

[Qt 6.4](#) > [使用 CMake 构建](#) > [Qt 5 和 Qt 6 兼容性](#)

Qt 5 和 Qt 6 兼容性

Qt 5 和 Qt 6 中 CMake API 的语义在很大程度上是兼容的。但是，在 Qt 5.14 之前，所有导入的 Qt 库目标和命令都包含版本号作为名称的一部分。这使得编写应该同时适用于 Qt 5 和 Qt 6 的 CMake 代码有些麻烦。因此，Qt 5.15 引入了无版本目标和命令，以便编写与不同 Qt 版本基本无关的 CMake 代码。

无版本目标

除了现有的导入目标外，Qt 5.15 还引入了无版本目标。也就是说，要链接到 Qt Core，可以同时引用，或者：
Qt6::CoreQt::Core

[Topics](#) >

```
find_package(Qt5 5.15 REQUIRED COMPONENTS Core)
endif()

add_executable(helloworld
    ...
)

target_link_libraries(helloworld PRIVATE Qt::Core)
```

上面的代码片段首先尝试查找 Qt 6 安装。如果失败，它会尝试查找 Qt 5.15 软件包。无论使用的是 Qt 6 还是 Qt 5，我们都可以使用导入的目标。Qt::Core

默认情况下定义无版本目标。在第一次调用之前设置 QT_NO_CREATE_VERSIONLESS_TARGETS 以禁用它们。
find_package()

注意：导入的 Qt::Core 目标将不具有 Qt6::Core 目标中可用的目标属性。

无版本命令

从 Qt 5.15 开始，Qt 模块还提供了其命令的无版本变体。例如，您现在可以使用 qt_add_translation 来编译翻译文件，无论您使用的是 Qt 5 还是 Qt 6。

在第一次调用之前设置 QT_NO_CREATE_VERSIONLESS_FUNCTIONS 以防止创建无版本命令。

```
find_package()
```

可能有些项目需要在一个CMake上下文中同时加载Qt 5和Qt 6（尽管不支持在一个库或可执行文件中混合Qt版本，所以要小心）。

在这样的设置中，无版本目标和命令将隐式引用通过 `a` 找到的第一个 Qt 版本。在第一次调用之前设置 `QT_DEFAULT_MAJOR_VERSION` CMake 变量以使版本显式。 `find_package`

支持较旧的Qt 5版本

如果您还需要支持早于 Qt 5.15 的 Qt 5 版本，您可以通过将当前版本存储在 CMake 变量中来实现：

```
find_package(QT NAMES Qt6 Qt5 REQUIRED COMPONENTS Core)
find_package(Qt${QT_VERSION_MAJOR} REQUIRED COMPONENTS Core)

add_executable(helloworld
    ...
)

target_link_libraries(helloworld PRIVATE Qt${QT_VERSION_MAJOR}::Core)
```

在这里，我们尝试找到第一个Qt 6，如果失败，则在名称下找到Qt 5。如果找到其中任何一个，将成功，并且 CMake 变量将定义为任一或。

`find_package(<PackageName>...)QTfind_packageQT_VERSION_MAJOR56`

然后，我们通过动态创建名称再次加载确定的Qt版本的包。这是必需的，因为期望包名称为“非此即彼”，否则将打印错误。 `Qt${QT_VERSION_MAJOR}CMAKE_AUTOMOCQt5Qt6`

我们可以使用相同的模式来指定导入库的名称。在调用之前，CMake 将解析为任一。

`target_link_librariesQt${QT_VERSION_MAJOR}::WidgetsQt5::WidgetsQt6::Widgets`

推荐做法

尽可能使用 CMake 命令的无版本变体。

无版本导入的目标对于需要使用 Qt 5 和 Qt 6 进行编译的项目最有用。由于缺少目标属性，因此不建议默认使用它们。

如果您需要支持早于 Qt 5.15 的 Qt 5 版本，或者无法控制是否在可能定义

`QT_NO_CREATE_VERSIONLESS_FUNCTIONS`或`QT_NO_CREATE_VERSIONLESS_TARGETS`的上下文中加载 CMake 代码，请使用 CMake 命令和目标版本化版本。在这种情况下，您仍可以通过变量确定实际命令或目标名称来简化代码。

Windows 中的 Unicode 支持

在Qt 6中，默认情况下为链接到Qt模块的目标设置了编译器定义。这与 qmake 行为一致，但与 Qt 5 中的 CMake API 行为相比，这是一个变化。 `UNICODE_UNICODE`

在目标上调用`qt_disable_unicode_defines ()` 以不设置定义。

```
find_package(Qt6 COMPONENTS Core)

add_executable(helloworld
```

```
qt_disable_unicode_defines(helloworld)
```

< 导入的目标

CMake 命令参考 >

©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

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



