搜索

构建通用项目类型

本章介绍如何为基于Qt的三种常见项目类型设置qmake项目文件：应用程序、库和插件。尽管所有项目类型都使用许多相同的变量，但每个项目类型都使用特定于项目的变量来自定义输出文件。

此处不介绍特定于平台的变量。有关更多信息，请参阅 Qtfor Windows - Deployment和Qt for macOS。

构建应用程序

模板告诉 qmake 生成一个将构建应用程序的生成文件。使用此模板，可以通过向CONFIG变量定义添加以下选项之一来指定应用程序的类型：app

选择	描述
窗户	该应用程序是一个 Windows GUI 应用程序。
安慰	app仅限模板：应用程序是 Windows 控制台应用程序。
测试用例	该应用程序是自动测试。

使用此模板时，将识别以下 qmake 系统变量。应在 .pro 文件中使用这些文件来指定有关应用程序的信息。有关其他依赖于平台的系统变量，您可以查看平台说明。

- 标头 - 应用程序的头文件列表。
- 源 - 应用程序的C++源文件的列表。
- FORMS- 应用程序的UI文件列表（使用Qt Designer创建）。
- LEXSOURCES- 应用程序的 Lex 源文件列表。
- YACCSOURCES- 应用程序的Yacc源文件列表。
- 目标 - 应用程序的可执行文件的名称。这默认为项目文件的名称。（扩展名（如果有）会自动添加）。
- DESTDIR - 放置目标可执行文件的目录。
- 定义 - 应用程序所需的任何其他预处理器定义的列表。
- 包含路径 - 应用程序所需的任何其他包含路径的列表。
- 依赖路径 - 应用程序的依赖关系搜索路径。
- VPATH- 查找所提供文件的搜索路径。
- DEF_FILE - 仅限 Windows：要为应用程序链接的 .def 文件。

```
TEMPLATE = app
DESTDIR  = c:/helloapp
HEADERS += hello.h
SOURCES += hello.cpp
SOURCES += main.cpp
DEFINES += USE_MY_STUFF
CONFIG  += release
```

对于单值项目，例如模板或目标目录，我们使用"=";但是对于多值项，我们使用"+="*添加到该*类型的现有项中。使用"="将变量值替换为新值。例如，如果我们编写，则所有其他定义都将被删除。
DEFINES=USE_MY_STUFF

构建测试用例

测试用例项目是一个旨在作为自动化测试运行的项目。任何可以通过将值添加到变量来标记为测试用例。
appapptestcaseCONFIG

对于测试用例项目，qmake 会将目标插入到生成的 Makefile 中。此目标将运行应用程序。如果测试以等于零的退出代码终止，则认为测试通过。check

目标通过SUBDIRS项目自动递归。这意味着可以从 SUBDIRS 项目中发出命令来运行整个测试套件。

Topics >

变量	描述
测试运行者	每个测试命令前面都有一个命令或外壳片段。一个示例用例是"超时"脚本，如果测试未在指定时间内完成，它将终止测试。
特斯塔格斯	附加到每个测试命令的其他参数。例如，传递其他参数以设置测试中的输出文件和格式（例如 QTestLib 支持的选项）可能很有用。-o filename,format

注意：变量必须在调用工具时设置，而不是在 .pro 文件中设置。大多数工具支持直接在命令行上设置 Makefile 变量：makemake

```
# Run tests through test-wrapper and use JUnit XML output format.
# In this example, test-wrapper is a fictional wrapper script which terminates
# a test if it does not complete within the amount of seconds set by "--timeout".
# The "-o result.xml,junitxml" options are interpreted by QTestLib.
make check TESTRUNNER="test-wrapper --timeout 120" TESTARGS="-o result.xml,junitxml"
```

测试用例项目可以使用以下选项进一步自定义：CONFIG

测试用例通常使用QTest或编写，但这不是使用 and 的要求。唯一的主要要求是测试程序在成功时退出时退出代码为零，失败时退出代码为非零退出代码。TestCaseCONFIG+=testcasemake check

构建库

模板告诉qmake生成一个将构建库的Makefile。使用此模板时，除了模板支持的系统变量外，还支持VERSION变量。使用 .pro 文件中的变量指定有关库的信息。libapp

使用模板时，可以将以下选项添加到CONFIG变量中，以确定构建的库的类型：lib

选择	描述
.dll	该库是共享库 （dll）。
静态库	该库是一个静态库。
.plugin	该库是一个插件。

还可以定义以下选项以提供有关库的其他信息。

- 版本 - 目标库的版本号。例如，2.3.1。

库的目标文件名取决于平台。例如，在 X11、macOS 和 iOS 上，库名称将以 为前缀。在 Windows 上，不会向文件名添加前缀。lib

构建插件

插件是使用模板构建的，如上一节所述。这告诉qmake为项目生成一个Makefile，该项目将为每个平台构建一个适合形式的插件，通常以库的形式。与普通库一样，VERSION变量用于指定有关插件的信息。lib

- 版本 - 目标库的版本号。例如，2.3.1。

构建Qt设计器插件

*Qt Designer*插件是使用一组特定的配置设置构建的，这些设置取决于Qt为您的系统配置的方式。为方便起见，可以通过添加到QT变量来启用这些设置。例如：designer

```
QT          += widgets designer
```

有关基于插件的项目的更多示例，请参阅Qt 设计器示例。

在调试和发布模式下生成和安装

有时，需要在调试和发布模式下生成项目。尽管CONFIG变量可以同时保存这两个选项，但只应用最后指定的选项。debugrelease

两种模式的构建

```
CONFIG += debug_and_release

CONFIG(debug, debug|release) {
    TARGET = debug_binary
} else {
    TARGET = release_binary
}
```

上述代码片段中的范围在每种模式下修改生成目标，以确保生成的目标具有不同的名称。为目标提供不同的名称可确保一个不会覆盖另一个。

当 qmake 处理项目文件时，它将生成一个生成文件规则，以允许在两种模式下构建项目。可以通过以下方式调用它：

```
make all
```

可以将该选项添加到项目文件中的变量中，以确保默认情况下在两种模式下生成项目：build_allCONFIG

```
CONFIG += build_all
```

这允许使用默认规则处理生成文件：

```
make
```

在两种模式下安装

该选项还可确保在调用安装规则时安装目标的两个版本：build_all

```
make install
```

可以根据目标平台自定义构建目标的名称。例如，一个库或插件的命名方式可能与 Unix 平台上使用的约定不同：

```
CONFIG(debug, debug|release) {
    mac: TARGET = $$join(TARGET,,,_debug)
    win32: TARGET = $$join(TARGET,,d)
}
```

‹ 创建项目文件 运行 qmake ›

联系我们

公司

关于我们
投资者
编辑部
职业
办公地点

发 牌

条款和条件
开源
常见问题

支持

支持服务
专业服务
合作 伙伴
训练

对于客户

支持中心
下载
Qt登录
联系我们
客户成功案例

社区

为Qt做贡献
论坛
维基
下载
市场

反馈 登录