Setting Up a Generic Project

Generic project support allows you to use Qt Creator as a code editor. You can change the way your project is built by modifying the `make` command in the **Projects** mode under **Build Settings**.

When you import a project, Qt Creator creates the following files that allow you to specify which files belong to your project and which include directories or defines you want to pass to your compiler: `.files`, `.includes`, and `.config`.

Importing a Generic Project

To import an existing generic project:

1. Select **File** > **New Project** > **Import Project** > **Import Existing Project**.
2. In **Import Existing Project**, enter the project name and select the location of the project file you want to import.

   Qt Creator automatically generates the following files in the project directory:

   - .files
   - .includes
   - .config
   - .creator
   - .cflags
   - .cxxflags

When the project is successfully imported, Qt Creator creates the project tree in the sidebar.

After importing a generic project into Qt Creator, open it by selecting the `.creator` file.

Working with Generic Project Files

For a generic project, you have to manually specify which files belong to your project and which include directories or defines you want to pass to your compiler.

Specifying Files

The list of files for a generic project is specified in the `.files` file. When you first create a generic project, Qt

.files file. Alternatively, you can add and remove files or directories using the context menu in the project tree.

If you frequently need to update the `.files` file, you can do so efficiently by using a script that updates the file for you. If the file is modified externally, you have to restart Qt Creator for the changes to take effect.

To update the `.files` on the **Git** repository use the following script:

```
git ls-files "*.cpp" "*.h" > MyProject.files
```

Precompiled Headers

To use precompiled headers in a generic project, add the `pch` tag after a file path in the `.files` file, separated by the pipe character (|). For example:

```
src/pch.h|pch
```

Specifying Include Paths and Framework Paths

The include paths are specified in the `.includes` file, one include path per line. The paths can be either absolute or relative to the `.includes` file.

Lines starting with "-F" are interpreted as framework paths.

Specifying Defines

The defines are specified in the `.config` file. The `.config` file is a regular C++ file, prepended to all your source files when they are parsed. Only use the `.config` file to add lines as in the example below:

```
#define NAME value
```

Forwarding Flags to Clang Code Model

The `.cxxflags` and `.cflags` files contain command line flags for the Clang code model on a single line.

For example, specify the `-std=c++11` to set the language version for parsing as C++11.

Providing Deployment Information

If you want to run your application on a generic remote Linux device, you first need to deploy your executable and possibly other files. Qt Creator does that for you automatically if you provide the necessary information. This works the same way as explained for CMake in Deploying Applications to Generic Remote Linux Devices, except that you also need to include your application binary in the list.

Qt Creator cannot automatically determine which executable to run.

In the **Projects** mode under **Run Settings**, define the executable file to run:

1. Click **Add** and select **Custom Executable**.
2. Define the configuration name, the location of the executable, any additional arguments and the working directory.

< Setting Up an Autotools Project Setting Up Nimble >

Qt DOCUMENTATION

Feedback Sign In