创建文件

可以使用向导模板将单个文件添加到项目中。下表列出了用于创建文件的向导模板。

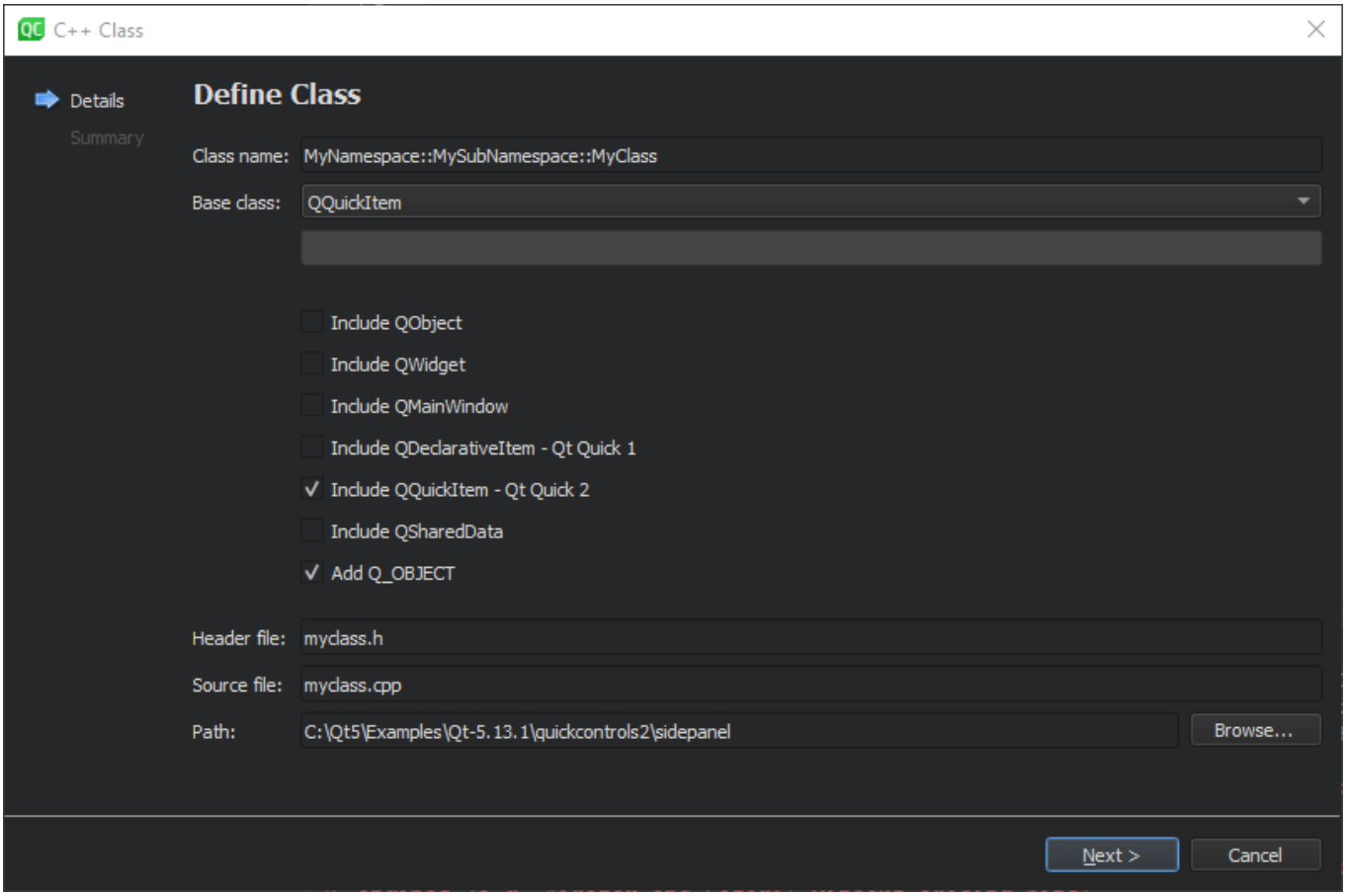
类别	向导模板	目的
C/C++	C++类	C++可以添加到C++项目的新类的标头和源文件。
	C/C++ 源文件	C++可以添加到C++项目中的源文件。
	C/C++ 头文件	C++可以添加到C++项目中的头文件。
建 模	状态图	包含状态机的样板代码的状态图 XML （SCXML） 文件。您可以使用 Qt SCXML 模块中的类在 Qt 应用程序中嵌入从文件创建的状态机。
	型	具有结构化关系图的通用建模语言 （UML） 样式模型。但是，模型编辑器使用 UML 的变体，并且仅提供用于指定模型元素外观的属性子集。有关详细信息，请参阅建模。
	划痕模型	使用临时文件暂存模型。
断续器	Qt 项目模型	可用于创建派生自 QAb 操作项模型、QAb 操作表模型或 QAb操作列表模型的类的源文件和头文件。
	Qt 设计器窗体类	Qt 设计器窗体和一个匹配类，用于实现基于 Qt 小部件的 UI。
	Qt 设计师表单	基于Qt小部件的项目的Qt设计器表单。如果已有 UI 逻辑的现有类，这将非常有用。
	Qt 资源文件	用于在应用程序可执行文件中存储二进制文件的资源文件。
	QML文件 （Qt 快速2）	QML 文件，该文件导入 Qt 快速 2.0 以在 Qt 快速项目中使用。
	Qt 快速用户界面文件	用于 Qt 快速项目的 UI 文件 （*.ui.qml*） 和相应的实现文件 （*.qml*） 。
	文件	可用于在 Qt 快速项目中编写应用程序逻辑的 JavaScript 文件。
断续器	片段着色器	片段着色器，用于为使用 OpenGL 渲染的三角形、点和线条生成最终像素颜色。您可以在Qt快速项目和基于Qt小部件的项目中使用它。
	顶点着色器 （开放英语/ES 2.0）	顶点着色器，用于转换使用 OpenGL 渲染的三角形、点和线条的位置、法线和纹理坐标。您可以在Qt快速项目和基于Qt小部件的项目中使用它。
	片段着色器 （桌面打开 GL）	片段着色器，用于Qt快速项目和基于Qt小部件的项目。
类别	向导模板	目的

常规	空文件	您可以使用任何文件名扩展保存的空文件。
	划痕缓冲器	使用临时文件的暂存缓冲区。您可以创建此类文件以临时存储您不打算保存的信息
爪哇岛	文件	可用于创建 Java 类的 Java 类文件。
蟒	蟒蛇类	蟒蛇类文件。
	蟒蛇文件	使用 UTF-8 编码的 Python 脚本文件。
尼姆（实验性）	尼姆脚本文件	使用 UTF-8 编码的空 Nim 脚本文件。
	尼姆文件	使用 UTF-8 编码的空 Nim 源文件。

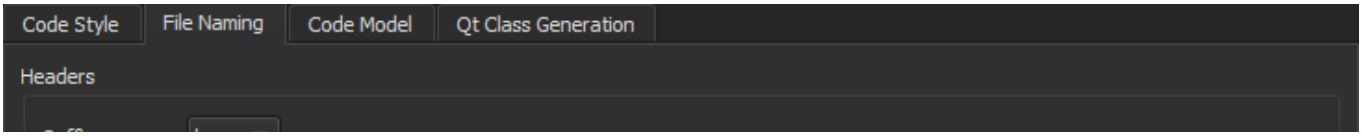
创建C++类

使用C++类向导可以为可以添加到C++项目的新类创建C++头文件和源文件。指定类的类名、基类以及标头和源文件。

该向导支持命名空间。若要使用命名空间，请在"类名"字段中输入限定的**类名**。例如：。该向导会在您键入时建议现有的命名空间和类名。`MyNamespace::MySubNamespace::MyClass`



The names of the header and source file are based on the class name. To change the default suffix of a file, select **Edit** > **Preferences** > **C++** > **File Naming**.

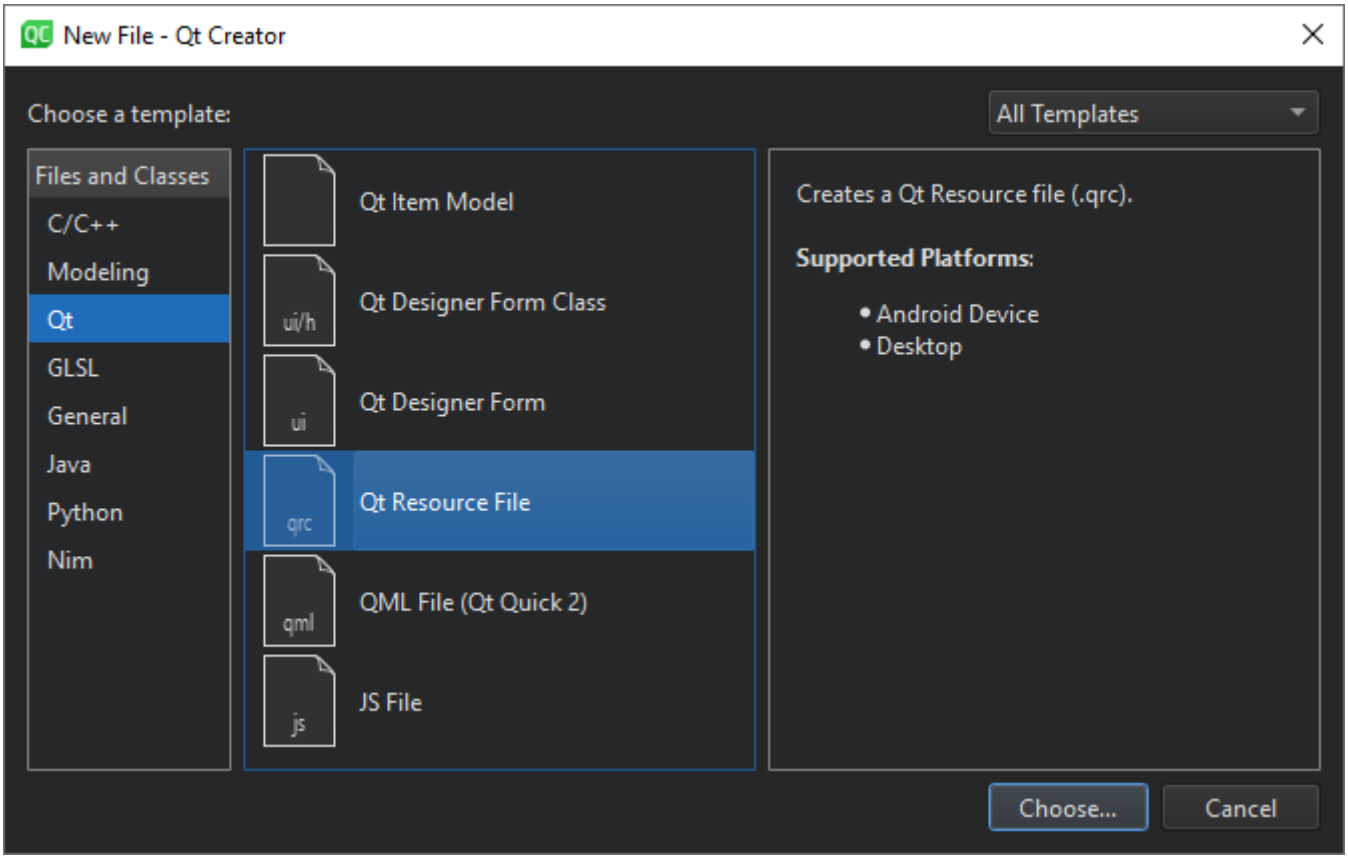


In the **License template** field, you can use predefined wizard variables to specify the path and filename of the license to use in the source and header files.

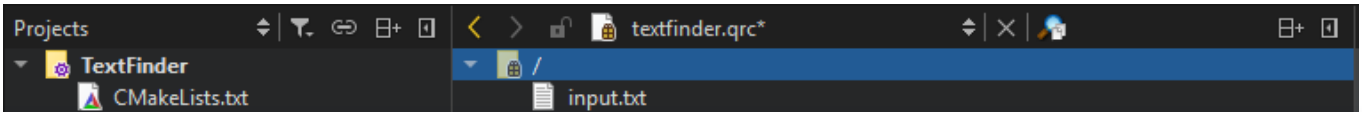
You can create your own project and class wizards. For more information, see Adding New Custom Wizards.

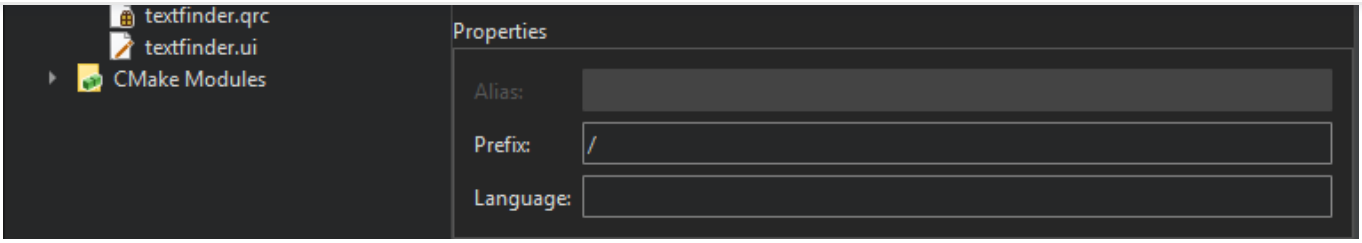
Creating Resource Files

Qt Creator supports the Qt Resource System, which is a platform-independent mechanism for storing files in the application's executable.



The wizard creates a resource collection file (.qrc) that you can manage in the resource editor.





Select **Add Files** to locate and add individual files.

To list the folders and files in ascending alphabetic order in the source tree, select **Sort Alphabetically** in the context menu.

By default, resources are accessible in the application under the same file name as they have in the source tree, with a prefix, or by a URL with a scheme. To specify a path prefix for all files in the file, select **Add Prefix** and enter the prefix in the **Prefix** field. `:/qrc.qrc`

Some resources need to change based on the user's locale, such as translation files or icons. You can specify a locale in the **Language** field.

Select **Remove** to remove the selected file from the resource collection. In the **Remove File** dialog, select the **Delete file permanently** check box to remove the file from the file system. To remove files that cannot be found in the file system, select **Remove Missing Files**.

The above functions are also available in the context menu in the **Projects** view.

Creating OpenGL Fragment and Vertex Shaders

Qt provides support for integration with OpenGL implementations on all platforms, which allows you to display hardware accelerated 3D graphics alongside a more conventional user interface. For more information, see Qt GUI.

You can use the QOpenGLShader class to compile OpenGL shaders written in the OpenGL Shading Language (GLSL) and in the OpenGL/ES Shading Language (GLSL/ES). QOpenGLShader and QOpenGLShaderProgram shelter you from the details of compiling and linking vertex and fragment shaders.

You can use Qt Creator code editor to write fragment and vertex shaders in GLSL or GLSL/ES. The code editor provides syntax highlighting and code completion for the files.