搜索

构建 QML 应用程序

在生成C++控制台应用程序中，我们展示了简单控制台应用程序的 CMakeLists.txt 文件。我们现在将对其进行扩展，以创建一个使用Qt Quick 模块的 QML应用程序。

这是完整的项目文件：

```
cmake_minimum_required(VERSION 3.16)

project(hello VERSION 1.0 LANGUAGES CXX)
```



```
find_package(Qt6 6.2 COMPONENTS Quick Gui REQUIRED)

qt_add_executable(myapp
    main.cpp
)

qt_add_qml_module(myapp
    URI hello
    VERSION 1.0
    QML_FILES
        main.qml
        FramedImage.qml
    RESOURCES
        img/world.png
)

target_link_libraries(myapp PRIVATE Qt6::Gui Qt6::Quick)
```

让我们来看看我们所做的更改。我们指定CMAKE_AUTOMOC、CMAKE_CXX_STANDARD和CMAKE_CXX_STANDARD_REQUIRED。

```
set(CMAKE_AUTOMOC ON)
```

在调用中，我们替换。这将找到模块并提供我们稍后链接的目标。`find_packageCoreQuickQt6QuickQt6::Quick`

```
find_package(Qt6 6.2 COMPONENTS Quick Gui REQUIRED)
```

请注意，应用程序仍将链接到它，因为依赖于它。`Qt6::CoreQt6::Quick`

qt_add_executable创建并最终确定应用程序目标：

```
qt_add_executable(myapp
    main.cpp
)
```

qt_add_qml_module传递可执行文件的目标、URI、模块版本和 QML 文件列表，以确保 myapp 成为 QML 模块。除此之外，这会将 QML 文件放入资源文件系统中。`qrc:/${URI}`

```
qt_add_qml_module(myapp
    URI hello
    VERSION 1.0
    QML_FILES
        main.qml
        FramedImage.qml
    RESOURCES
        img/world.png
)
```

首先，确保运行。其次，它创建一个目标，该目标在QML_FILES中运行文件。`qt_add_qml_moduleqmlcachegenmyapp_qmllintqmllint`

通过添加引用的资源，它们会自动添加到与 QML 文件相同的根路径下的应用程序 - 也在资源文件系统中。通过保持资源系统中的路径与源和构建目录中的路径一致，我们确保始终找到图像，因为它是相对于 FramedImage.qml 解析的。如果我们从那里加载 main.qml，它指的是资源文件系统中的图像，或者如果我们使用该工具查看它，它指的是实际文件系统中的图像。`qml`

在命令中，我们链接反对。`target_link_librariesQt6::QuickQt6::Core`

```
target_link_libraries(myapp PRIVATE Qt6::Gui Qt6::Quick)
```

‹ CMake 入门

构建可重用的 QML 模块 ›

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