

[Qt 6.4](#) > [使用 CMake 构建](#) > [构建可重用的 QML 模块](#)

# 构建可重用的 QML 模块

下面的示例演示如何创建向 QML 公开 C++ 库。该示例的目录结构如下所示：

```
├── CMakeLists.txt
├── example
│   └── mylib
│       ├── CMakeLists.txt
│       ├── mytype.cpp
│       └── mytype.h
```

[Topics](#) >

换为斜杠。这与引擎在[导入路径](#)中搜索模块时使用的逻辑相同。de 声明一个类并使用声明性注册宏将其公开给引擎。CMakeLists.txt 添加 `add_subdirectory(mytype.h)`

在子目录中我们再次调用。但是，调用略有不同：CMakeLists.txt 添加 `qt6_add_qml_module`

```
qt6_add_qml_module(mylib
    URI example.mylib
    VERSION 1.0
    SOURCES
        mytype.h mytype.cpp
)
```

若要添加 C++ 类型，需要指定 `SOURCES` 参数。未创建 `mylib` 的目标。因此，如果传递给的目标不存在，则会自动创建一个库目标，在这种情况下是必需的。`qt6_add_qml_module`

构建项目时，除了库之外，还会构建 QML 插件。插件自动生成的类从 `extension` 扩展自。mylib 库本身已经包含向引擎注册类型的代码。但是，这仅在我们可以链接到库的情况下有用。为了使模块在 [QML 运行时工具](#) 加载的 QML 文件中可用，需要一个可以加载的插件。然后，插件负责实际链接到库，并确保类型被注册。

`QmlEngineExtensionPlugin.qml`

请注意，仅当模块除了注册类型之外不执行任何操作时，才能自动生成插件。如果它需要做一些更高级的事情，比如注册图像提供者，你仍然需要手动编写插件。`qt6_add_qml_module` 对此表示支持。

`initializeEngineNO_GENERATE_PLUGIN_SOURCE`

此外，请确保在构建目录中正确配置了 CMake，这通常意味着你可以将构建目录的路径传递给

在结束之前，将 QML 文件添加到模块中。在 `mylib` 文件夹下，添加一个文件 `Mistake.qml`

```
import example.mylib

MyType{
    answer: 43
}
```

并调整呼叫：`qt6_add_qml_module`

```
qt6_add_qml_module(mylib
    URI example.mylib
    VERSION 1.0
    SOURCES
        mytype.h mytype.cpp
    QML_FILES
        Mistake.qml
)
```

如前所述，我们犯了一个错误，因为它实际上是一个只读属性。这说明了集成：CMake 创建一个目标，一旦我们运行它，就会警告这个问题：`answerqmlintqmlintqmlint`

```
$> cmake --build . --target mylib_qmlint
...
Warning: Mistake.qml:4:13: Cannot assign to read-only property answer
    answer: 43
           ^^
```

[◀ 构建 QML 应用程序](#)

[在命令行上生成项目 ▶](#)

©2022 Qt 有限公司 此处包含的文档贡献的版权归 他们各自的所有者。此处提供的文档根据自由软件基金会发布的 [GNU 自由文档许可证版本 1.3](#) 的条款进行许可。Qt 和相应的徽标是 Qt 有限公司在芬兰和/或其他国家/地区的 [商标](#) 全球。所有其他商标均为其各自所有者的财产。



联系我们



投资者  
编辑部  
职业  
办公地点

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