# 1.生成dll，创建一个项目，选择library

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

然后点击next-》next-》ok，会生成如下的文件

|  |
| --- |
|  |

#### 右击Sources文件夹，-》新建-》选择Qt-》qt resourcefile（qrc）

|  |
| --- |
|  |

然后填写名称

|  |
| --- |
|  |

ok-》ok出现如下界面

|  |
| --- |
|  |

#### 关闭这个界面，并且右击生成的Plugin.qrc，添加Resource文件，选择Qml File

|  |
| --- |
|  |

填写qml文件的名称-》ok

|  |
| --- |
|  |

点击下一步，再点击完成，面如下：

|  |
| --- |
|  |

#### 在MyButton.qml中添加代码

|  |
| --- |
| import QtQuick 2.0  import QtQuick.Controls 2.15  Button{  width: 100  height: 60  text: "Button1"    } |

#### 在myqmlwidget\_plugin.cpp中注册控件给外部使用

|  |
| --- |
| #include "myqmlwidget\_plugin.h"  #include "myqmlwidget.h"  #include <qqml.h>  void MyQmlwidgetPlugin::***registerTypes***(const char \***uri**)  {  // @uri MyQmlWidget  qmlRegisterType<MyQmlWidget>(uri, 1, 0, "MyQmlWidget");  qmlRegisterType(QUrl("qrc:/MyButton.qml"),uri,1,0,"MyButton");  } |

#### 然后生成dll，点击QtCreator的项目按钮，取消勾选shadow build选项，然后点击build生成dll

# 2、使用生成的dll

创建一个qt quick2项目，叫做testQmlDll

|  |
| --- |
|  |

#### 先编译一下这个项目，取消勾选shadow build选项，然后点击build生成，这样子输出文件夹也在源码文件夹里面，如下图

|  |
| --- |
|  |

#### 在这个目录的Release项目项目创建一个文件夹，名字必须和控件的名称一样

|  |
| --- |
|  |

#### 将上面生成的dll和qmldir(在输出目录的根目录下面)文件拷贝到这个目录下面

|  |
| --- |
|  |

#### 在test项目在使用

|  |  |
| --- | --- |
| import QtQuick 2.15  import QtQuick.Window 2.15  import MyQmlWidget 1.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Hello World")  MyButton{  }  } |  |

# 使用打包后的qml控件，在为控件添加属性的时候会报错，但是程序有能够正常运行，这个怎么解决？

|  |
| --- |
|  |

效果：

|  |
| --- |
| 点击按钮也有输出 |

解决方法：

1.将输出的控件拷贝到Qt对应的工具链的目录中，这个方法不太好

2.在需要用到该dll的项目的.pro文件中添加dll的路径，注意必须是模块的父路径，可以使用绝对路径或者相对路径

|  |
| --- |
| #testQmlDll.pro  QT += quick  # You can make your code fail to compile if it uses deprecated APIs.  # In order to do so, uncomment the following line.  #DEFINES += QT\_DISABLE\_DEPRECATED\_BEFORE=0x060000 # disables all the APIs deprecated before Qt 6.0.0  SOURCES += \  main.cpp  RESOURCES += qml.qrc  # Additional import path used to resolve QML modules in Qt Creator's code model  QML\_IMPORT\_PATH =F:\Projects\qt-project2022\qmlBookDemos\testQmlDll\release #路径  # Additional import path used to resolve QML modules just for Qt Quick Designer  QML\_DESIGNER\_IMPORT\_PATH =  # Default rules for deployment.  qnx: target.path = /tmp/$${TARGET}/bin  else: unix:!android: target.path = /opt/$${TARGET}/bin  !isEmpty(target.path): INSTALLS += target |

这样子做了还不够，无效属性的报错仍然没有消除，

1.将上面的文件夹复杂到qt工具链下面的qml命令里面。

2.需要使用qmlplugindump命令

|  |
| --- |
|  |

格式：qmlplugindump dll组件名称 主版本号.次版本号 组件模块的路径/plugins.qmltypes。注意需要将这个dll生成为release版本

如：

|  |
| --- |
| qmlplugindump.exe Qmlwidget 1.0 F:\Projects\qt-project2022\qmlBookDemos\testQmlDll\release\Qmlwidget\plugins.qmltypes |

此时在命令窗口有这个文件的内容。如果在对应的文件夹没有生成这个文件，可以手动创建这个文件，并且将命令行窗口的内容复制进去。

内容如下：

|  |
| --- |
| Module {  dependencies: ["QtQuick 2.0"]  Component {  name: "QmlWidget"  defaultProperty: "data"  prototype: "QQuickItem"  exports: ["QmlWidget 1.0"]  exportMetaObjectRevisions: [0]  }  Component {  prototype: "QQuickButton"  name: "MyButton 1.0"  exports: ["MyButton 1.0"]  exportMetaObjectRevisions: [0]  isComposite: true  defaultProperty: "data"  }  } |

此外，需要在exports的内容前面添加模块路径,修改后的内容：

|  |
| --- |
| Module {  dependencies: ["QtQuick 2.0"]  Component {  name: "QmlWidget"  defaultProperty: "data"  prototype: "QQuickItem"  exports: ["QmlWidget/QmlWidget 1.0"]  exportMetaObjectRevisions: [0]  }  Component {  prototype: "QQuickButton"  name: "MyButton 1.0"  exports: ["QmlWidget/MyButton 1.0"]  exportMetaObjectRevisions: [0]  isComposite: true  defaultProperty: "data"  }  } |