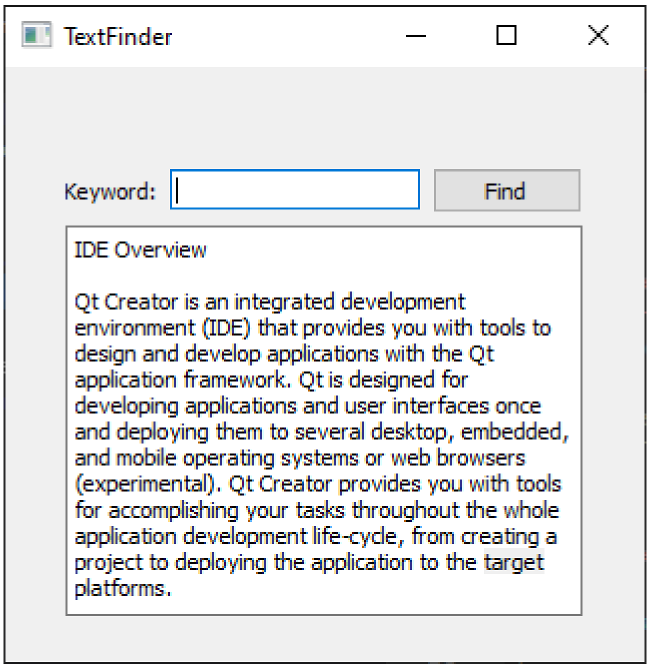
创建基于Qt小部件的应用程序

本教程介绍如何使用Qt创建器创建一个小型Qt应用程序，文本查找器。它是Qt UI工具文本查找器示例的简化版本。应用程序用户界面是使用 Qt 设计器从 Qt 小部件构造的。应用程序逻辑是使用代码编辑器以C++编写的。

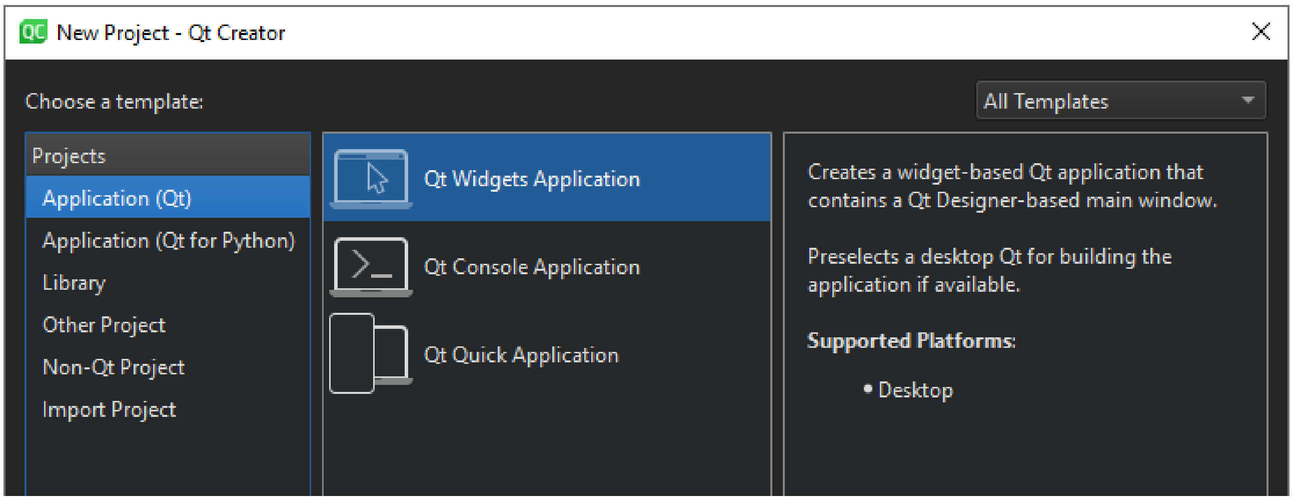


IDE Overview

Qt Creator is an integrated development environment (IDE) that provides you with tools to design and develop applications with the Qt application framework. Qt is designed for developing applications and user interfaces once and deploying them to several desktop, embedded, and mobile operating systems or web browsers (experimental). Qt Creator provides you with tools for accomplishing your tasks throughout the whole application development life-cycle, from creating a project to deploying the application to the target platforms.

创建文本查找器项目

1. 选择"**文件**>**新项目**>**应用程序**（Qt）> Qt **小部件应用程序**>**选择**"。



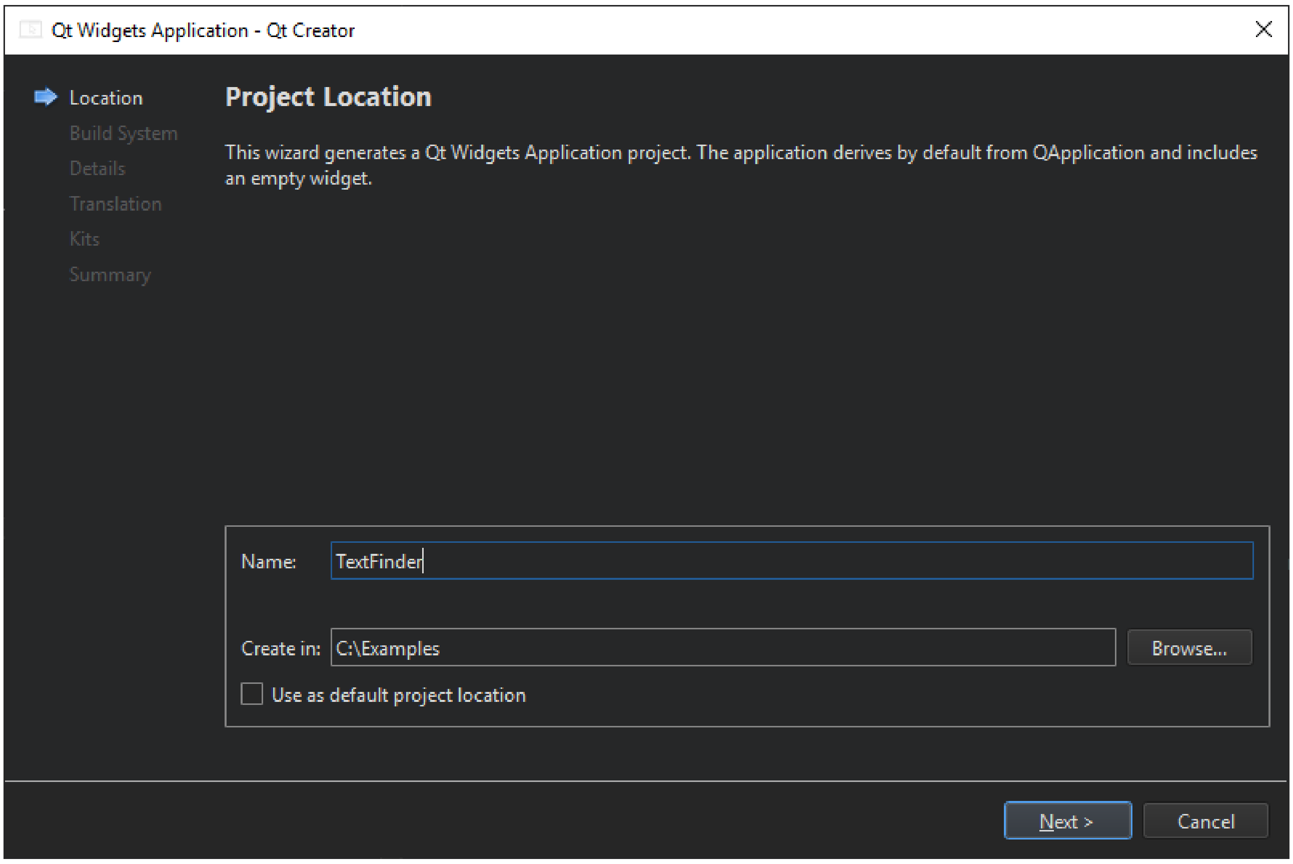
Creates a widget-based Qt application that contains a Qt Designer-based main window.

Preselects a desktop Qt for building the application if available.

Supported Platforms:

- Desktop

将打开“**简介**”和“**项目位置**”对话框。


Qt Widgets Application - Qt Creator

Project Location

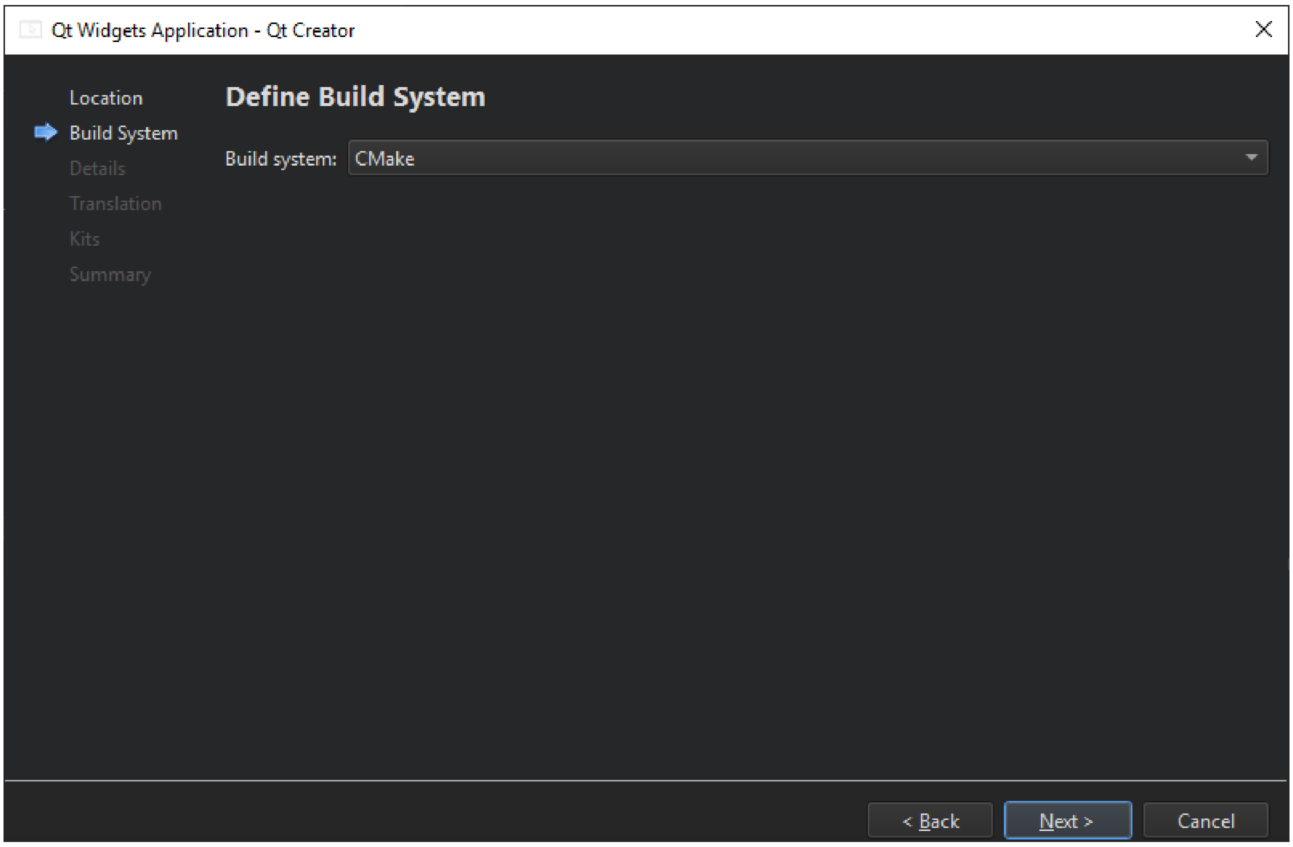
This wizard generates a Qt Widgets Application project. The application derives by default from QApplication and includes an empty widget.

Name: TextFinder

Create in: C:\Examples

Use as default project location

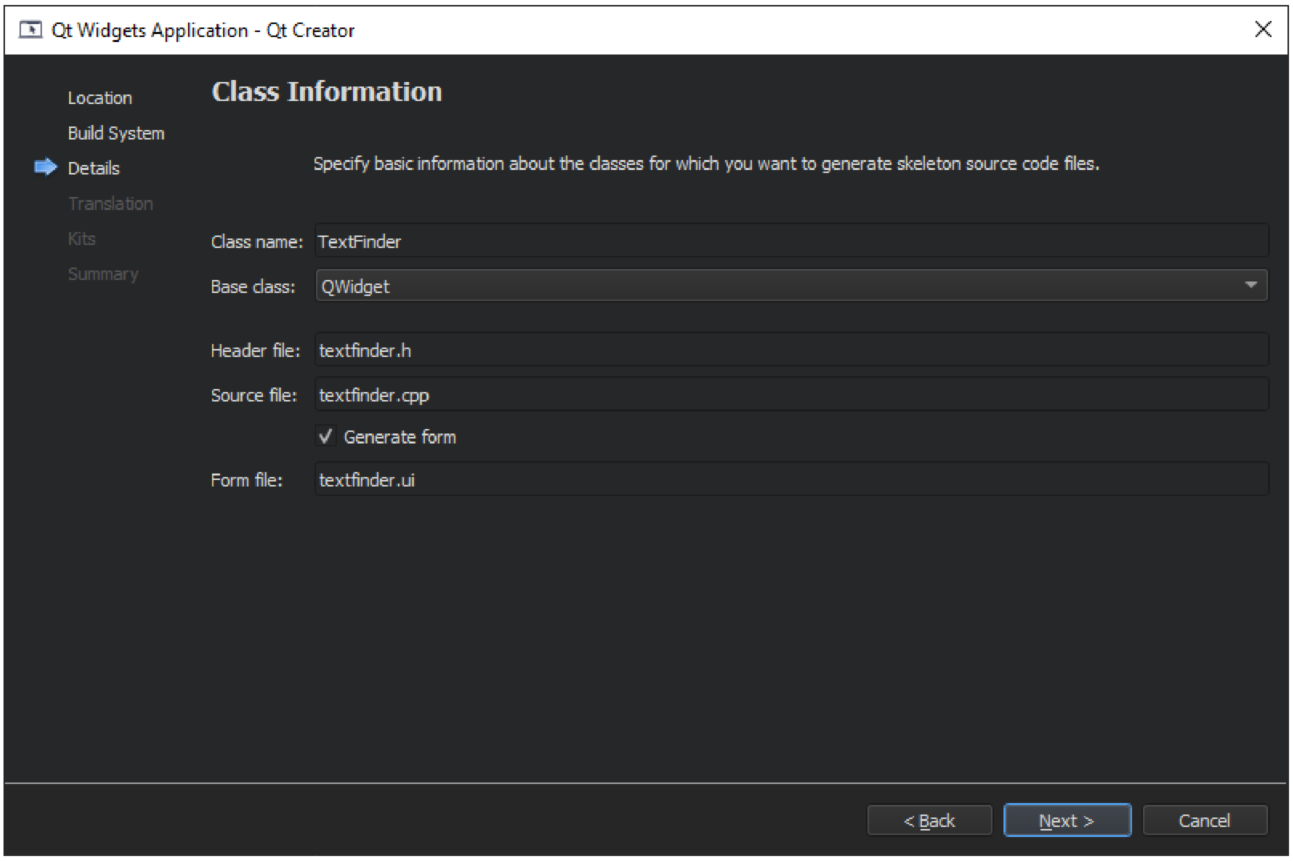
2. 在“**名称**”字段中，键入“**文本查找器**”。
3. 在“**创建位置**”字段中，输入项目文件的路径。例如。`C:\Qt\examples`
4. 选择“**下一步**”（在 Windows 和 Linux 上）或“**继续**”（在 macOS 上）以打开“**定义生成系统**”对话框。


Qt Widgets Application - Qt Creator

Define Build System

Build system: CMake

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



Qt Widgets Application - Qt Creator

Class Information

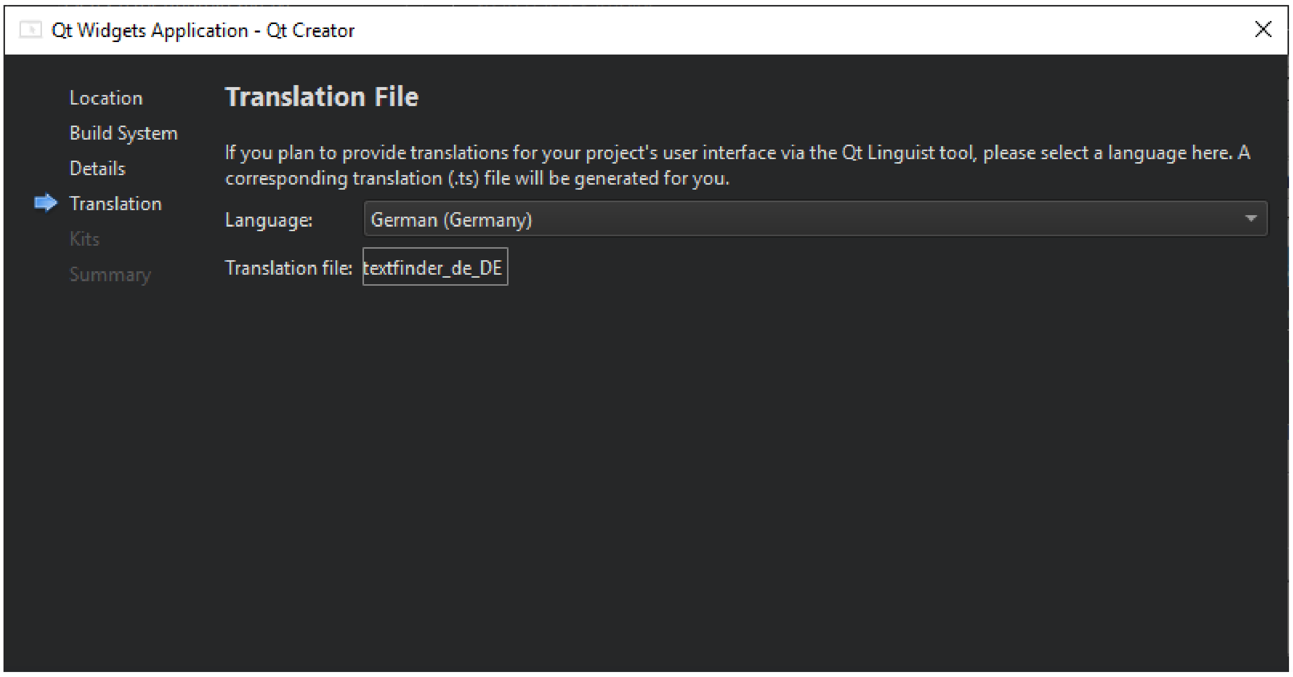
Specify basic information about the classes for which you want to generate skeleton source code files.

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

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.



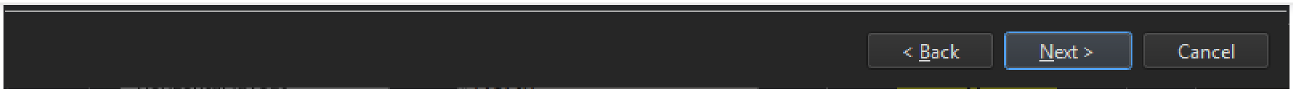
Qt Widgets Application - Qt Creator

Translation File

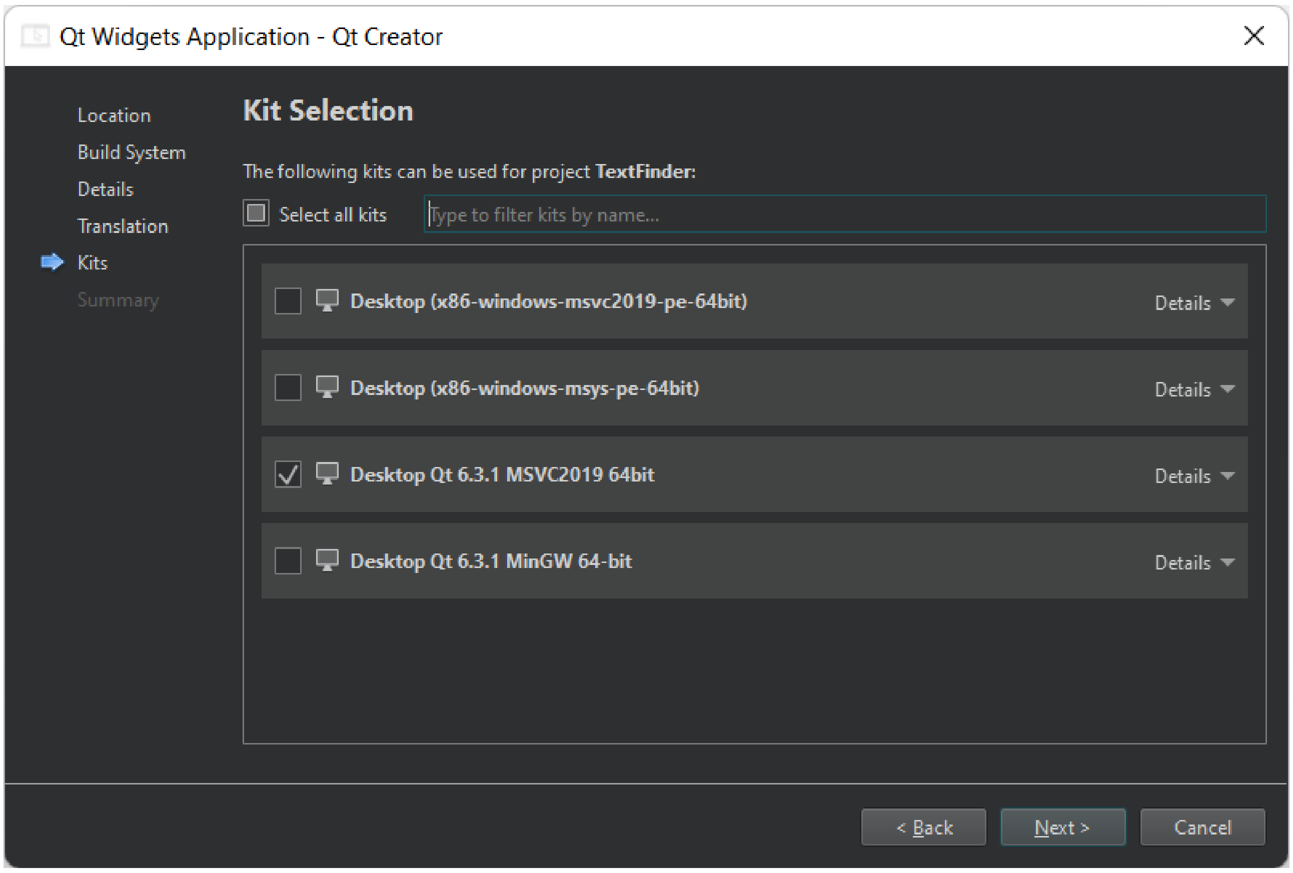
If you plan to provide translations for your project's user interface via the Qt Linguist tool, please select a language here. A corresponding translation (.ts) file will be generated for you.

Language: German (Germany)
Translation file: textfinder_de_DE

Qt DOCUMENTATION



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.

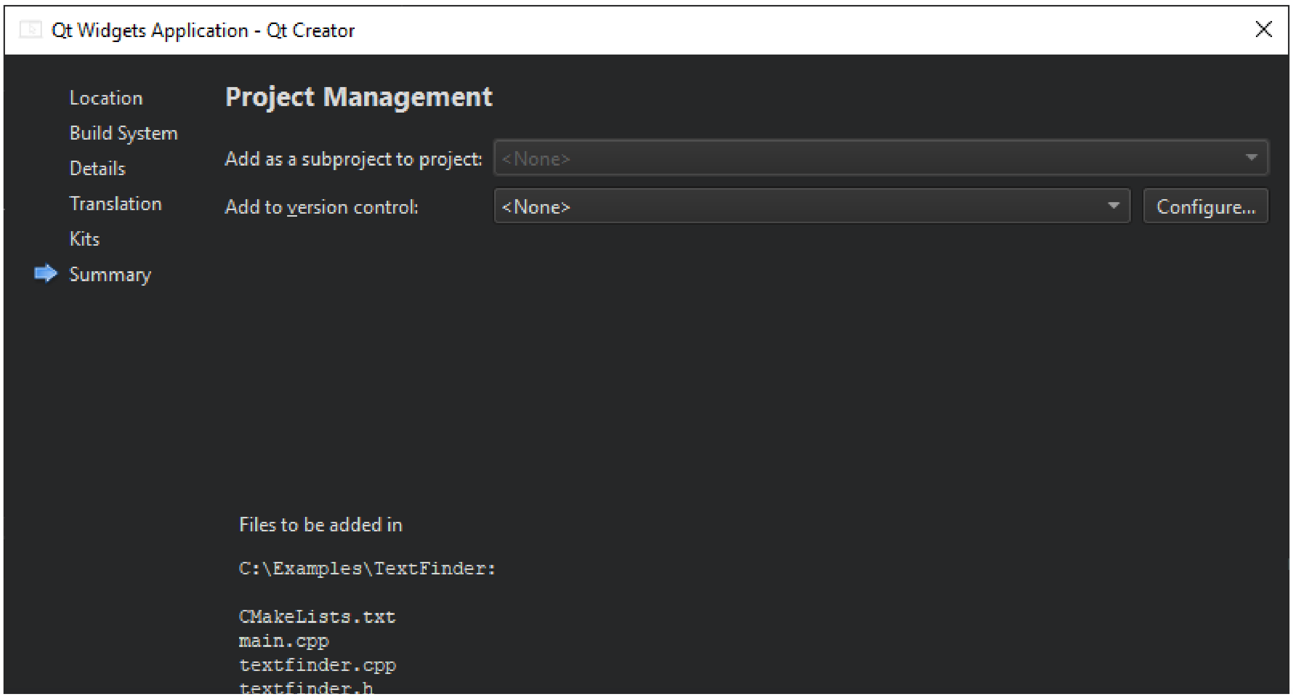


Kit Selection

The following kits can be used for project **TextFinder**:

- [ ] Desktop (x86-windows-msvc2019-pe-64bit)
- [ ] Desktop (x86-windows-msys-pe-64bit)
- [✓] Desktop Qt 6.3.1 MSVC2019 64bit
- [ ] Desktop Qt 6.3.1 MinGW 64-bit

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



Project Management

Add as a subproject to project: <None>

Add to version control: <None>

Files to be added in

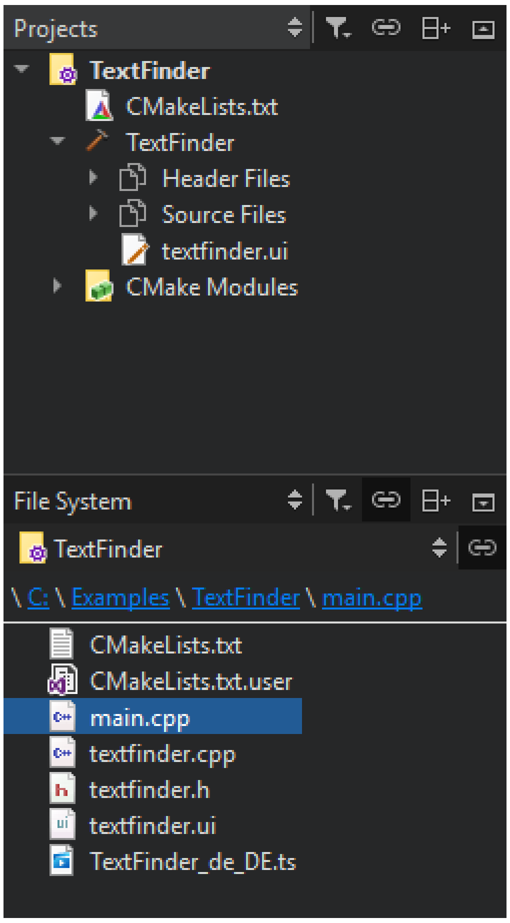
```
C:\Examples\TextFinder:

CMakeLists.txt
main.cpp
textfinder.cpp
textfinder.h
```

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

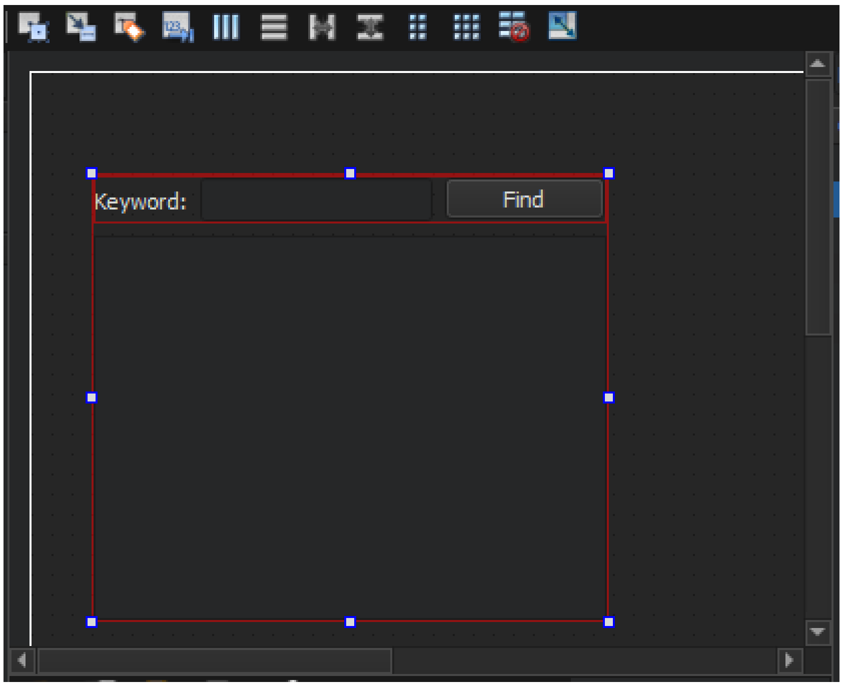


The .h and .cpp files come with the necessary boiler plate code.

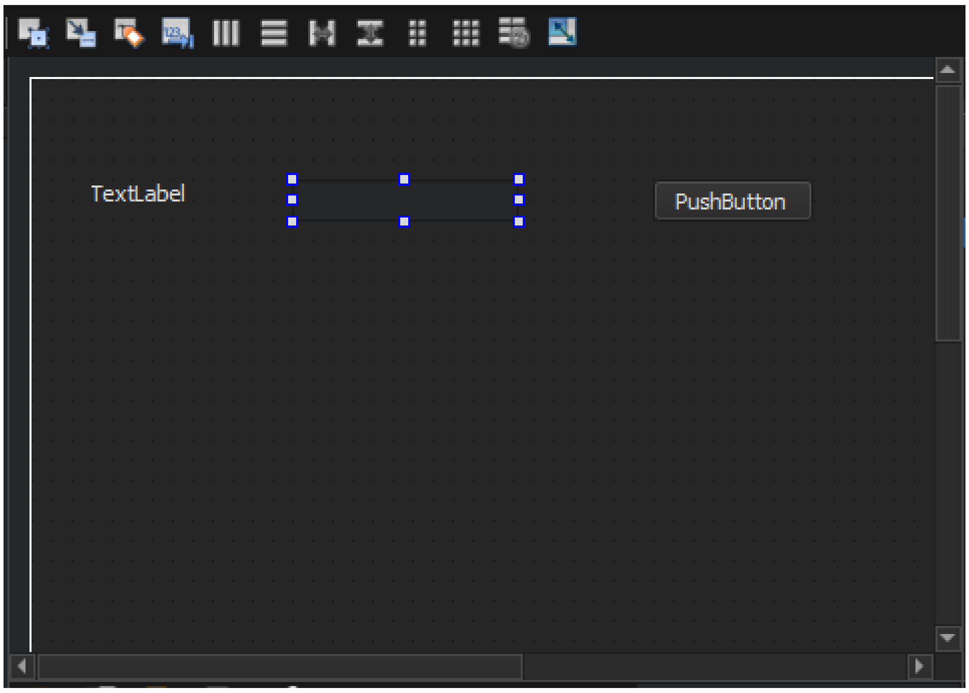
If you selected CMake as the build system, Qt Creator created a CMakeLists.txt project file for you.

Filling in the Missing Pieces

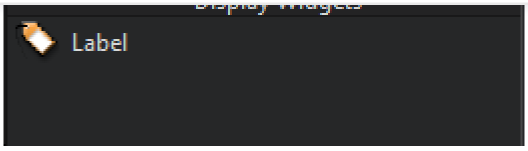
Begin by designing the user interface and then move on to filling in the missing code. Finally, add the find functionality.



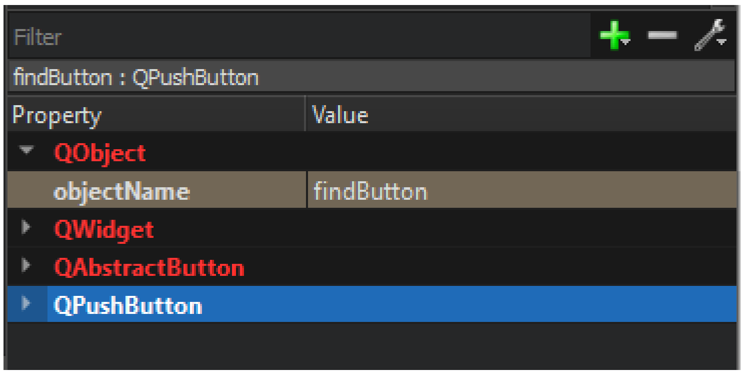
1. In the **Editor** mode, double-click the textfinder.ui file in the **Projects** view to launch the integrated Qt Designer.
2. Drag and drop the following widgets to the form:
   - **Label** (QLabel)
   - **Line Edit** (QLineEdit)
   - **Push Button** (QPushButton)



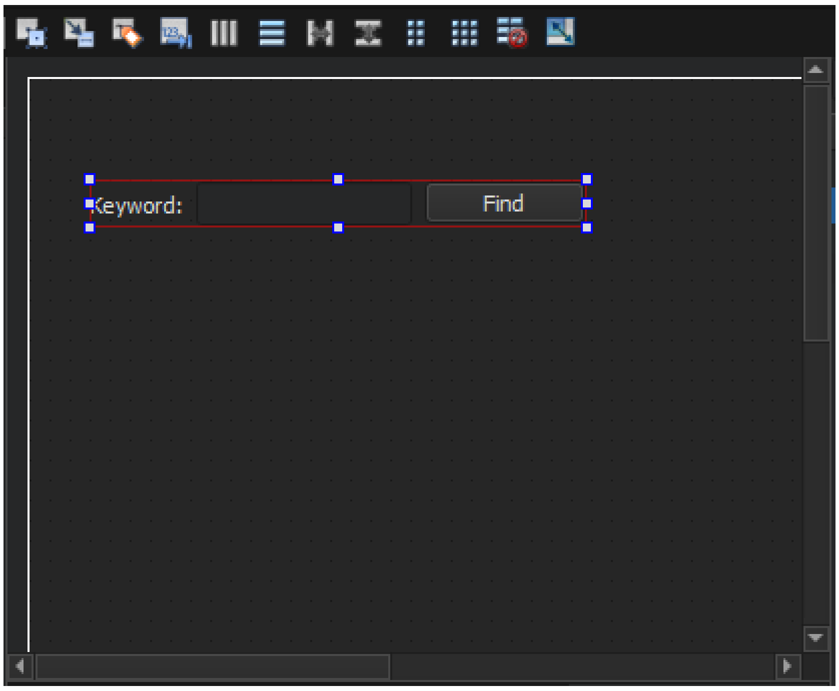
Note: To easily locate the widgets, use the search box at the top of the **Sidebar**. For example, to find the **Label** widget, start typing the word **label**.



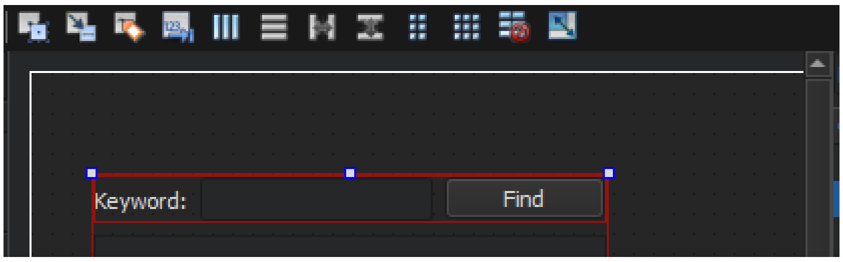
3. Double-click the **Label** widget and enter the text **Keyword**.
4. Double-click the **Push Button** widget and enter the text **Find**.
5. In the **Properties** view, change the **objectName** to **findButton**.

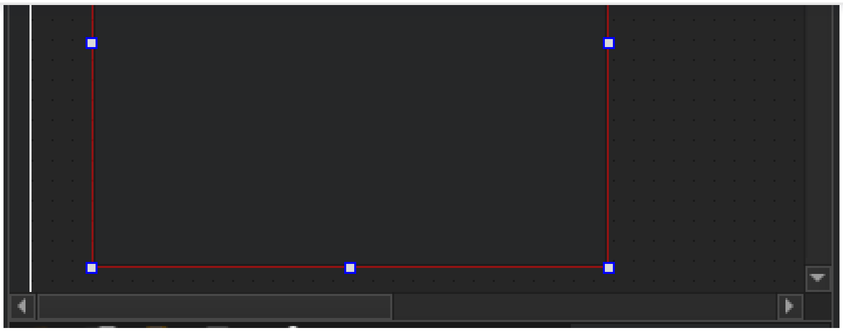


6. Press **Ctrl+A** (or **Cmd+A**) to select the widgets and select **Lay out Horizontally** (or press **Ctrl+H** on Linux or Windows or **Ctrl+Shift+H** on macOS) to apply a horizontal layout (QHBoxLayout).



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

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 object. You need to add a private function, , to read and display the contents of the input text file in the QTextEdit.`UiloadTextFile()`

1. In the **Projects** view in the **Edit view**, double-click the file to open it for editing.`textfinder.h`
2. Add a private function to the section, after the pointer, as illustrated by the following code snippet:`privateUi::TextFinder`

```
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 with QTextEdit::setPlainText(). This is illustrated by the following code snippet:`textEdit`

```
        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 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:`on_findButton_clicked()`

```
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 in the constructor, as illustrated by the following code snippet:`loadTextFile()`

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

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

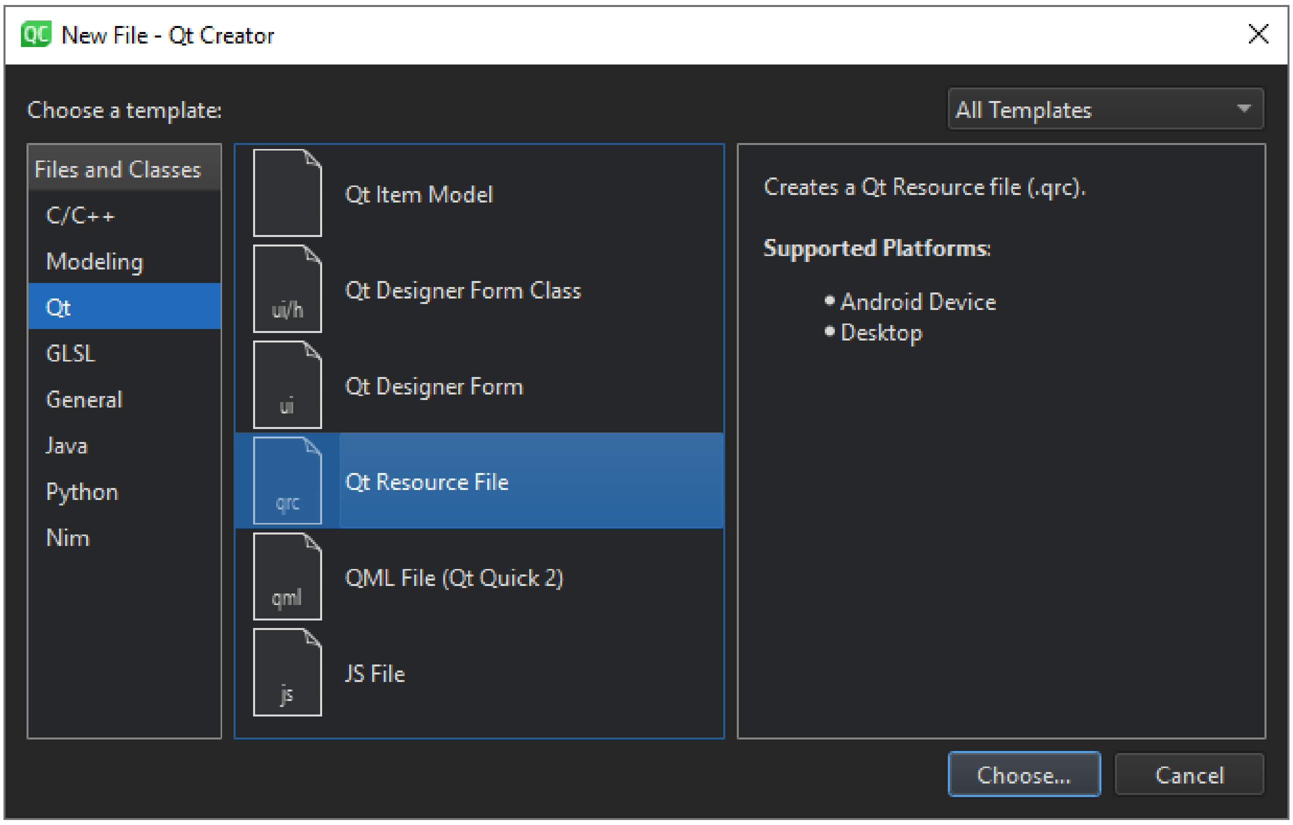
```
QMetaObject::connectSlotsByName(TextFinder);
```

Creating a Resource File

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**.

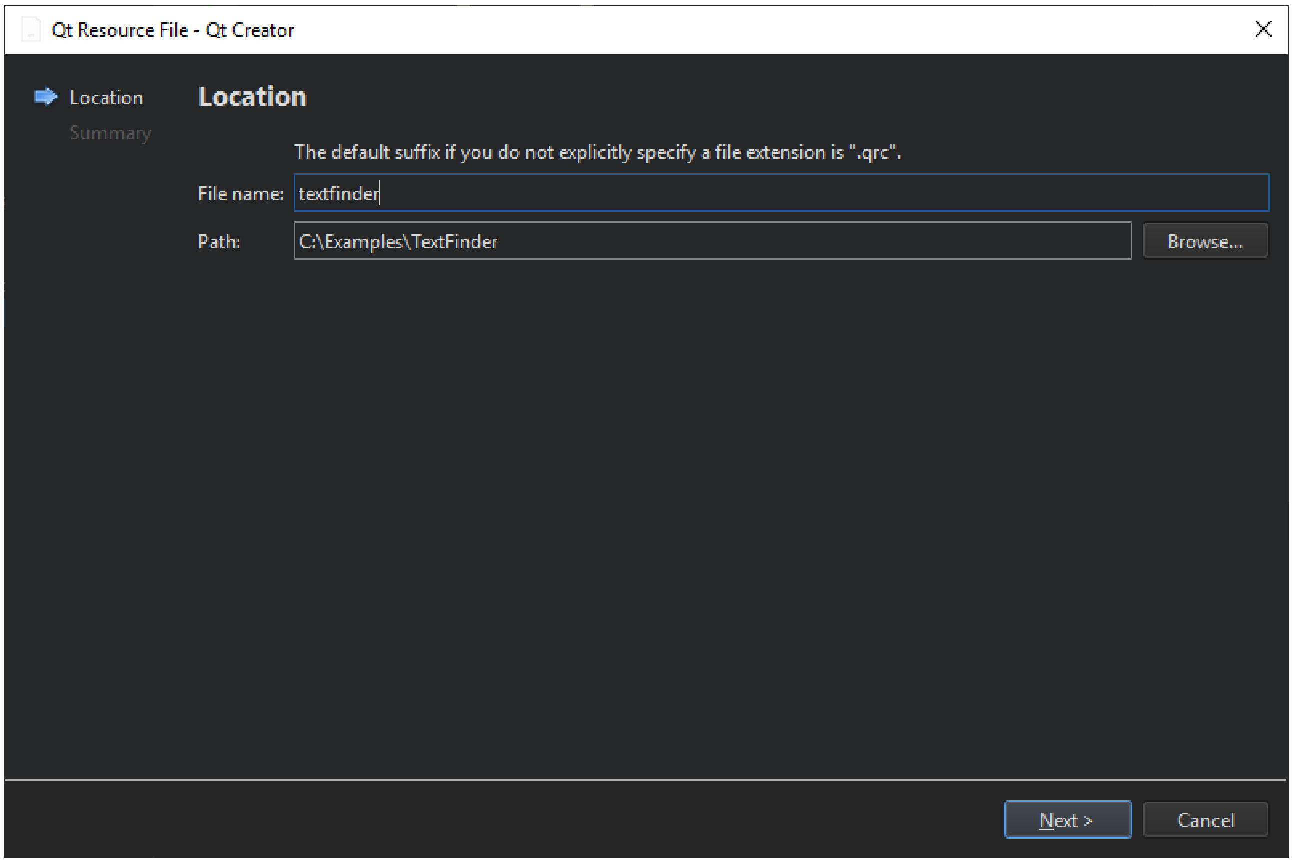


Creates a Qt Resource file (.qrc).

Supported Platforms:

- Android Device
- Desktop

The **Choose the Location** dialog opens.



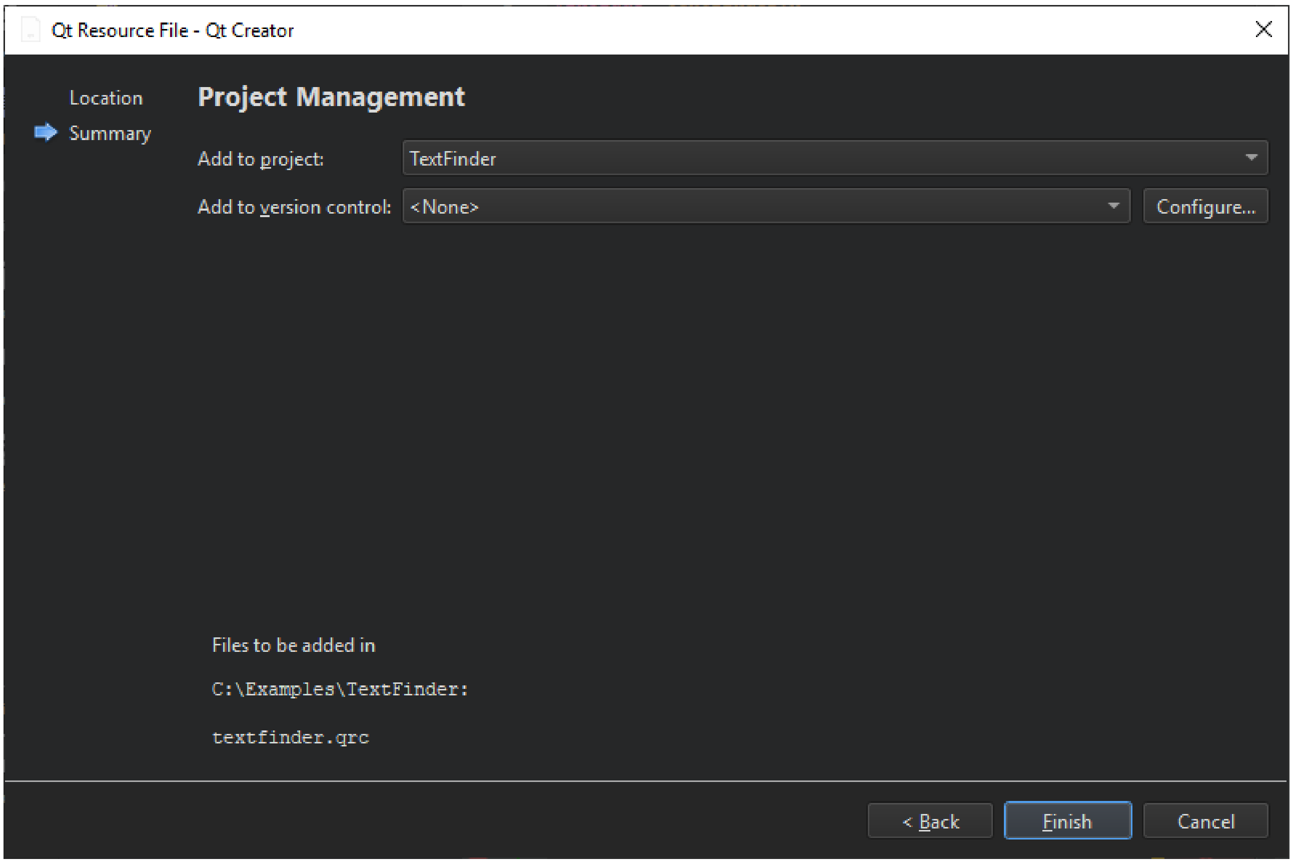
Location

The default suffix if you do not explicitly specify a file extension is ".qrc".

File name: textfinder

Path: C:\Examples\TextFinder

The **Project Management** dialog opens.


Qt Resource File - Qt Creator

Project Management

Add to project: TextFinder

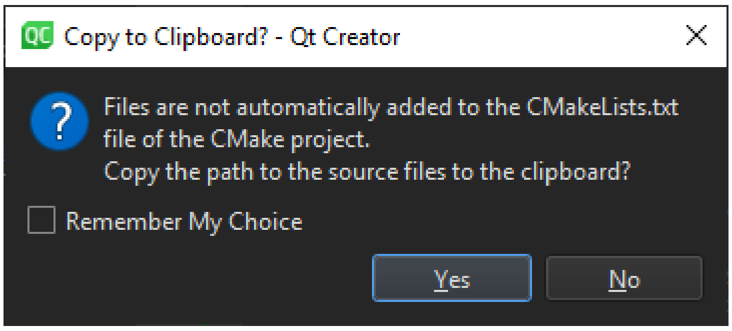
Add to version control: <None>

Files to be added in

C:\Examples\TextFinder:

textfinder.qrc

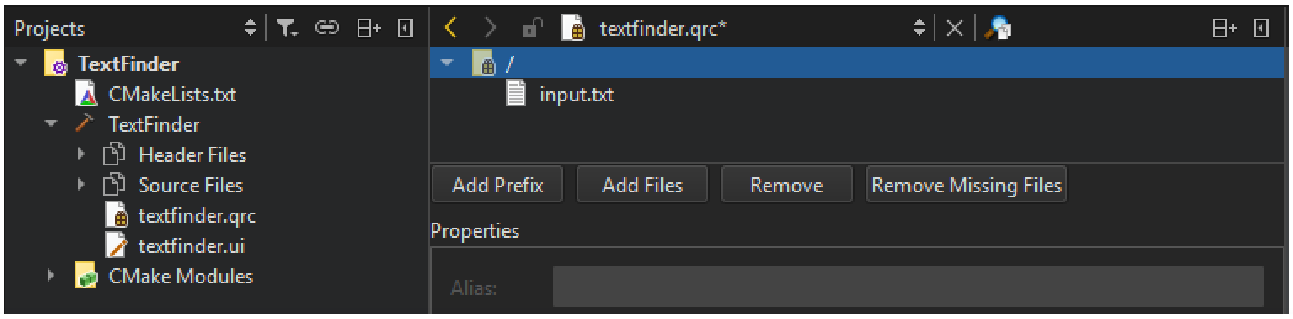
4. In the **Add to project** field, select **TextFinder** and select **Finish** or **Done** to open the file in the code editor.
5. In the **Copy** to Clipboard dialog, select **Yes** to copy the path to the resource file to the clipboard for adding it to the CMakeLists.txt file.


Copy to Clipboard? - Qt Creator

Files are not automatically added to the CMakeLists.txt file of the CMake project.
Copy the path to the source files to the clipboard?

Remember My Choice

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 ›

Qt DOCUMENTATION

Support Services
Professional Services
Partners
Training

Support Center
Downloads
Qt Login
Contact Us
Customer Success

Community

Contribute to Qt
Forum
Wiki
Downloads
Marketplace

Feedback Sign In