将 UI 项目转换为应用程序

Qt 快速 UI 项目对于创建用户界面非常有用。要将它们用于Qt Creator中的应用程序开发，您必须添加：

- 项目配置文件（清单.txt或 .pro）
- C++代码 （.cpp）
- 资源文件
- 将应用程序部署到设备所需的代码

有关集成 QML 和C++的更多信息，请参阅概述 - QML 和C++集成。

注意： 从Qt设计工作室2.3.0开始，Qt设计工作室项目向导模板生成可以使用CMake构建的项目。您可以在 Qt 创建器中打开 *CMakeLists.txt*项目文件以继续开发项目。

有关使用 Qt 设计工作室创建项目的更多信息，请参阅 Qt 设计工作室手册。

如果要使用 qmake 作为生成系统，可以使用 Qt 创建器向导模板创建使用 qmake 生成系统生成的 Qt 快速应用程序，然后将源文件从 Qt UI 快速项目复制到应用程序项目。

可以使用项目配置文件中的选项自动将所有 QML 文件和相关资产添加到 Qt 资源收集文件 （.qrc）。但是，大文件应作为外部二进制资源包含在内，而不是将它们编译到二进制文件中。RESOURCES

该向导会自动将该选项添加到项目文件中，以指定所需的 QML 导入路径。仅当多个子目录包含 QML 文件时，才需要该路径。QML_IMPORT_PATH

然后，可以使用主C++源文件中的 QQuickView 类在应用程序启动时显示主 QML 文件。

*Qt快速设计器组件*模块是在安装Qt设计工作室时安装的。如果要在 Qt Creator 中编辑的项目中的模块中使用 Qt Quick Studio 组件或效果，则必须生成该模块并将其安装到 Qt 中才能生成项目。有关更多信息，请参见将 Qt 快速设计器组件添加到 Qt 安装中。

Qt快速时间轴模块是在安装Qt设计工作室时安装的。如果只安装 Qt 创建器和 Qt，请记住还要选择 Qt 快速时间轴模块进行安装。如果您的Qt早于5.14，您必须构建Qt快速时间轴模块并将其安装到Qt上才能构建项目。有关更多信息，请参阅 将 Qt 快速时间线模块添加到 Qt 安装。

转换为项目

要将具有 .qmlproject 文件的项目转换为具有 .pro 文件的项目，请执行以下操作：

1. 选择“**文件**>**新项目**>**应用程序 （Qt）** > **Qt 快速应用程序**>**选择**”。
2. 在“**生成系统**”字段中，选择 qmake 作为用于生成和运行项目的生成系统，然后选择“**下一步**”（或在 macOS 上**继续**）。
3. 按照向导的说明创建项目。
4. 在文件资源管理器中，将源文件从 Qt Quick UI 项目目录复制到应用程序项目目录中的子目录中。出于这些说明的目的，该目录称为 。qml
5. 打开应用程序项目文件，然后编辑该选项的值以添加以下行：RESOURCES

```
RESOURCES += \
    $$files(qml/*)
```

```
QML_IMPORT_PATH = qml/imports
```

其中 是导入路径。qml/imports

7. 选择“**生成**>**运行 qmake**”以将该选项应用于生成配置。RESOURCES
8. 打开*主.cpp*文件，并将 QQml 应用程序引擎对象替换为 Q快速视图对象：

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

其中 是导入路径， 是 Qt 快速 UI 项目中主 QML 文件的路径和名称。
qrc:/qml/importsqrc:/qml/ProgressBar.ui.qml

9. 选择“**生成**>**运行**”以生成并运行项目。

例如，如果将 *ProgressBar* 示例的源文件从 Qt 设计工作室安装（位于目录中）复制到空的 Qt 快速应用程序项目并进行必要的更改，*则主.cpp*文件应如下所示：\share\qtcreator\examples\ProgressBar

```
#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 file for compiling them into the binary..qrc

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

If you really need large assets, they should either be loaded directly from the file system or use the Qt resource system in the dynamic way. For more information, see The Qt Resource System in the Qt documentation.

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 and commands instead.`nmake` `nmake install`

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.

Qt DOCUMENTATION