常见问题

本节包含有关Qt创建者的一些常见问题的解答。您还可以在"已知问题"和"操作方法"部分或特殊区域（如调试）的"疑难解答"部分中找到问题的答案。

一般问题

如何重置所有Qt创建者设置?

删除Qt创建者创建的设置文件。

有关文件在每个受支持平台上的位置的详细信息，请参阅设置文件的位置。

Qt创建者附带了MinGW，我应该将此版本与Qt一起使用吗?

使用针对 Qt 版本构建的版本。

Qt 创建者找不到帮助程序应用程序，如 Git 或编译器。我该怎么办?

确保应用程序在启动Qt创建器时位于系统PATH中。此外，选择"**编辑**>**首选项**"以检查为应用程序指定的设置。许多插件指定了它们所需的工具的路径或它们在其中运行的环境。

这与macOS尤其相关，当Qt创建者启动时，macOS可能不在路径中。`/usr/local/bin`

如何更改Qt创建者的界面语言?

Qt创建者已被本地化为多种语言。如果系统语言是受支持的语言之一，则会自动选择它。要更改语言，请选择"**编辑**>**首选项**>**环境**"，然后在"语言"字段中选择一**种语言**。选择"**立即重新启动**"以重新启动 Qt Creator 并使更改生效。

报告的问题是否已得到解决?

您可以在Qt项目错误跟踪器中查找任何问题。

Qt 设计者集成问题

为什么自定义小部件在设计模式下不加载，即使它在独立的Qt设计器中工作?

Qt设计器从标准位置获取插件，并加载与其构建密钥匹配的插件。独立和集成Qt设计器的位置不同。

有关更多信息，请参阅 添加 Qt 设计器插件。

快速问题

为什么我的 QML 导入下方有一条红线，即使我有模块?

表.txt"文件中。`QML_IMPORT_PATH.proset`

这还可以实现 QML 代码的代码完成并删除错误消息。

以下示例说明了如何为 qmake 项目指定导入路径，以便在不同目标平台的生成和运行工具包之间切换时，该路径可以正常工作：

```
TEMPNAME = $${QMAKE_QMAKE}
MY_QTPATH = $$dirname(TEMPNAME)
QML_IMPORT_PATH += $$MY_QTPATH/../qml
message("my QML Import Path: "$$QML_IMPORT_PATH)
```

有关如何设置 CMake 项目的导入路径的更多信息，请参见导入 QML 模块。

当 Qt 创建者抱怨缺少 OpenGL 支持时，我该怎么办?

Qt 创建器的某些部分，如 QML 探查器，使用 Qt 快速 2，它依赖于 OpenGL API 进行绘制。不幸的是，使用OpenGL 可能会导致问题，特别是在远程设置和过时的驱动程序中。在这些情况下，Qt 创建器会在控制台上显示与 OpenGL 相关的错误消息，或将其记录在 Windows 调试器日志中。

修补程序和解决方法会有所不同，具体取决于您的设置。作为最后的手段，您可以禁用受影响的插件。

虚拟机

尝试在虚拟机的设置中启用 *3D 加速*。对于虚拟框，还要确保已安装来宾插件，包括实验性的 *Direct3D 支持*。

窗户

Check whether Qt Creator has been compiled with OpenGL/Desktop, or ANGLE as a backend. The official binaries are always built with ANGLE (a library that maps OpenGL ES API to DirectX).

- ANGLE backend: This requires a Windows version newer than Windows XP. If you have problems, try updating your graphics drivers or update your DirectX version. Run to check whether *Direct3D Acceleration* is indeed enabled.`dxdiag.exe`
- OpenGL backend: Make sure your graphics driver supports OpenGL 2.1 or newer. Try to update your graphics driver.

Unix

Run for a quick check whether OpenGL works in general. Check the output of to get more details like the OpenGL driver and renderer (search for *OpenGL* in the application's output).`glxgearsglxinfo`

If you are using the Mesa driver, you can force OpenGL to be rendered in software by setting the environment variable.`LIBGL_ALWAYS_SOFTWARE`

Disabling plugins

You can disable the Qt Creator plugins, at the expense of losing functionality:

- Launch Qt Creator from command line, with the arguments.`-noload QmlProfiler -noload QmlDesigner`
- Disable the plugins permanently by selecting **Help** > **About Plugins**.

Help Questions

Qt Creator comes fully integrated with Qt documentation and examples using the Qt Help plugin. The integrated Qt Reference Documentation is available for Qt 4.4 and later. Qt Creator and other Qt deliverables contain documentation as .qch files. All the documentation is accessible in the **Help** mode.

To view the documentation that is available and to add documentation, select **Edit** > **Preferences** > **Help** > **Documentation**. For more information, see Adding External Documentation.

Debugger Questions

For information on troubleshooting debugger, see Troubleshooting Debugger.

If I have a choice of GDB versions, which should I use?

On Linux and Windows, use the Python-enabled GDB versions that are installed when you install Qt Creator and Qt. On macOS, GDB is no longer officially supported. To build your own Python-enabled GDB, follow the instructions in Building GDB.

You must use Python version 2.6 or 2.7.

For more information on setting up debugger, see Setting Up Debugger.

How do I generate a core file in Qt Creator?

To trigger the GDB command that generates a core file while debugging, select **View** > **Views** > **Debugger Log**. In the **Command** field, type and press **Enter**. The core file is created in the current working directory. You can specify another location for the file, including a relative or absolute path, as an argument of the command.`gcore`

Compiler Questions

How can I make use of my multi-core CPU with Qt Creator?

On Linux and macOS, go to **Projects** mode, select your configuration in the **Build Settings**, locate the **Build Steps**, and add the following value, where is the amount of cores in your CPU: `<num>-j <num>`

On Windows, nmake does not support the parameter. Instead, we provide a drop-in replacement called jom. You can download a precompiled version of jom from Qt Downloads. Put jom.exe in a location in the %PATH%. Go to the **Build Settings** and set jom.exe as the make command.`-j`

> **Note:** Unlike GNU make, jom automatically detects your cores and spawns as many parallel processes as your CPU has cores. You can override this behavior by using the parameter as described above.`-j`

Qt Installation Questions

I cannot use QSslSocket with the Qt I installed from binary packages. What should I do?

The Qt in the binary packages is built with QT_NO_OPENSSL defined. Rebuilding it is possible. For more information, see http://www.qtcentre.org/threads/19222-Qssl.

Which development packages from the distribution are needed on Ubuntu or Debian?

```
sudo apt-get install libglib2.0-dev libSM-dev libxrender-dev libfontconfig1-dev libxext-dev
```

```
sudo apt-get install libgl-dev libglu-dev
```

Platform Related Questions

Where is application output shown in Qt Creator?

On Unix (Linux and macOS): and related functions use the standard output and error output. When you run or debug the application, you can view the output in Application Output.`qDebug()`

For console applications that require input, select **Projects** > **Run Settings** > **Run in terminal**. To specify the terminal to use, select **Edit** > **Preferences** > **Environment** > **System**.

On Windows: Output is displayed differently for *console applications* and *GUI applications*.

For qmake projects, the setting in the .pro file specifies that the application is built as a console application using some other runtime.`CONFIG += console`

This is the standard behavior for CMake projects. To create a GUI application on Windows or an application bundle on macOS, you must specify the or property for the qt_add_executable command in the CMakeLists.txt file.`WIN32MACOSX_BUNDLE`

When you run a console application, you can view the output in the console window of the calling application. If the calling application is a GUI application (for example, a release-built version of Qt Creator), a new console window is opened. For this type of application, and related functions use standard output and error output.`qDebug()`

We recommend that you select **Projects** > **Run Settings** > **Run in terminal** for console applications.

For GUI applications, and related functions use the Windows API function . The output is displayed in **Application Output**. However, Qt Creator can show output from only one source at the time for it to be displayed correctly. You can use an external debug output viewer, such as the DebugView for Windows to display output from GUI applications.`qDebug()` `OutputDebugString()`

Questions about New Features

Will a requested feature be implemented?

If it is a scheduled feature, you can see this in the task tracker. If a feature already has been implemented, it is mentioned in the changes file for the upcoming release.

Why does Qt Creator not use tabs for editors?

This question comes up from time to time, so we have considered it carefully. Here are our main reasons for not using tabs:

- Tabs do not scale. They work fine if you have 5 to 6 editors open, they become cumbersome with 10, and if you need more horizontal space than the tab bar, the interface does not work at all.
- Tabs do not adapt to your working set.
- The common solution is to give the user the ability to reorder tabs. Now user has to manage tabs instead of writing code.
- Tabs force you to limit the amount of open editors because otherwise you get confused.

Consider the following use case: *Developers want to switch editors.*

One common factor in many use cases is switching editors while working on a set of open files. While working on files A and B, users sometimes need to look at file C. They can press **Ctrl+Tab** to move between the files and have the files open in the correct editor according to file type. The list is sorted by last used.

Typically, users also work on multiple classes or functions that are related, even though they are defined or declared in different files. Qt Creator provides two shortcuts for that: **F2** to follow the symbol under cursor and **Ctrl+Shift+U** to find references to it.

In addition, developers can:

- Press **F4** to switch between header and source.
- Press **Alt+Left** to move backwards in the navigation history.
- Use the locator (Ctrl+K) to simply tell Qt Creator where to go.

The locator can be used to open files, but opening files is also just a step on the way to accomplish a task. For example, consider the following use case: *Fix AFunction in SomeClass which comes from someclass.cpp/someclass.h*.

With a tabbed user interface, developers would search for someclass.cpp in the tab bar, and then search for , only to find out that the function is not located in that file. They would then search for someclass.h in the tab bar, find our that the function is inline, fix the problem, and forget where they came from.`::AFunction`

With Qt Creator, developers can type to find the function. Typically, they only need to type 3 to 4 characters of the function name. They can then fix the problem and press **Alt+Back** to go back to where they were.`Ctrl+K m AFun`

Other locator filters include for classes, for all symbols, and (thanks to a community contribution) for symbols in the current file.`c:.`

‹ Using the Help Mode

How-tos ›

© 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.

Qt The Qt Company

Contact Us

Company

About Us
Investors
Newsroom
Careers
Office Locations

Licensing

Terms & Conditions
Open Source
FAQ

Support Services
Professional Services
Partners
Training

Support Center
Downloads
Qt Login
Contact Us
Customer Success

Community

Contribute to Qt
Forum
Wiki
Downloads
Marketplace

© 2022 The Qt Company