Converting UI Projects to Applications

Qt Quick UI projects are useful for creating user interfaces. To use them for application development in Qt Creator you have to add:

- Project configuration file (CMakeLists.txt or .pro)
- C++ code (.cpp)
- Resource files
- Code needed for deploying applications to devices

For more information about integrating QML and C++, see Overview - QML and C++ Integration.

> **Note:** Since Qt Design Studio 2.3.0, Qt Design Studio project wizard templates generate projects that can be built with CMake. You can open the *CMakeLists.txt* project file in Qt Creator to continue developing the project.

For more information, see Designer-Developer Workflow.

If you want to use qmake as the build system, you can use a Qt Creator wizard template to create a Qt Quick application that is built using the qmake build system and then copy the source files from the Qt UI Quick project to the application project.

You can use the `RESOURCES` option in the project configuration file to automatically add all the QML files and related assets to a Qt resource collection file (.qrc). However, large files should be included as external binary resources instead of compiling them into the binary.

The wizard automatically adds the `QML_IMPORT_PATH` option to the project file for specifying the required QML import path. The path is only needed if more than one subdirectory contains QML files.

Then you can use the QQuickView class in the main C++ source file to show the main QML file when the application starts.

The *Qt Quick Designer Components* module is installed when you install Qt Design Studio. If you use Qt Quick Studio Components or Effects from the module in a project that you want to edit in Qt Creator, you have to build the module and install it to your Qt to be able to build your project. For more information, see Adding Qt Quick Designer Components to Qt Installations.

The Qt Quick Timeline module is installed when you install Qt Design Studio. If you only install Qt Creator and Qt, remember to also select the Qt Quick Timeline module for installation. If your Qt is older than 5.14, you must build the Qt Quick Timeline module and install it to your Qt to be able to build your project. For more information, see Adding Qt Quick Timeline Module to Qt Installations.

Converting into qmake Projects

To convert a project that has a .qmlproject file to one that has a .pro file:

1. Select **File** > **New Project** > **Application (Qt)** > **Qt Quick Application** > **Choose**.
2. In the **Build system** field, select qmake as the build system to use for building and running the project, and then select **Next** (or **Continue** on macOS).
3. Follow the instructions of the wizard to create the project.
4. In the file explorer, copy the source files from the Qt Quick UI project directory to a subdirectory in the application project directory. For the purpose of these instructions, the directory is called `qml`.
5. Open the application project file, and edit the value of the `RESOURCES` option to add the following line:

6. Also edit the value of the QML_IMPORT_PATH option to specify the QML import path:

```
QML_IMPORT_PATH = qml/imports
```

   Where `qml/imports` is the import path.

7. Select **Build** > **Run qmake** to apply the RESOURCES option to the build configuration.
8. Open the *main.cpp* file and replace the QQmlApplicationEngine object with a QQuickView object:

```
QQuickView view;
view.engine()->addImportPath("qrc:/qml/imports");
view.setSource(QUrl("qrc:/qml/ProgressBar.ui.qml"));
if (!view.errors().isEmpty())
    return -1;
view.show();
```

   Where `qrc:/qml/imports` is the import path and `qrc:/qml/ProgressBar.ui.qml` is the path to and the name of the main QML file in the Qt Quick UI project.

9. Select **Build** > **Run** to build and run your project.

For example, if you copy the source files of the *ProgressBar* example from your Qt Design Studio installation (located in the `\share\qtcreator\examples\ProgressBar` directory) to an empty Qt Quick application project and make the necessary changes, the *main.cpp* file should look as follows:

```
#include <QGuiApplication>
#include <QQmlApplicationEngine>
#include <QQuickView>

int main(int argc, char *argv[])
{
    QCoreApplication::setAttribute(Qt::AA_EnableHighDpiScaling);

    QGuiApplication app(argc, argv);

    QQuickView view;
    view.engine()->addImportPath("qrc:/qml/imports");
    view.setSource(QUrl("qrc:/qml/ProgressBar.ui.qml"));
    if (!view.errors().isEmpty())
        return -1;
    view.show();

    app.exec();
}
```

Handling Large Data Files

Graphical assets used in the UI, such as images, effects, or 3D scenes are a typical cause for performance problems in UIs. Even building the application requires huge amounts of memory if you try to include large asset files, such as 100-MB 3D models or 64-MB textures, into the `.qrc` file for compiling them into the binary.

First try to optimize your assets, as described in Optimizing Designs and Creating Optimized 3D Scenes.

Adding Custom Fonts

To use custom fonts from the Qt Quick UI project, call the QFontDatabase::addApplicationFont() function from the *main.cpp* file.

Adding Qt Quick Designer Components to Qt Installations

If you use Qt Quick Studio Components or Effects in your project, you have to check out and install the *Qt Quick Designer Components* module from Qt Code Review.

For example:

```
git clone https://code.qt.io/qt-labs/qtquickdesigner-components.git
```

Then use qmake from your Qt installation to build the module and to add it to your Qt. Switch to the directory that contains the sources (usually, qtquickdesigner-components), make sure you checkout the qmake branch, and enter the following commands:

```
<path_to_qmake>\qmake -r
make
make install
```

On Windows, use the `nmake` and `nmake install` commands instead.

If you prefer CMake instead and you want to benefit from the QML compilation, then you can checkout the dev branch instead. CMake is only supported since Qt 6.2. Enter the following commands:

```
mkdir build
cd build
cmake -GNinja -DCMAKE_INSTALL_PREFIX=<path_to_qt_install_directory> <path_to_qtquickdesigner-components>
cmake --build .
cmake --install .
```

Adding Qt Quick Timeline Module to Qt Installations

> **Note:** You only need to do this if your Qt version is older than 5.14.

Check out the Qt Quick Timeline module from Qt Code Review.

For example:

```
git clone "https://codereview.qt-project.org/qt/qtquicktimeline"
```

To use qmake, you need to check out a branch or tag that contains the qmake configuration files.

For example:

```
git checkout v5.15.2
```

Then build the module and add it to your Qt as described in the previous section.

‹ Converting Qt 5 Projects into Qt 6 Projects

Using External Tools ›