

# Using QML Modules with Plugins

**QML modules** may use **C++ plugins** to expose components defined in C++ to QML applications.

To create a QML module:

1. Create custom components and place all the `.qml` files in a directory dedicated to your module. For example: `imports\asset_imports`.
2. For Qt Quick UI projects (`.qmlproject`), specify the path to the directory that contains the module in the `.qmlproject` file of the application where you want to use the module as a value of the `importPaths` variable. For example `importPaths: [ "imports", "asset_imports" ]`.
3. Create a `qmlDir` file for your module and place it in the module directory. For more information, see [Module Definition qmlDir Files](#).
4. Create a `qmltypes` file, as instructed in [Generating Type Description Files](#).
5. Create a directory named `designer` in your module directory.
6. Create a `.metainfo` file for your module and place it in the `designer` directory. Use a `metainfo` file delivered with Qt, such as `qtquickcontrols2.metainfo`, as an example.
7. Import the module into the project, as instructed in [Importing QML Modules](#).

## Generating Type Description Files

When [registering QML types](#), make sure that the QML module has a `plugins.qmltypes` file. Ideally, it should be located in the same directory as the `qmlDir` file. The `qmltypes` file contains a description of the components exported by the module's plugins and is loaded by Qt Creator when the module is imported.

For more information, see [Type Description Files](#).

## Dumping Plugins Automatically

If a module with plugins lacks the `qmltypes` file, Qt Creator tries to generate a temporary file itself by running the `qmlDump` program in the background. However, this automatic dumping is a fallback mechanism with many points of failure and you cannot rely upon it.

## Importing QML Modules

By default, Qt Creator will look in the QML import path of Qt for QML modules.

If you use `qmake` and your application adds additional import paths that Qt Creator should use, specify them using `QML_IMPORT_PATH` in the `.pro` file of your application: `QML_IMPORT_PATH += path/to/module`.

If you use `CMake`, add the following command to the `CMakeLists.txt` file to set the QML import path:

The import path affects all the targets built by the CMake project.

< UI Files

Developing Widget Based Applications >

© 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.



Contact Us

Company

- About Us
- Investors
- Newsroom
- Careers
- Office Locations

Licensing

- Terms & Conditions
- Open Source
- FAQ

Support

- Support Services
- Professional Services
- Partners
- Training

For Customers

- Support Center
- Downloads
- Qt Login
- Contact Us
- Customer Success

Community

- Contribute to Qt
- Forum
- Wiki
- Downloads
- Marketplace

