

Qt学习目录

Qt学习目录

- 1、Qt的几个名词和概念
- 2、视频02：创建一个Qt工程helloworld
- 3、视频03：项目运行发布和设置图标
 - 1、配置电脑Qt环境
 - 2、release版本与Debug版本
 - 3、问题：文件打包给别人，却因为没有Qt后者没有加入相关环境而导致无法打开
 - 4、添加自动图标
- 4、视频04：项目文件组成及分析
 - 1、QWidget
 - 2、MyFirst_Widget.pro
 - 3、信号与槽
 - 4、main文件解析
 - 5、Widget.h文件解析
 - 6、widget.cpp文件解析

1、Qt的几个名词和概念

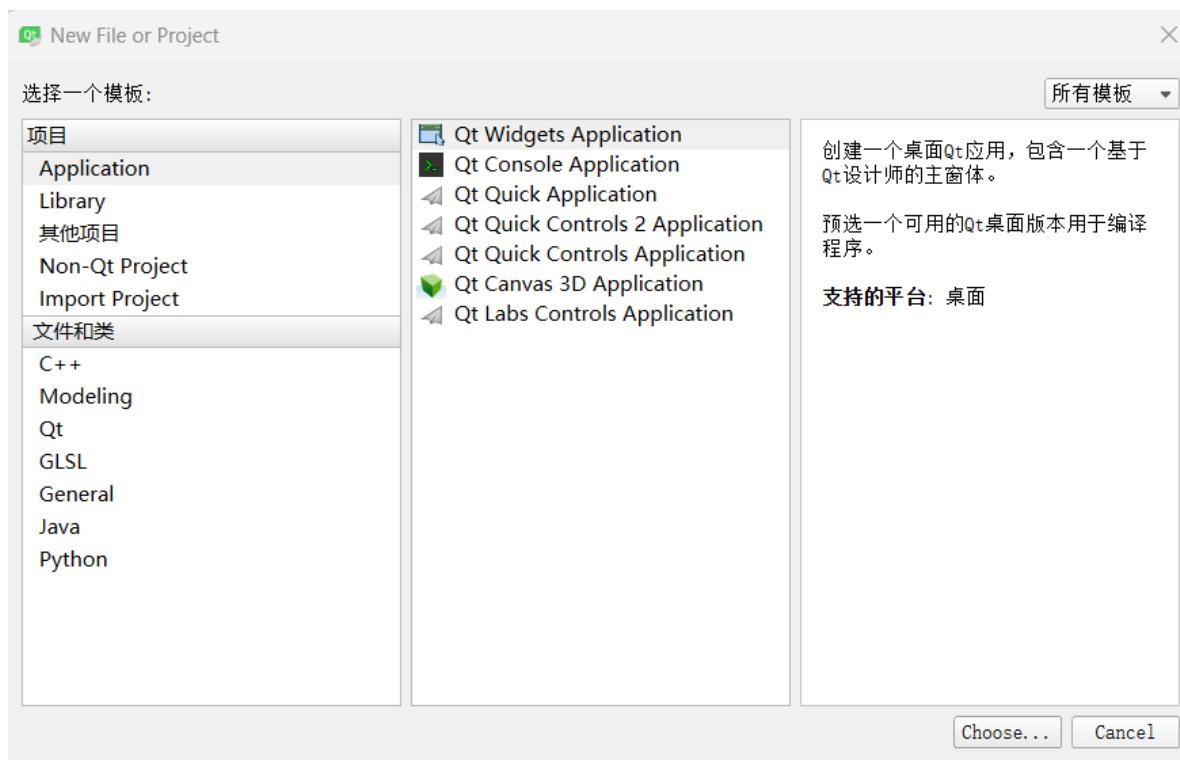
Qt：一个跨平台的C++**图形用户界面应用程序框架**，支持Win、Mac和Unix。

Qt Creator：IDE，集成开发环境，类似于Visual Studio，包括C++编辑器、项目生成管理工具、图形界面的调试器等

Qt Designer：专门用来设计图形界面的。包含在Qt Creator里面。

2、视频02：创建一个Qt工程helloworld

- 1、选择创建 Qt Widgets Application：窗体小部件；并选择项目位置；



- Location
- Kits
- Details
- 汇总

项目介绍和位置

This wizard generates a Qt Widgets Application project. The application derives by default from QApplication and includes an empty widget.

名称:

创建路径:

☐ 设为默认的项目路径

下一步 (N) 取消



- Location
- Kits
- Details
- 汇总

Kit Selection

Qt Creator can use the following kits for project **untitled**:

☒ Select all kits

☒ Desktop Qt 5.6.1 MinGW 32bit

详情 ▾

下一步 (N) 取消

2、定义类信息

Dialog：对话框

生成的自定义类名为HelloDialog，是从基类QDialog派生而来。

←

^

Location

Kits

Details

汇总

类信息

指定您要创建的源码文件的基本类信息。

类名 (C): HelloDialog

基类 (B): QDialog

头文件 (H): hellodialog.h

源文件 (S): hellodialog.cpp

创建界面 (G): ☒

界面文件 (F): hellodialog.ui

3、项目的汇总信息：

←

^

Location

Kits

Details

汇总

项目管理

作为子项目添加到项目中: <None>

添加到版本控制系统 (V): <None> Configure...

要添加的文件

E:\3code\Qt\Qt_Exercise\01_helloworld\helloworld:

hellodialog.cpp

hellodialog.h

hellodialog.ui

helloworld.pro

main.cpp

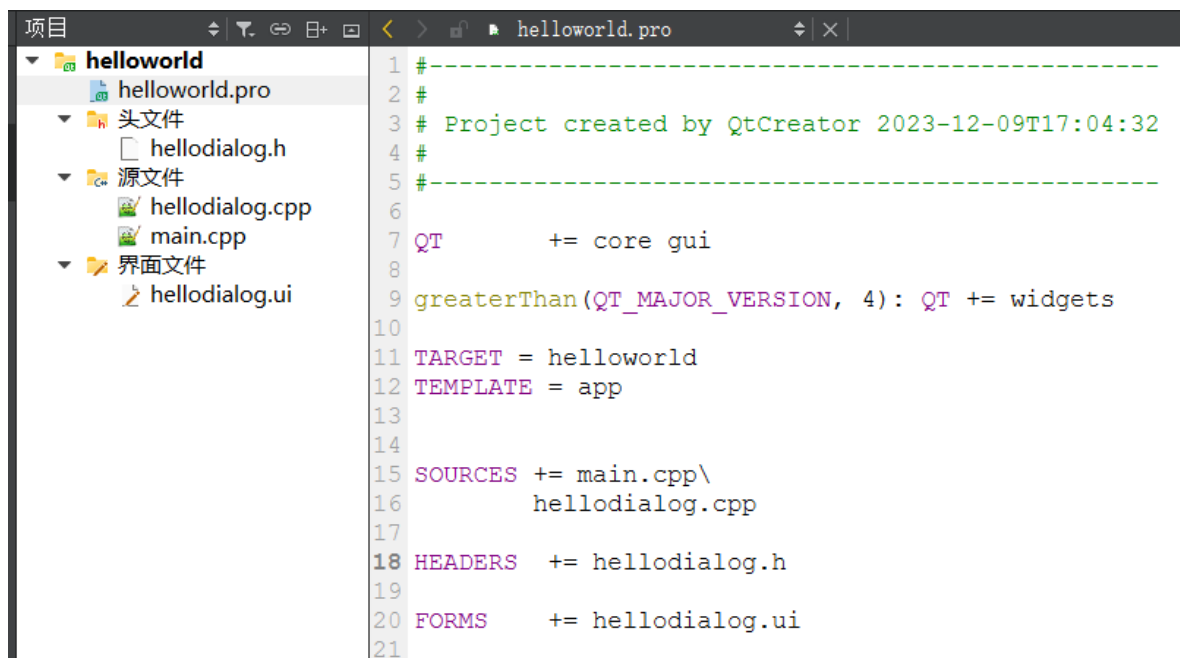
完成 (F) 取消

4、项目创建完成：

helloworld.pro：整个项目的工程文件，包含了项目的基本信息；

hellodialog.h、hellodialog.cpp、main.cpp：自定义类的一些C++文件；



hellodialog.ui：设计的界面文件



3、视频03：项目运行发布和设置图标

1、配置电脑Qt环境

1、问题：编译好的exe文件打开不了？

 hellodialog.o	2023-12-09 17:14	O 文件
 helloworld.exe	2023-12-09 17:14	应用程序
 main.o	2023-12-09 17:14	O 文件
 moc_hellodialog.cpp	2023-12-09 17:14	CPP 文件
 moc_hellodialog.o	2023-12-09 17:14	O 文件

2、原因：Qt的一些dll环境没有加入电脑环境

3、解决方法：

我的电脑→属性→高级系统设置→环境变量→系统变量→Path→编辑→新建→浏览→Qt安装位置里面的bin

设置好环境变量之后，便可以打开helloworld.exe文件

2、release版本与Debug版本

Debug版本：helloworld.exe占据1Mb

Release版本：helloworld.exe占据25kb

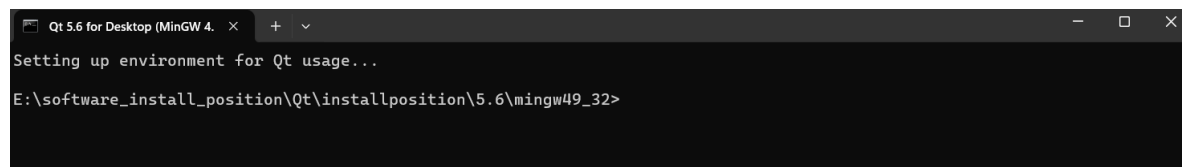
左下角选择版本为Release版本。



3、问题：文件打包给别人，却因为没有Qt后者没有加入相关环境而导致无法打开

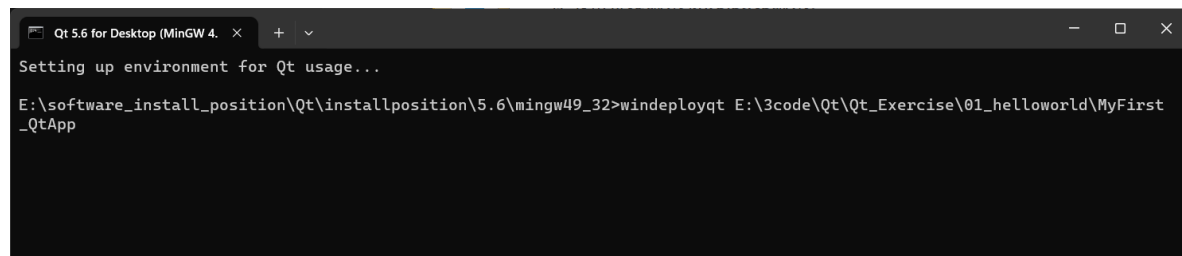
因为电脑的环境变量已设置（如第一点），直接使用windeployqt.exe

使用Qt 5.6 for Desktop 打开如下：



使用Qt的工具windeployqt给可执行文件helloworld.exe添加动态链接库（动态编译、动态链接的方式）格式：

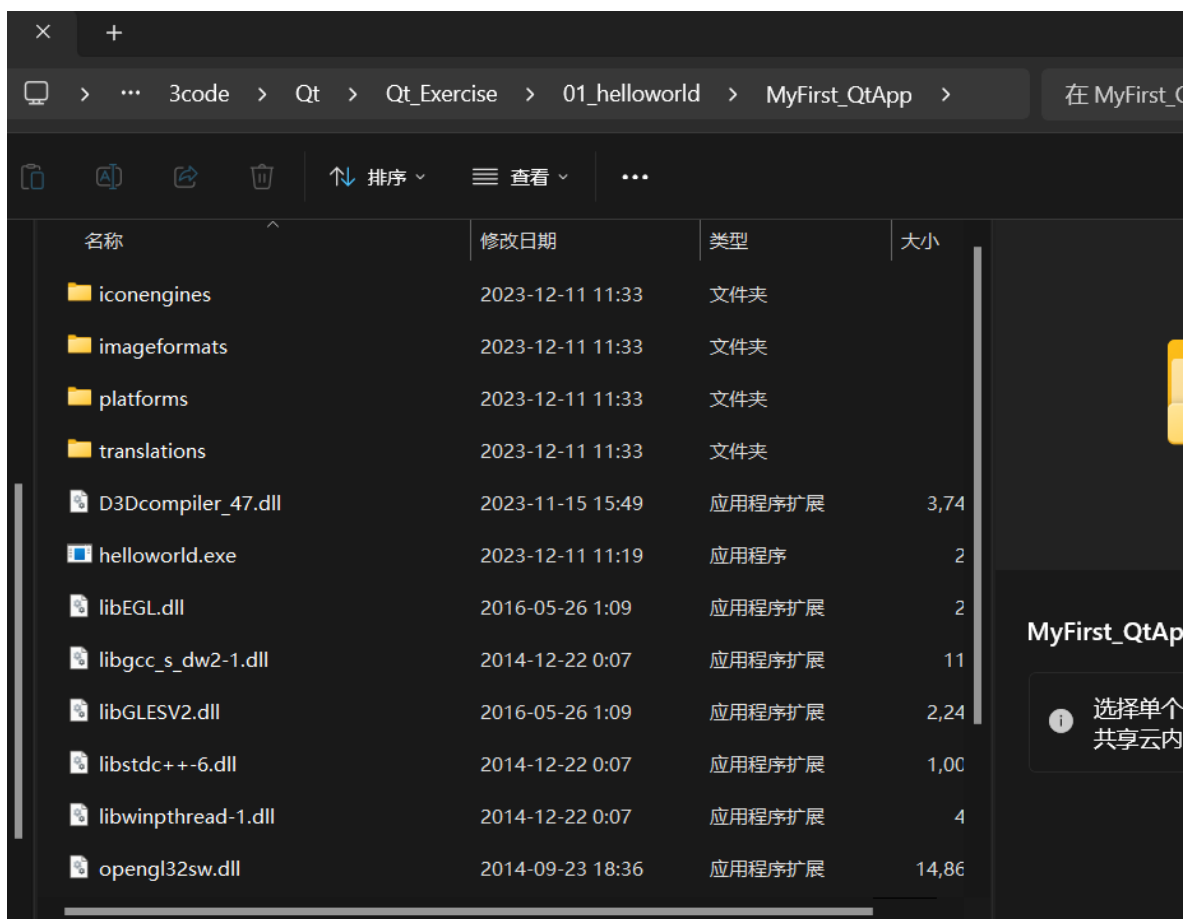
1 | windeployqt 可执行文件所在的文件路径



效果如下：

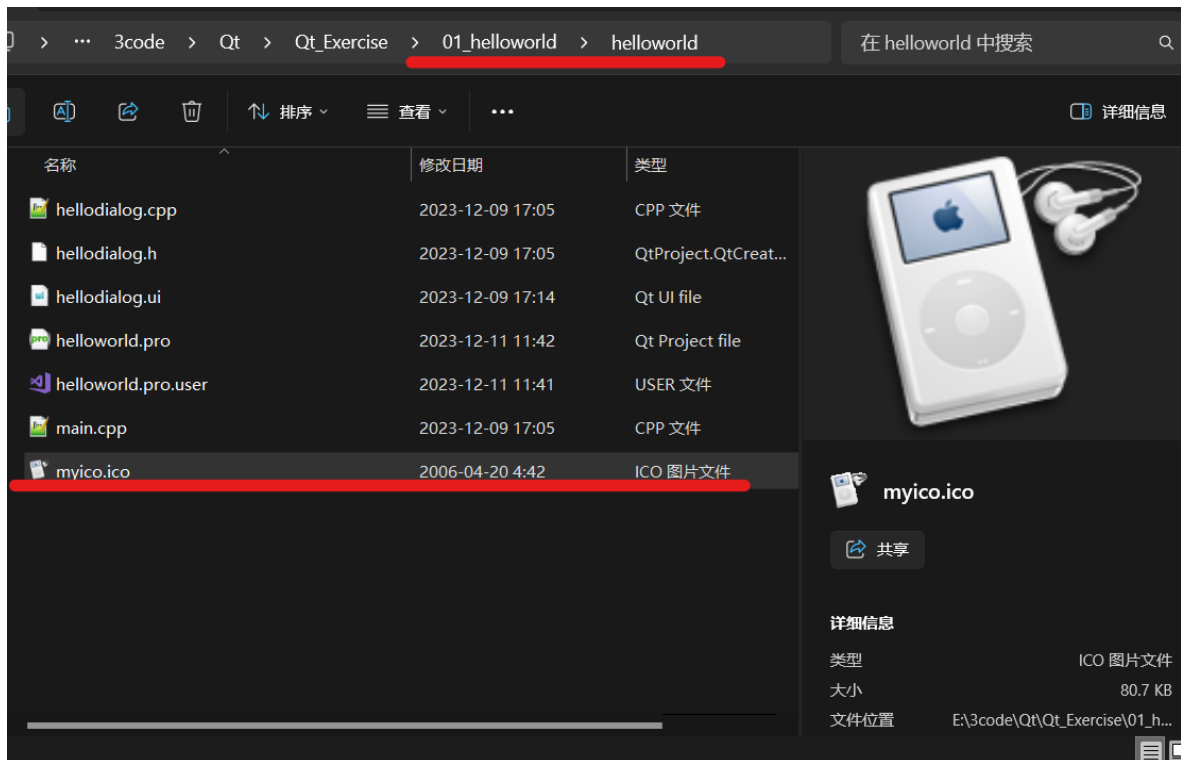
```
Qt 5.6 for Desktop (MinGW 4. x + v
E:\3code\Qt\Qt_Exercise\01_helloworld\MyFirst_QtApp\helloworld.exe 32 bit, release executable
Adding Qt5Svg for qsvgicon.dll
Direct dependencies: Qt5Core Qt5Widgets
All dependencies : Qt5Core Qt5Gui Qt5Widgets
To be deployed : Qt5Core Qt5Gui Qt5Svg Qt5Widgets
Updating Qt5Core.dll.
Updating Qt5Gui.dll.
Updating Qt5Svg.dll.
Updating Qt5Widgets.dll.
Updating libGLESV2.dll.
Updating libEGL.dll.
Updating D3Dcompiler_47.dll.
Updating opengl32sw.dll.
Updating libgcc_s_dw2-1.dll.
Updating libstdc++-6.dll.
Updating libwinpthread-1.dll.
Patching Qt5Core.dll...
Creating directory E:/3code/Qt/Qt_Exercise/01_helloworld/MyFirst_QtApp/iconengines.
Updating qsvgicon.dll.
Creating directory E:/3code/Qt/Qt_Exercise/01_helloworld/MyFirst_QtApp/imageformats.
Updating qdds.dll.
Updating qgif.dll.
Updating qicns.dll.
Updating qico.dll.
Updating qjpeg.dll.
Updating qsvg.dll.
Updating qtga.dll.
Updating qtiff.dll.
Updating qwbmp.dll.
Updating qwebp.dll.
```

效果如下：



4、添加自动图标

1、图表样式如下图所示（要将图标放入helloworld工程里面，而不是Debug/Release版本）：

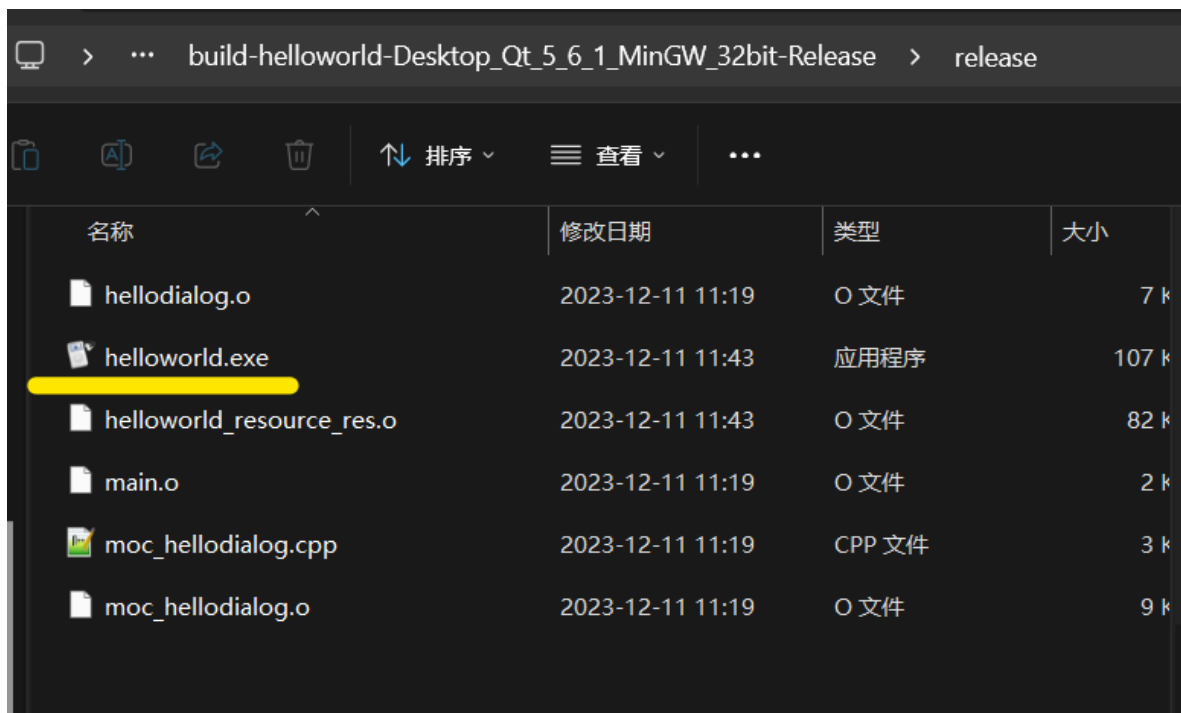
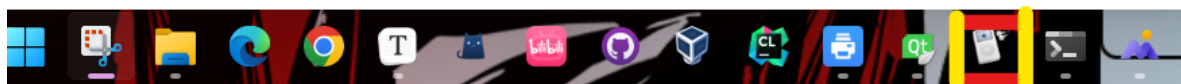
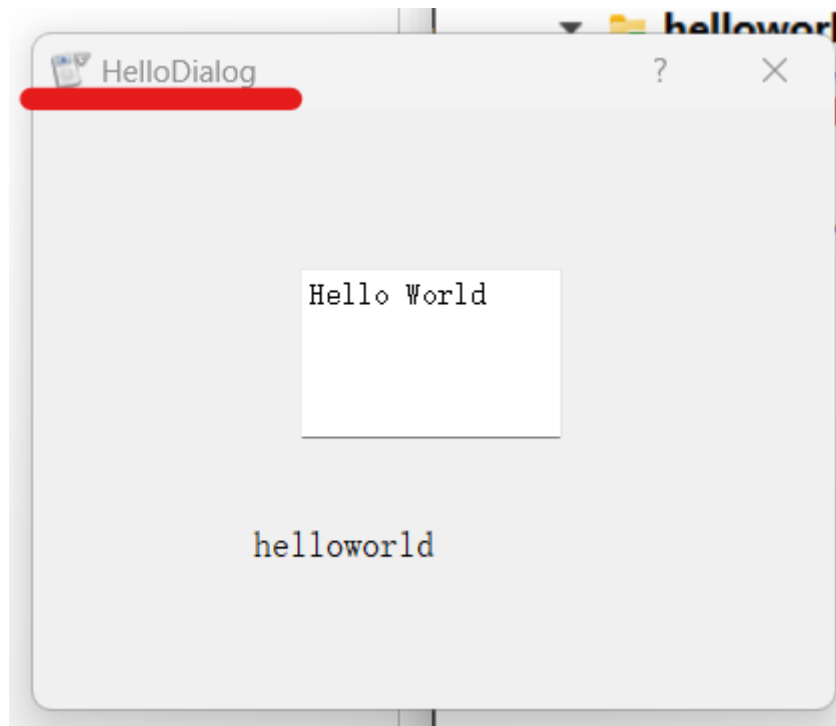


2、重新打开工程文件并在helloworld.pro文件里面添加如下所示代码：

```
1 | RC_ICONS = myico.ico
```



3、重新运行，发现图标都变样了：



4、视频04：项目文件组成及分析

1、QWidget

可视界面的基类

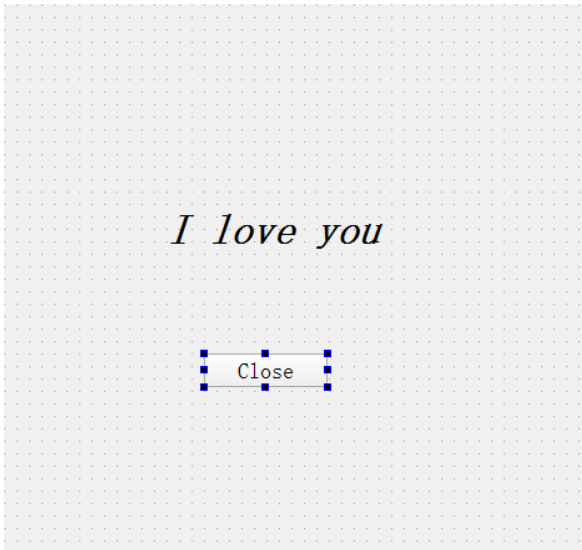
2、MyFirst_Widget.pro

文件代码内容：

```
1  QT      += core gui
2  // 核心图形用户界面，如果是基于控制台/命令的则不需要这行代码
3  // 如果有对话框，则自动添加
4
5  //QT      += sql //如果用到数据库，则将对模块添加
6
7  greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
8  // 条件执行语句，如果当前QT的版本大于4，当前项目加入 widgets 模块，QT4之前用的更多的是
   GUI，为了兼容之前版本
9
10 TARGET = MyFirst_widget
11 // 生成的可执行文件名称
12
13 TEMPLATE = app
14 // 生成出一般的应用程序
15
16 SOURCES += main.cpp\
17           widget.cpp
18           // 源文件
19
20 HEADERS  += widget.h
21 // 头文件
22
23 FORMS    += widget.ui
24 // ui文件
25
```

3、信号与槽

给对话框添加一个按钮，右下角改名为：Close，点击Close则退出当前对话框：widget



对象	类
Widget	QWidget
label	QLabel
pu...on	QPushButton

Filter

pushButton : QPushButton

属性	值
font	A [SimSun, 11]
字体族	Adobe 宋体 Std L
点大小	11
粗体	<input type="checkbox"/>
斜体	<input type="checkbox"/>
下划线	<input type="checkbox"/>
删除线	<input type="checkbox"/>
字距调整	<input checked="" type="checkbox"/>
反锯齿	首选默认
cursor	箭头
mouseTracking	<input type="checkbox"/>
focusPolicy	StrongFocus
contextMenuP...	DefaultContextMenu
acceptDrops	<input type="checkbox"/>
toolTip	
toolTipDuration	-1
statusTip	
whatsThis	
accessibleName	
accessibleDesc...	
layoutDirection	LeftToRight
autoFillBackgr...	<input type="checkbox"/>
styleSheet	
locale	Chinese, China
inputMethodH...	ImhNone

发送者	信号	接收者	槽
pushButton	clicked()	Widget	close()

Action Editor
Signals & Slots Editor

4、main文件解析

```

1  #include "widget.h"
2  #include <QApplication>
3
4  int main(int argc, char *argv[])
5  {
6      QApplication a(argc, argv);
7      // qt的标准应用程序类，任何Qt widget程序都要有一个 QApplication 对象
8
9      widget w;
```

```

10     // 创建一个窗体对象
11
12     w.show();
13     // 展示窗体
14
15     return a.exec();
16     // 启动应用程序的执行
17 }
18

```

5、Widget.h文件解析

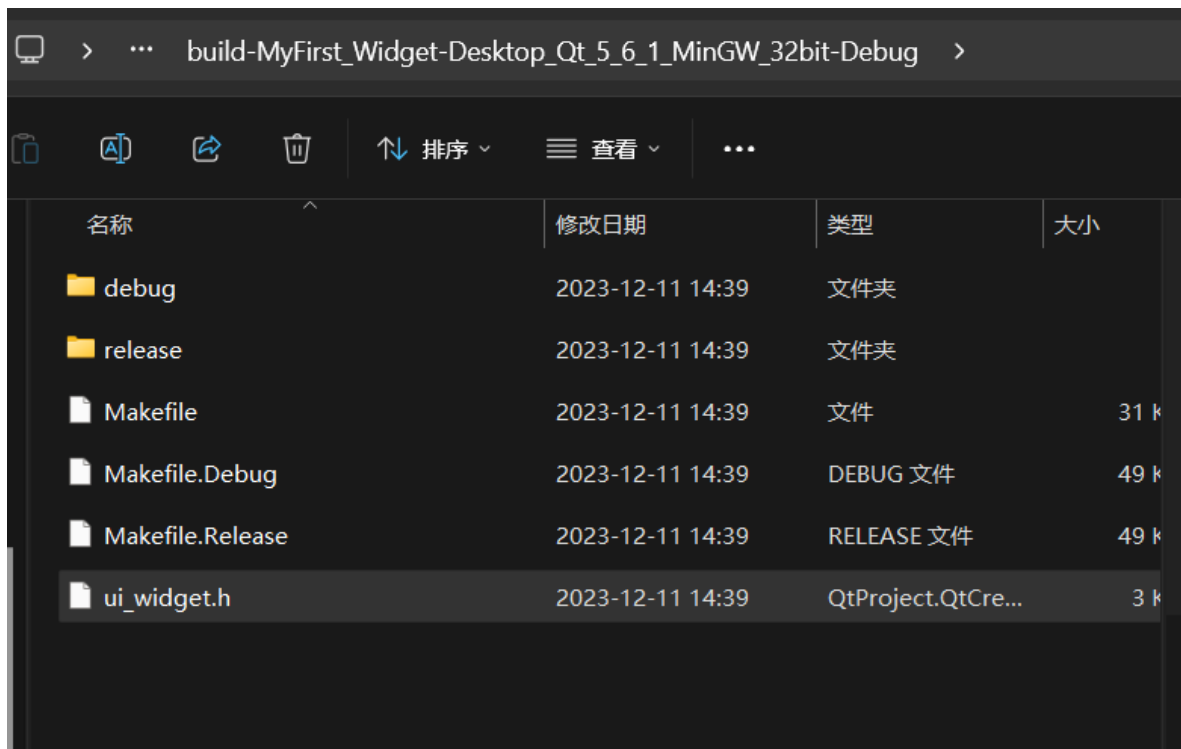
```

1  #ifndef WIDGET_H
2  #define WIDGET_H
3
4  #include <QWidget>
5
6  namespace Ui {
7  class widget;
8  }
9  // 定义了一个名称空间 Ui，在里面使用一个类
10
11  // -----
12  // 上述 widget 类与下方的 widget 并非同一个类
13  // -----
14
15  // -----此处 widget 为自定义类-----
16  // 来自于 QWidget 类
17  class widget : public QWidget
18  {
19      Q_OBJECT
20      // QT中使用了信号与槽都需要使用这个宏
21
22  public:
23      explicit widget(QWidget *parent = 0);
24      ~widget();
25
26  private:
27      Ui::widget *ui;
28      // 这里的 ui 指针是在 Ui 命名空间里面的 widget类的指针
29  };
30
31  #endif // WIDGET_H

```

解释Ui里面 Widget 类的由来

1、下图所示 ui_widget.h 并没有出现在工程目录当中，只出现在文件夹里面



2、打开ui_widget.h

```

class Ui_Widget
{
public:
    QLabel *label;
    QPushButton *pushButton;

    void setupUi(QWidget *Widget)
    {
        if (Widget->objectName().isEmpty())
            Widget->setObjectName(QStringLiteral("Widget"));
        Widget->resize(463, 438);
        label = new QLabel(Widget);
        label->setObjectName(QStringLiteral("label"));
        label->setGeometry(QRect(130, 140, 181, 81));
        QFont font;
        font.setPointSize(20);
        font.setBold(true);
        font.setItalic(true);
        font.setWeight(75);
        label->setFont(font);
        pushButton = new QPushButton(Widget);
        pushButton->setObjectName(QStringLiteral("pushButton"));
        pushButton->setGeometry(QRect(160, 280, 99, 27));

        retranslateUi(Widget);
        QObject::connect(pushButton, SIGNAL(clicked()), Widget, SLOT(close()));

        QMetaObject::connectSlotsByName(Widget);
    } // setupUi

    void retranslateUi(QWidget *Widget)
    {
        Widget->setWindowTitle(QApplication::translate("Widget", "Widget", 0));
        label->setText(QApplication::translate("Widget", "I love you", 0));
        pushButton->setText(QApplication::translate("Widget", "Close", 0));
    } // retranslateUi
};

namespace Ui {
    class Widget: public Ui_Widget {};
} // namespace Ui

QT_END_NAMESPACE

```

3、Ui里面Widget的来源如上图所示

4、设置的按钮操作也会显示在这里：

```

        QObject::connect(pushButton, SIGNAL(clicked()), Widget, SLOT(close()));

        QMetaObject::connectSlotsByName(Widget);
    } // setupUi

```

6、widget.cpp文件解析

1、代码

```

1  #include "widget.h"
2  #include "ui_widget.h"
3
4  Widget::Widget(QWidget *parent) : QWidget(parent), ui(new Ui::Widget)
5      // 初始化列表，派生类的初始化要先初始化基类QWidget
6  {
7      ui->setupUi(this);
8      // setupUi用来设置窗口各种控件的属性，（在ui_widget.h文件里面）

```

```
9   }
10
11   widget::~~widget()
12   {
13       delete ui;
14       // 构造函数出现了 new
15   }
```

ui创建了一个label和PushButton，都会出现在Ui_Widget类里面

```
QT_BEGIN_NAMESPACE
class Ui_Widget
{
public:
    QLabel *label;
    QPushButton *pushButton;

    void setupUi(QWidget *Widget)
    {
        if (Widget->objectName().isEmpty())
```

7、Widget.ui文件

编译Ui界面时，会自动生成出相应代码