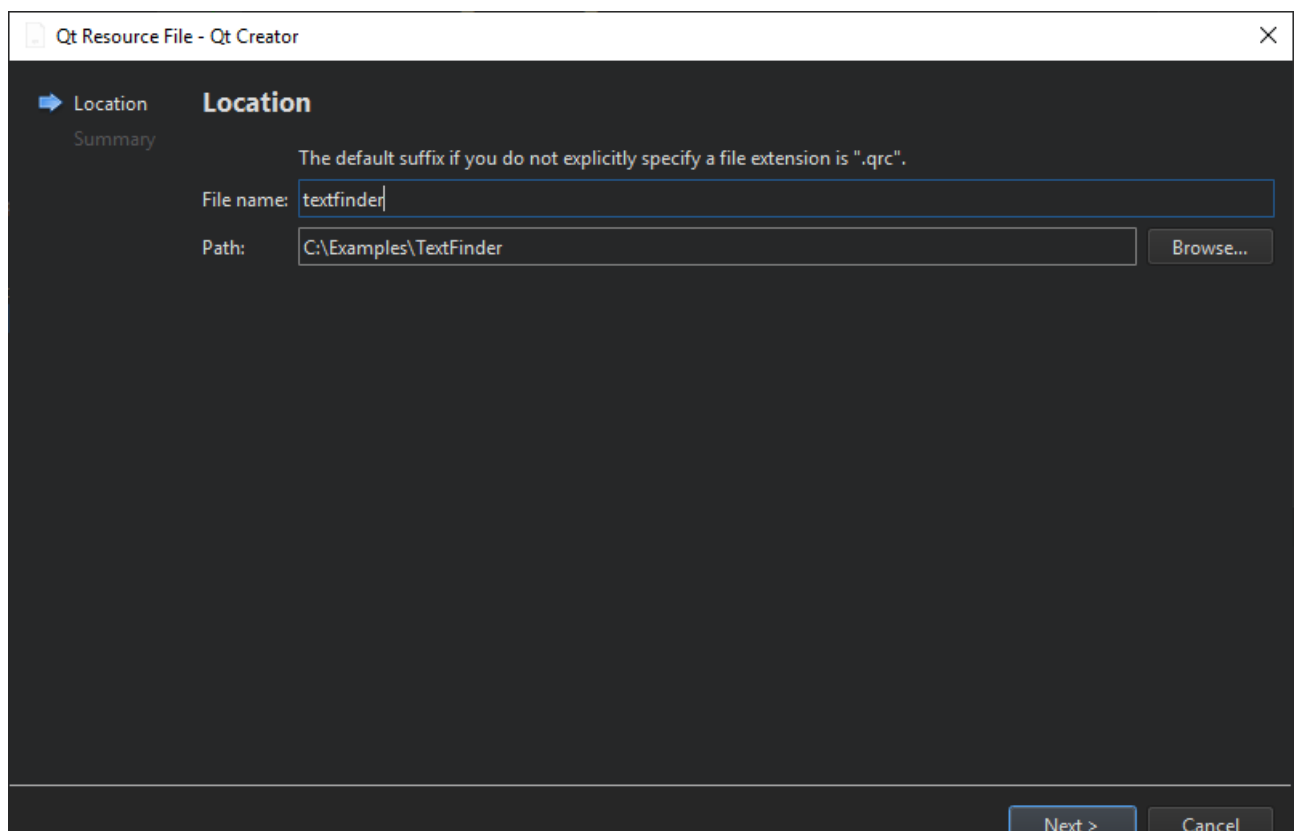
You need a resource file (.qrc) within which you embed the input text file. The input file can be any .txt file with a paragraph of text. Create a text file called input.txt and store it in the textfinder folder.

To add a resource file:

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

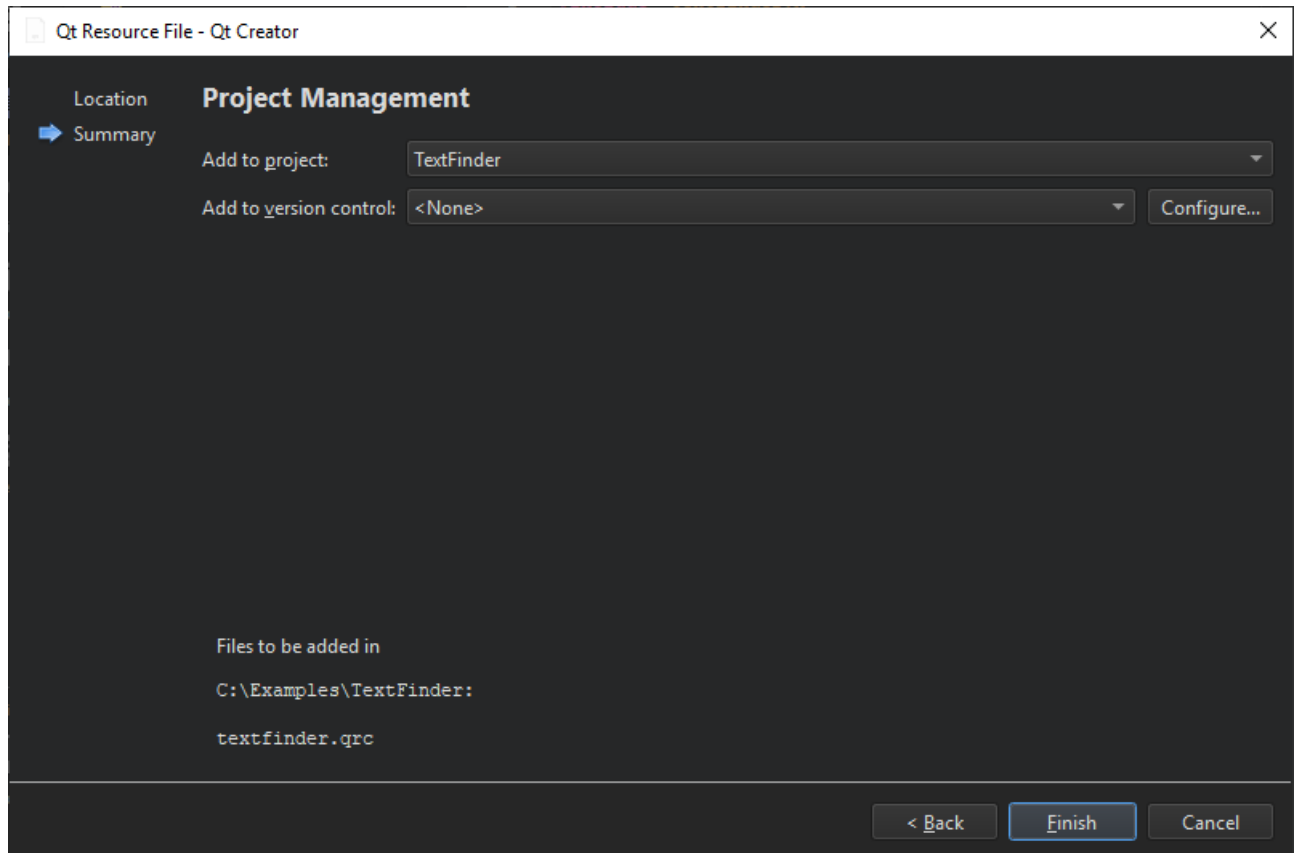


The **Choose the Location** dialog opens.



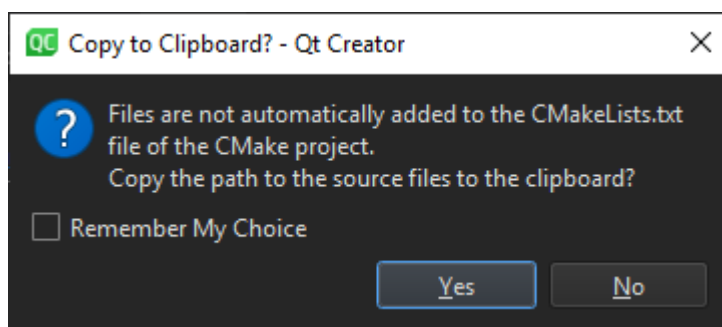
3. In the **Path** field, enter the path to the project, and select **Next** or **Continue**.

The **Project Management** dialog opens.



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

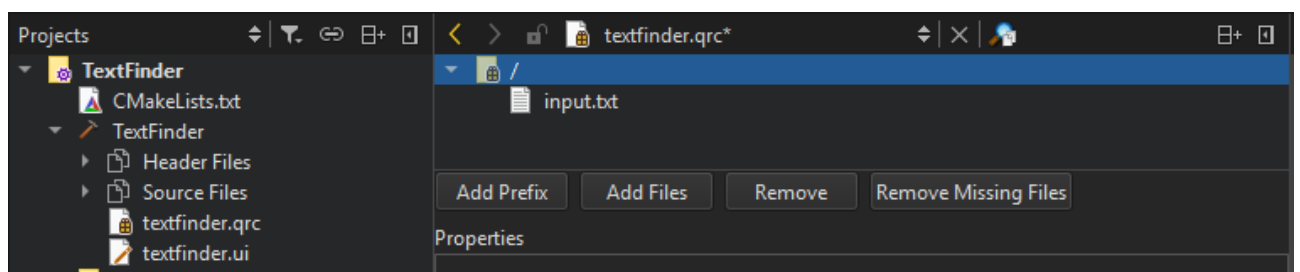
5. In the **Copy to Clipboard** dialog, select **Yes** to copy the path to the resource file to the clipboard for adding it to the CMakeLists.txt file.



6. Select **Add > Add Prefix**.

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

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




Adding Resources to Project File

For the text file to appear when you run the application, you must specify the resource file as a source file in the *CMakeLists.txt* file that the wizard created for you:

```
set(PROJECT_SOURCES
    main.cpp
    textfinder.cpp
    textfinder.h
    textfinder.ui
    ${TS_FILES}
    textfinder.qrc
)
```

Compiling and Running Your Application

Now that you have all the necessary files, select the  button to compile and run your Application.

[< Creating a Qt Quick Application](#)[Creating a Mobile Application >](#)

© 2022 The Qt Company Ltd. Documentation contributions included herein are the copyrights of their respective owners. The documentation provided herein is licensed under the terms of the [GNU Free Documentation License version 1.3](#) as published by the Free Software Foundation. Qt and respective logos are trademarks of The Qt Company Ltd in Finland and/or other countries worldwide. All other trademarks are property of their respective owners.

[Contact Us](#)

Company

[About Us](#)
[Investors](#)
[Newsroom](#)
[Careers](#)
[Office Locations](#)

Licensing

[Terms & Conditions](#)
[Open Source](#)
[FAQ](#)



Support

- Support Services
- Professional Services
- Partners
- Training

For customers

- Support Center
- Downloads
- Qt Login
- Contact Us
- Customer Success

Community

- Contribute to Qt
- Forum
- Wiki
- Downloads
- Marketplace