搜索

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
```


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.txtadd_subdirectorymytype.h

在子目录中我们再次调用。但是，调用略有不同：CMakeLists.txtqt6_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 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。QQmlEngineExtensionPluginqml

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

在结束之前，将 QML 文件添加到模块中。在 .h 文件夹中，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：answerqmllintqmllintqmllint

```
$> cmake --build . --target mylib_qmllint
...
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 ›