Qt Creator

Qt Creator is a **cross-platform IDE** (integrated development environment) tailored to the needs of Qt developers.

Projects

One of the major advantages of Qt Creator is that it allows a team of developers to share a project across different development platforms with a common tool for development and debugging.

But why do you need projects? To be able to build and run applications, Qt Creator needs the same information as a compiler would need. This information is specified in the project build and run settings.

Creating a project allows you to:

- Group files together
- Add custom build steps
- Include forms and resource files
- Specify settings for running applications

You can either create a project from scratch or import an existing project. Qt Creator generates all the necessary files, depending on the type of project you create. For example, if you choose to create a graphical user interface (GUI) application, Qt Creator generates an empty .ui file that you can modify with the integrated Qt Designer.

Qt Creator is integrated with cross-platform systems for build automation: **qmake** and **CMake**. In addition, you can import generic projects that do not use qmake or CMake, and specify that Qt Creator ignores your build system.

Editors

Qt Creator comes with a code editor and an integrated Qt Designer for designing and building graphical user interfaces (GUIs) from Qt widgets.

Code Editor

As an IDE, Qt Creator differs from a text editor in that it knows how to build and run applications. It understands the **C++** and **QML** languages as code, not just as plain text. This allows it to:

- Enable you to write well formatted code
- Anticipate what you are going to write and complete the code
- Display inline error and warning messages
- Enable you to semantically navigate to classes, functions, and symbols
- Provide you with context-sensitive help on classes, functions, and symbols

- Rename symbols in an intelligent way, so that other symbols with the same name that belong to other scopes are not renamed
- Show you the locations in code where a function is declared or called

UI Designer

Qt Creator provides two integrated visual editors, Qt Designer and Qt Quick Designer.

Qt Designer is a tool for designing and building graphical user interfaces (GUIs) from Qt widgets. You can compose and customize your widgets or dialogs and test them using different styles and resolutions.

Widgets and forms created with Qt Designer are integrated seamlessly with programmed code, using the Qt signals and slots mechanism, that lets you easily assign behavior to graphical elements. All properties set in Qt Designer can be changed dynamically within the code. Furthermore, features like widget promotion and custom plugins allow you to use your own widgets with Qt Designer.

UIs that use widgets are clearly structured and enforce a platform look and feel, which makes them useful for traditional applications. However, they are static, and do not fully make use of the large high-resolution screens, touch input, and significant graphics power that are becoming common in portable consumer devices, such as mobile phones, media players, set-top boxes, and netbooks.

Qt Quick Designer allows you to easily develop animations by using a declarative programming language called QML. In QML, a user interface is specified as a tree of objects with properties.

You use a visual editor to create items, screens, and applications, as well as define changes in their state, transitions from one state to another, and user actions that change the states. Qt Quick Designer generates the necessary code for you.

You can use Qt or JavaScript to implement the application logic.

Languages

You can use the code editor to write code in Qt **C++** or in the **QML** declarative programming language.

QML

You can use QML to build highly dynamic, custom user interfaces from a rich set of QML elements. Qt Quick helps programmers and designers collaborate to build the fluid user interfaces that are becoming common in portable consumer devices, such as mobile phones, media players, set-top boxes and netbooks.

QML is an extension to JavaScript, that provides a mechanism to declaratively build an object tree of QML elements. QML improves the integration between JavaScript and Qt's existing QObject based type system, adds support for automatic property bindings and provides network transparency at the language level.

Targets

Qt Creator provides support for building and running Qt applications for **desktop** environment (Windows, Linux, and Mac OS) and **mobile** devices (Symbian, Maemo, and MeeGo). Build settings allow you to quickly switch between build targets.

When you build an application for a mobile device target with a device connected to the development PC, Qt Creator generates an installation package, installs in on the device, and executes it.

You can publish the installation packages on the Nokia Store. For Symbian devices, the packages must be signed.

Tools

Qt Creator is integrated with a set of helpful tools, such as version control systems and Qt Simulator.

Version Control Systems

The recommended way to build a project is to use a version control system. Qt Creator uses the version control system's command line clients to access your repositories. The following version control systems are supported:

- **Git**

- **Subversion**
- **Perforce**
- **CVS**
- **Mercurial**

The functions available to you in Qt Creator depend on the version control system. Basic functions are available for all the supported systems. They include comparing files with the latest versions stored in the repository and displaying the differences, viewing versioning history and change details, annotating files, and committing and reverting changes.

Qt Simulator

You can use the Qt Simulator to test Qt applications that are intended for mobile devices in an environment similar to that of the device. You can change the information that the device has about its configuration and environment.

The Qt Simulator is installed as part of the Qt SDK. After it is installed, you can select it as a build target in Qt Creator.

Debuggers

Qt Creator does not include a debugger. It provides a debugger plugin that acts as an interface between the Qt Creator core and external native debuggers:

- GNU Symbolic Debugger (**gdb**)
- Microsoft Console Debugger (**CDB**)
- internal Java Script debugger

Qt Creator displays the raw information provided by the native debuggers in a clear and concise manner with the goal to simplify the debugging process as much as possible without losing the power of the native debuggers. You can use the native debuggers to debug the C++ language.

You can connect mobile devices to your development PC and debug processes running on the devices.

Contact

Contact us on IRC on the freenode network in #qt-creator

If you prefer mail you can always subscribe to our mailing list [1]

Links

- [2] Gitorious
- Bugtracker
- [3] Qt Lab Blog