

Qt –MainWindow

Yih-Chuan Lin

CSIE Windows Programming Class

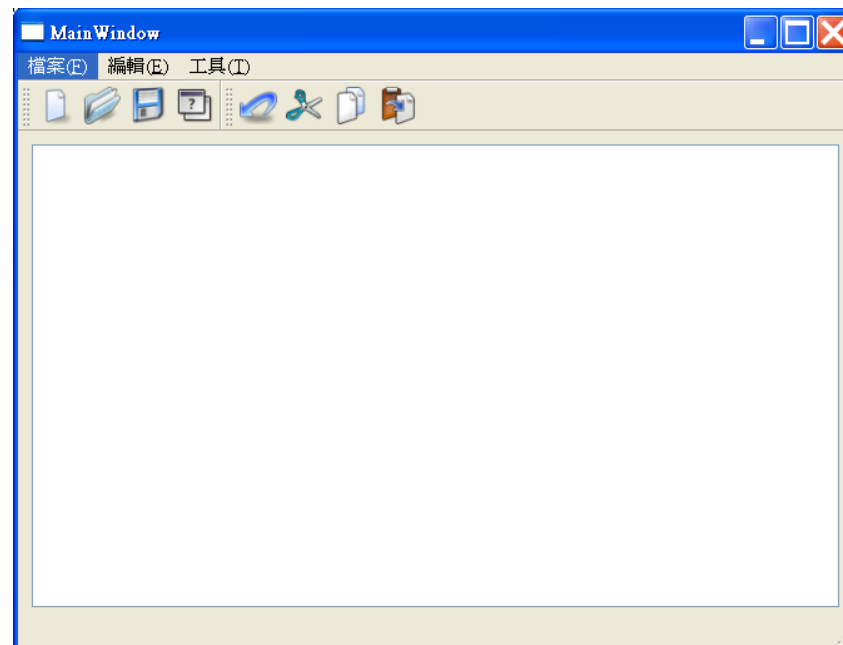
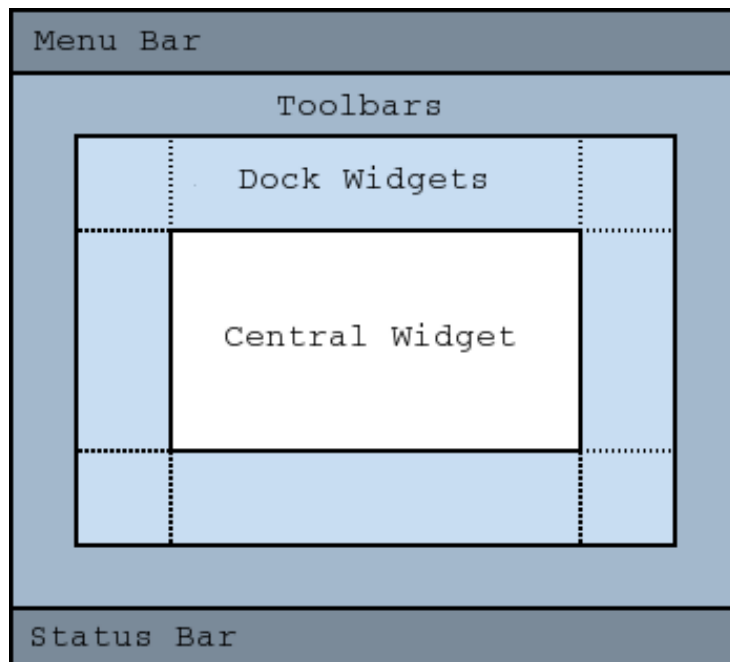
National Formosa University

Qt- MainWindow

- 學習目的
 - 認識Qt MainWindow之主視窗框架
 - 練習使用Qt Designer工具製作主視窗人機介面
 - 學習如何整合Qt Designer視窗人機介面於應用程式中

Qt- MainWindow

Qt 主視窗之框架: 提供建立應用程式人機界面的架構

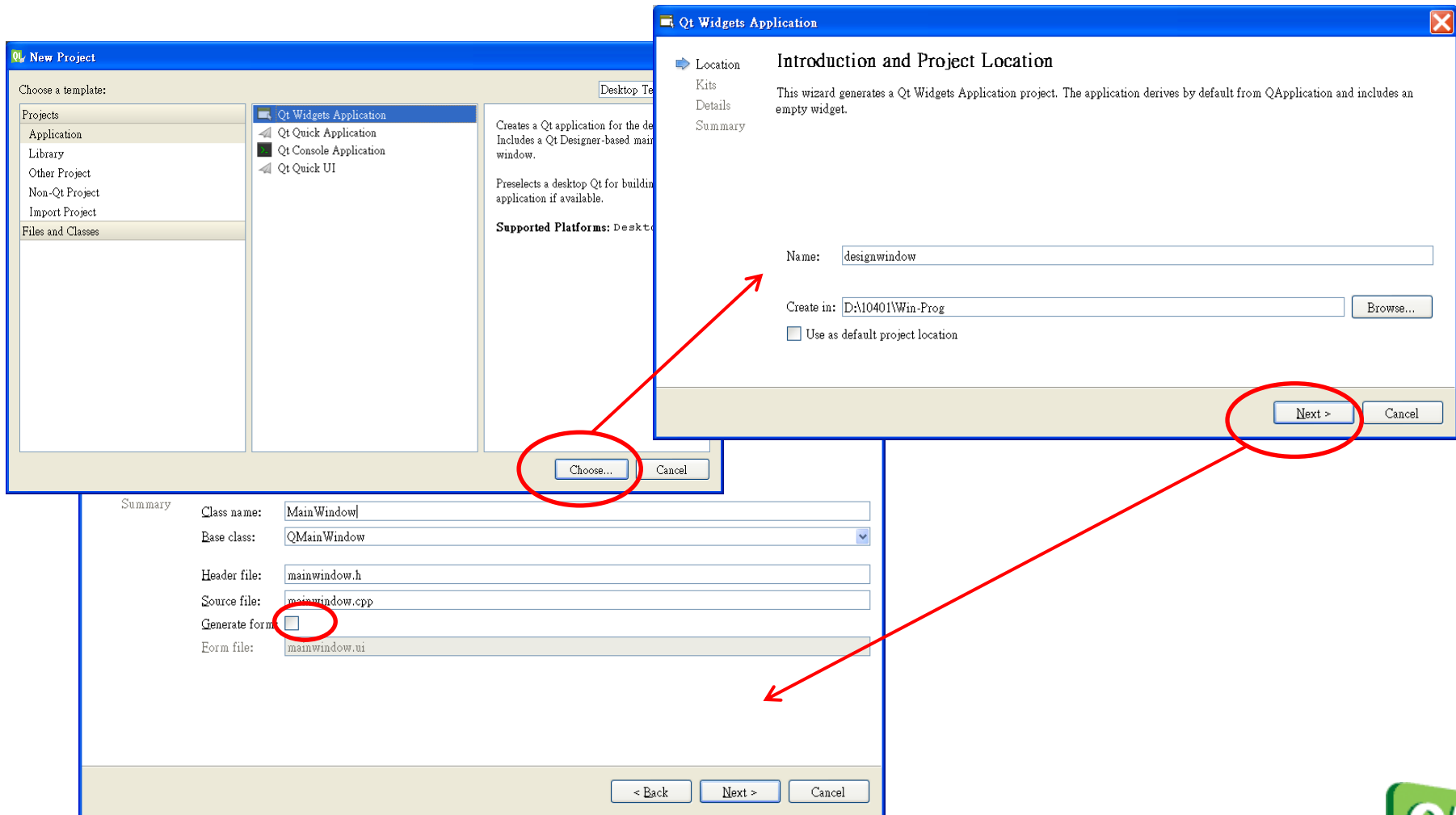


相關類別：

QMainWindow, QMenuBar, QToolBar, QDockWidget, QStatusBar.

Qt- MainWindow

- Create a mainwindow application:

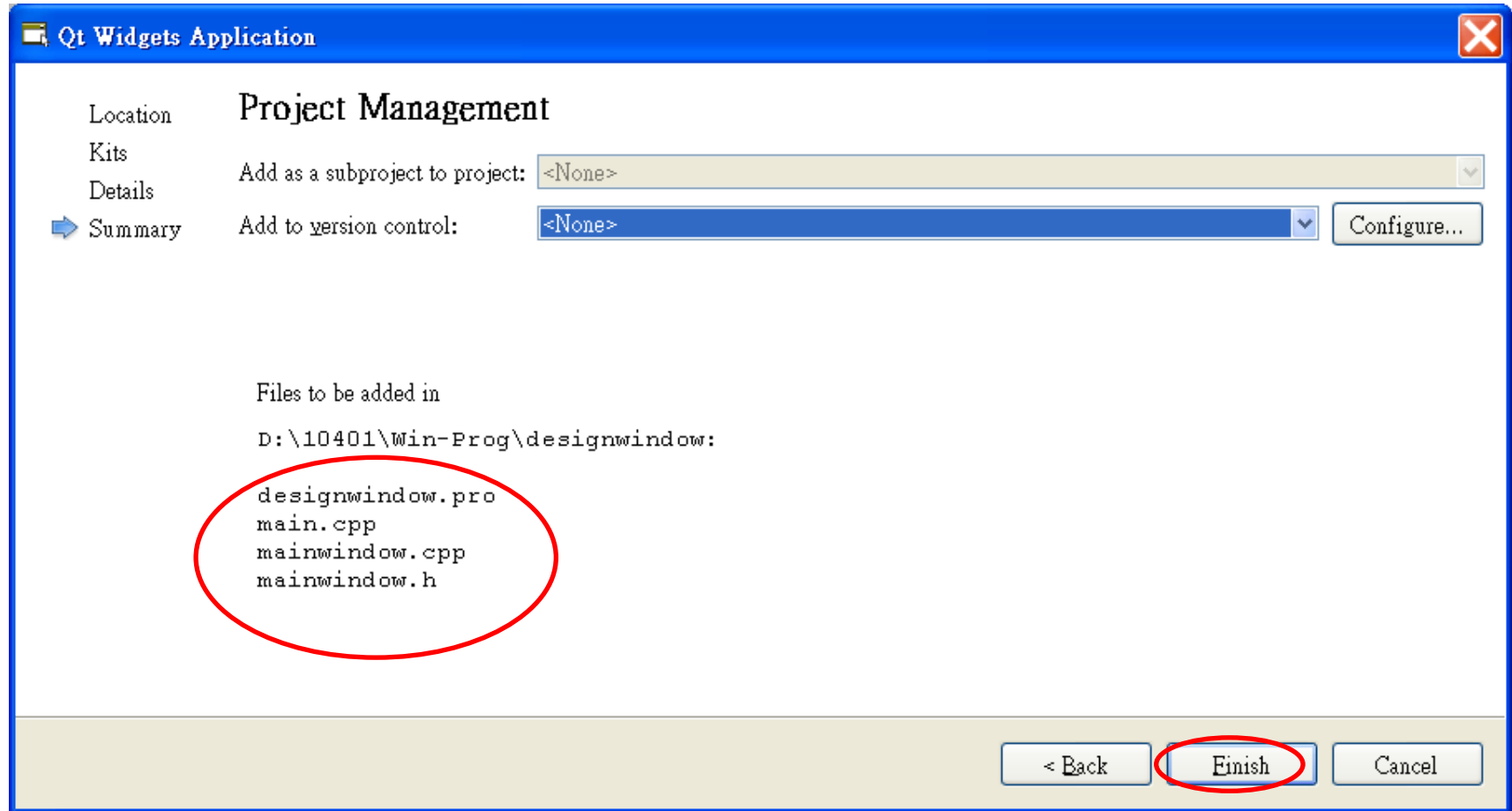




Code less.
Create more.
Deploy everywhere.

Qt- MainWindow

- Create a dialog-based application:



Qt- MainWindow

- Create a mainwindow application:

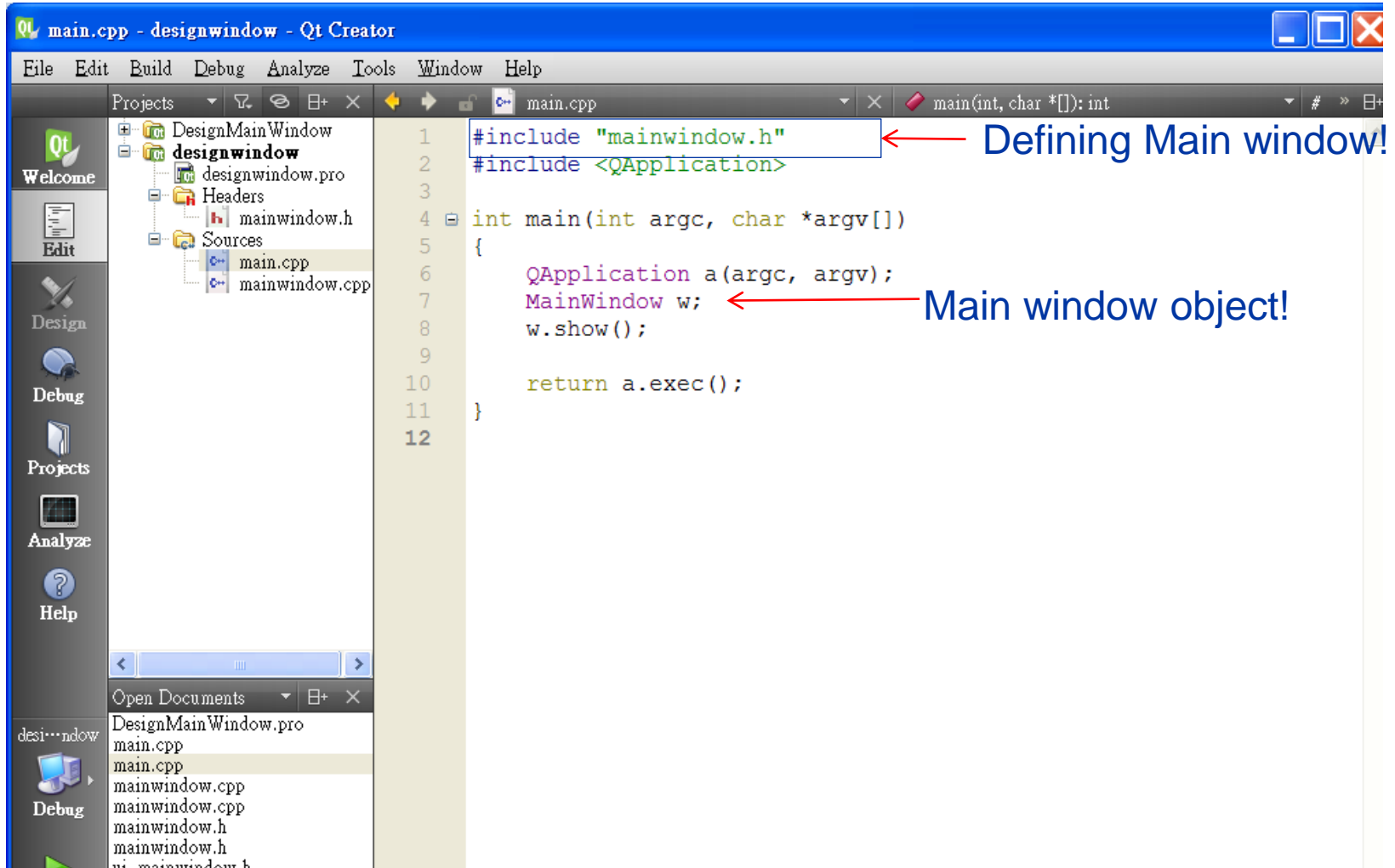
The screenshot illustrates the steps to create a Qt application:

- Qt Widgets Application Wizard:** The 'Project Management' tab is active. The 'Files to be added in' section lists:
 - D:\10401\Win-Prog\designwindow:
 - designwindow.pro
 - main.cpp
 - mainwindow.cpp
 - mainwindow.h
 The 'Finish' button is highlighted with a red circle.
- Project Files:** A tree view shows the project structure:
 - designwindow
 - designwindow.pro
 - Headers
 - mainwindow.h
 - Sources
 - main.cpp
 - mainwindow.cpp
- Code Editor:** The mainwindow.cpp file is open, showing the following code:


```
1 #include <mainwindow.h>
2 #include <QApplication>
3
4 int main(int argc, char *argv[])
5 {
6     QApplication a(argc, argv);
7     MainWindow w;
8     w.show();
9
10    return a.exec();
11 }
12
```
- Build Settings:** The 'Build Settings' dialog is open. The 'Debug' configuration is selected in the dropdown. The 'Shadow build' checkbox is unchecked. The 'Build directory' is set to D:\10401\Win-Prog\designwindow.

Qt- MainWindow

- Check out the main function:



The screenshot shows the Qt Creator IDE with the `main.cpp` file open. The left sidebar displays the project structure for `DesignMainWindow`, including `designwindow.pro`, `Headers` (containing `mainwindow.h`), and `Sources` (containing `main.cpp` and `mainwindow.cpp`). The main editor area shows the following C++ code:

```

1  #include "mainwindow.h"
2  #include <QApplication>
3
4  int main(int argc, char *argv[])
5  {
6      QApplication a(argc, argv);
7      MainWindow w;
8      w.show();
9
10     return a.exec();
11 }
12

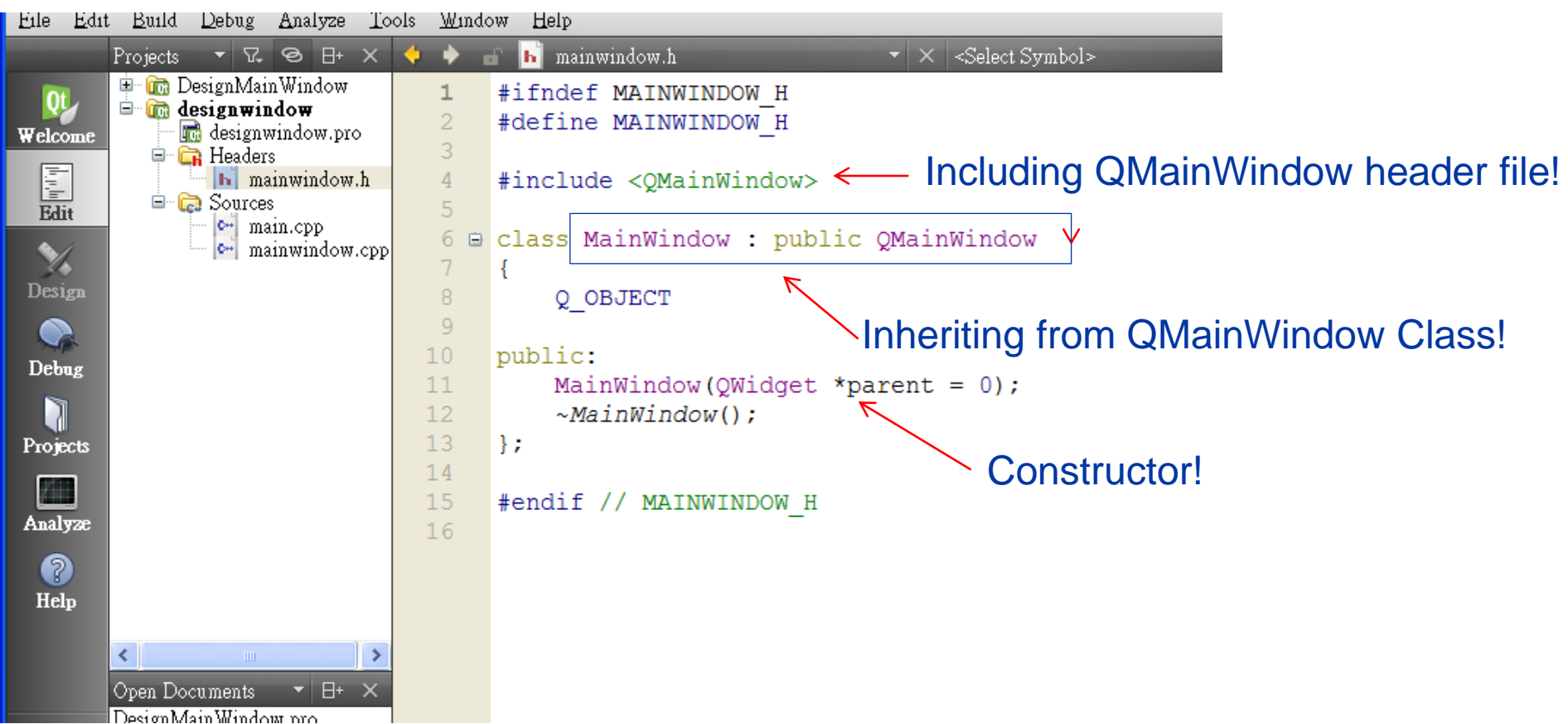
```

Two red arrows point to specific lines in the code with blue text annotations:

- An arrow points from the text **Defining Main window!** to line 1, `#include "mainwindow.h"`.
- An arrow points from the text **Main window object!** to line 7, `MainWindow w;`.

Qt- MainWindow

- Check out the definition of MainWindow class:



```
1  #ifndef MAINWINDOW_H
2  #define MAINWINDOW_H
3
4  #include <QMainWindow>
5
6  class MainWindow : public QMainWindow
7  {
8      Q_OBJECT
9
10 public:
11     MainWindow(QWidget *parent = 0);
12     ~MainWindow();
13 };
14
15 #endif // MAINWINDOW_H
16
```

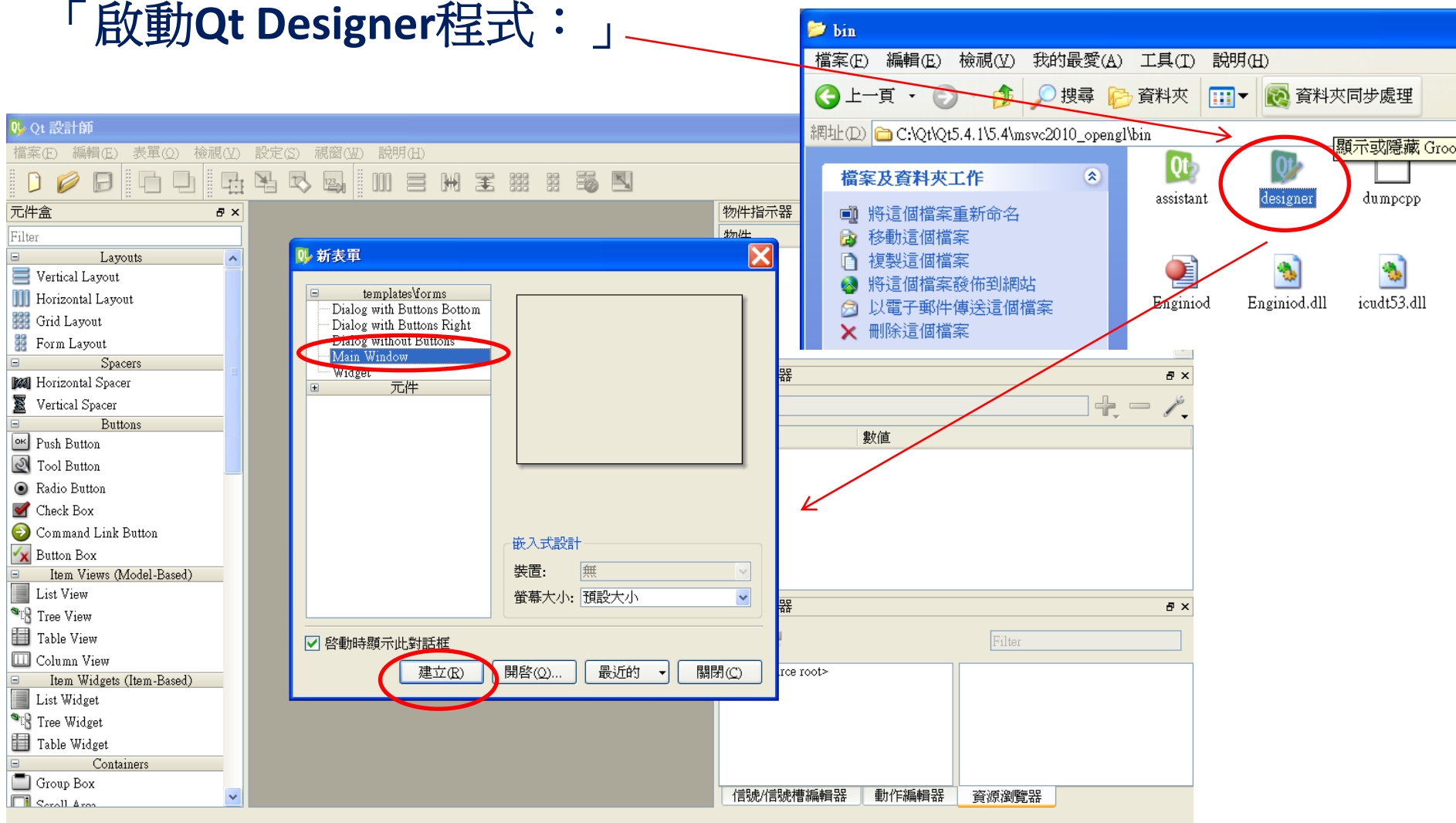
Including QMainWindow header file!

Inheriting from QMainWindow Class!

Constructor!

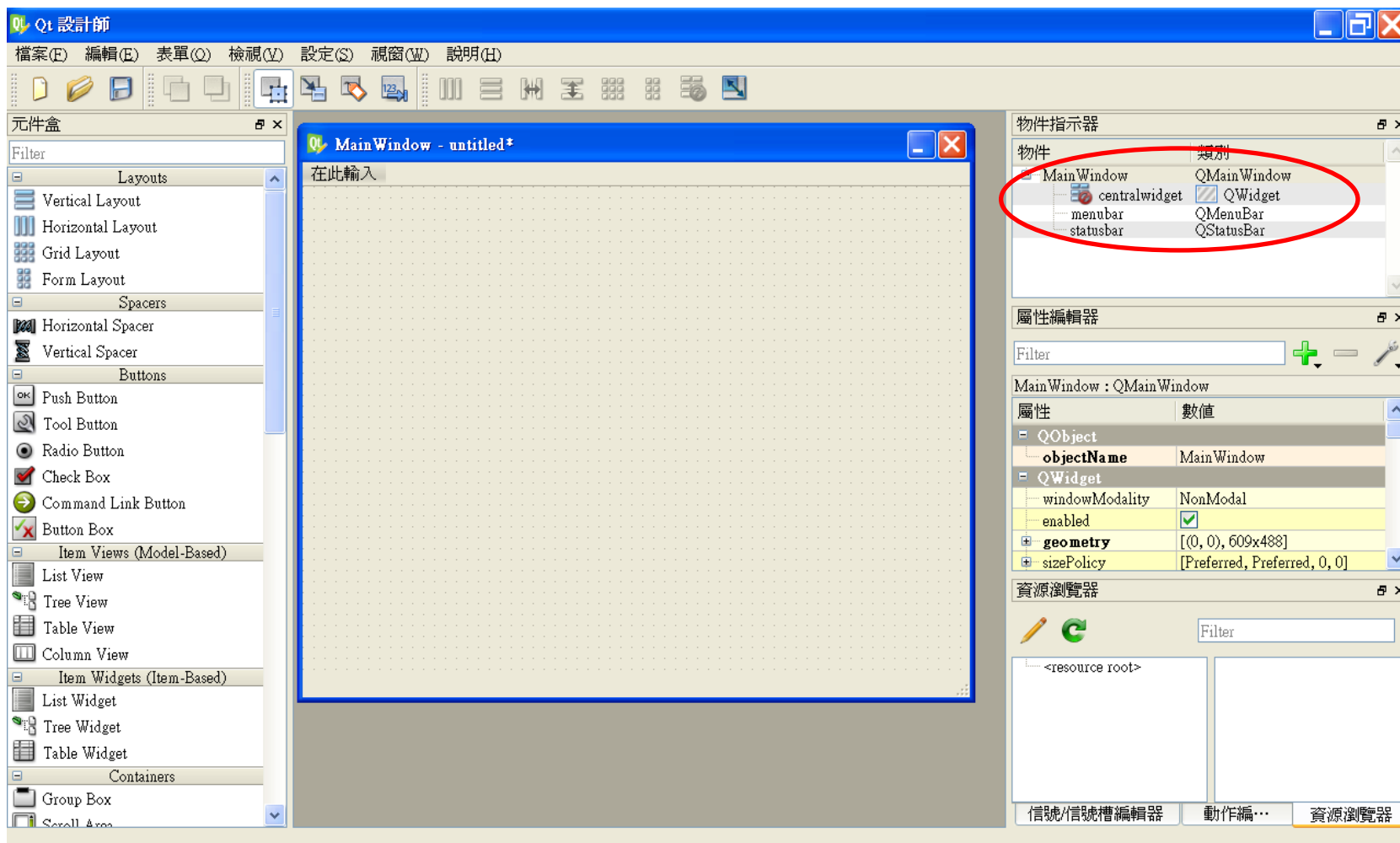
Qt- MainWindow

「啟動Qt Designer程式：」



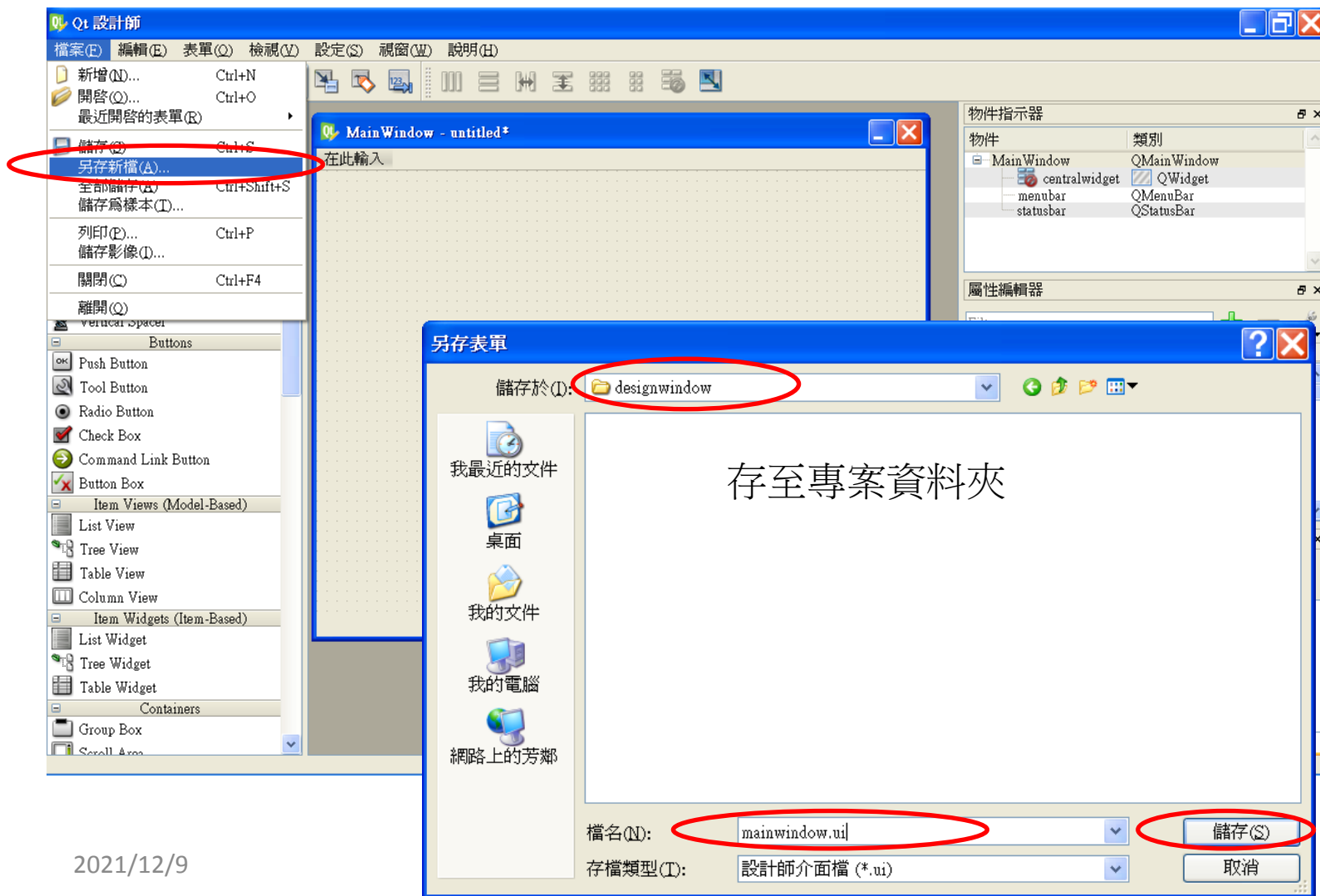
Qt- MainWindow

主視窗初始人機界面



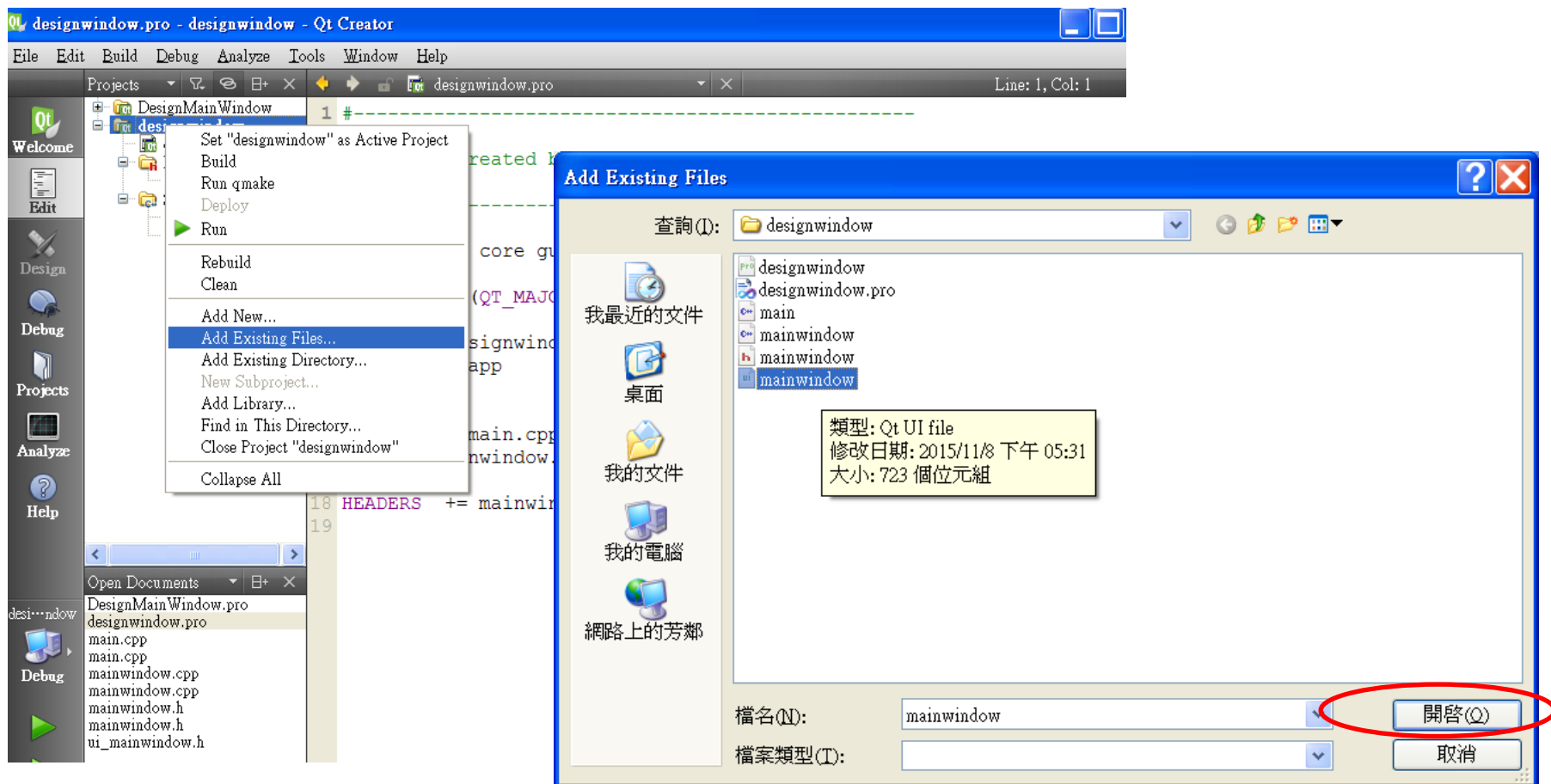
Qt- MainWindow

另存人機界面檔(mainwindow.ui)



Qt- MainWindow

新增人機界面檔(mainwindow.ui)至專案

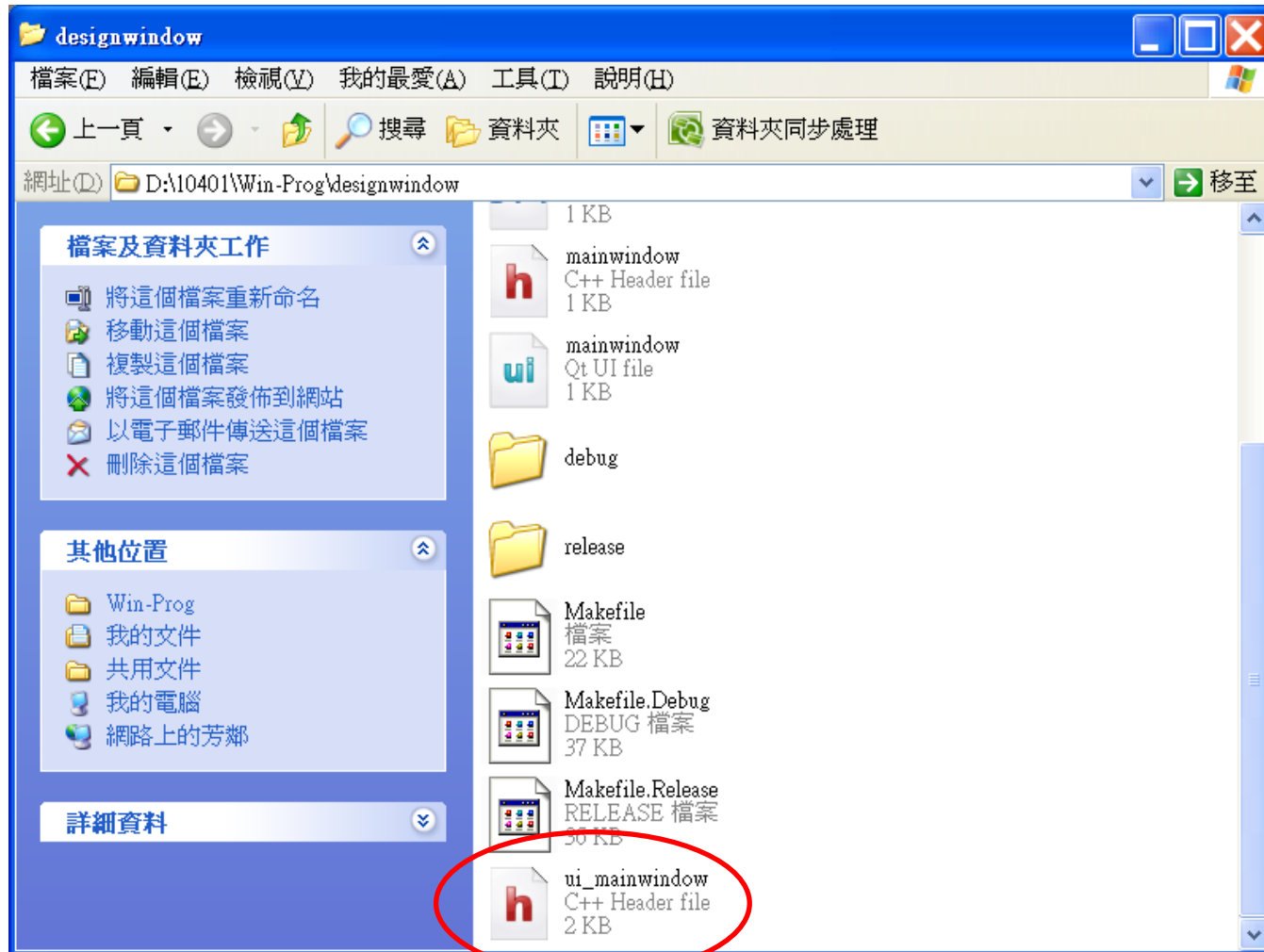




Code less.
Create more.
Deploy everywhere.

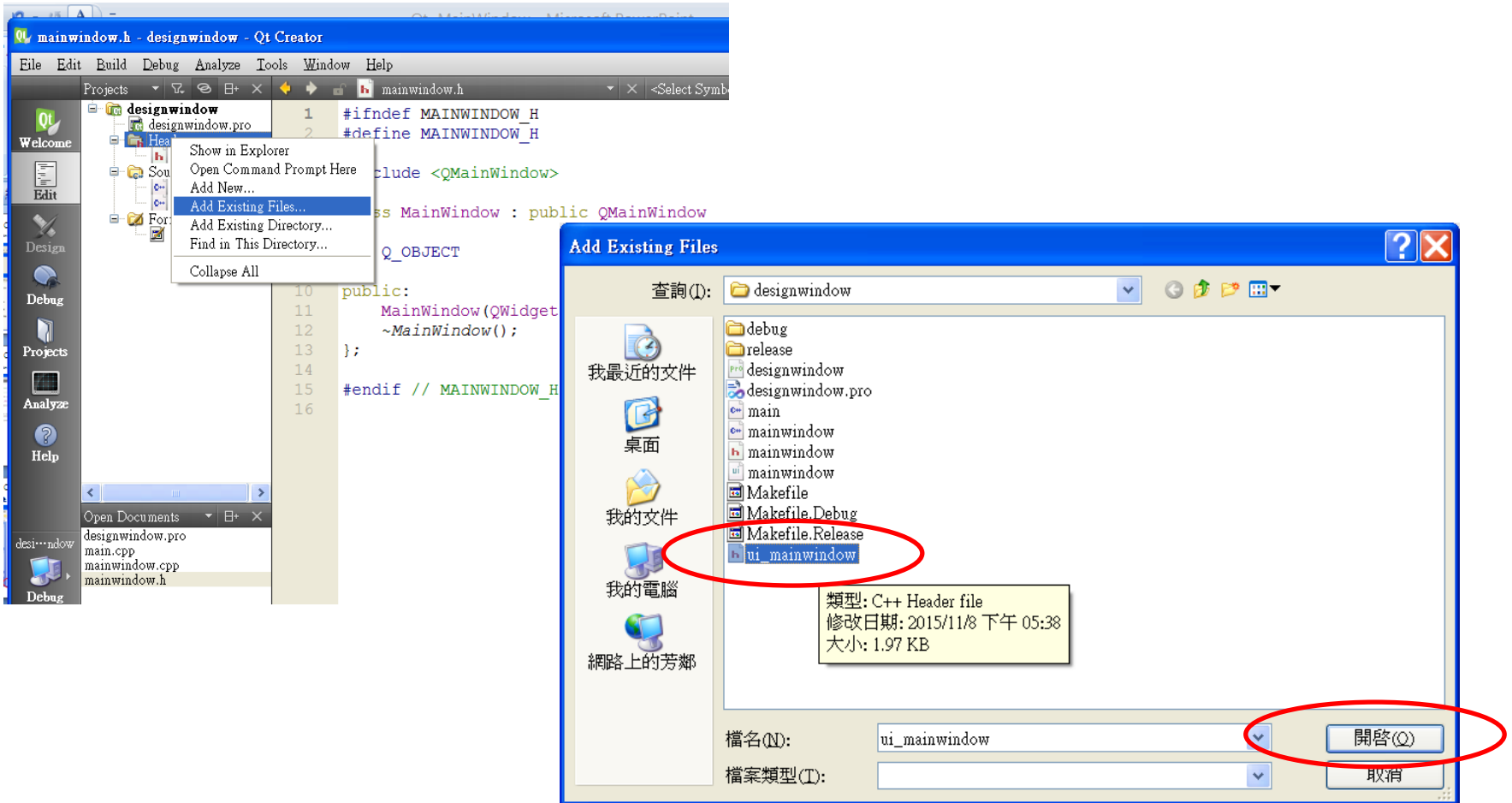
Qt- MainWindow

建置專案(Build All)後、查找 ui_mainwindow.h 標頭檔



Qt- MainWindow

引入ui_mainwindow.h標頭檔



Qt- MainWindow

檢視人機介面定義類別： Ui_MainWindow

The screenshot shows the Qt Creator IDE with the file `ui_mainwindow.h` open. The left sidebar displays the project structure for `designwindow`, including `designwindow.pro`, `Headers` (containing `mainwindow.h` and `ui_mainwindow.h`), `Sources` (containing `main.cpp` and `mainwindow.cpp`), and `Forms` (containing `mainwindow.ui`). The main editor area shows the C++ header file `ui_mainwindow.h` with the following content:

```

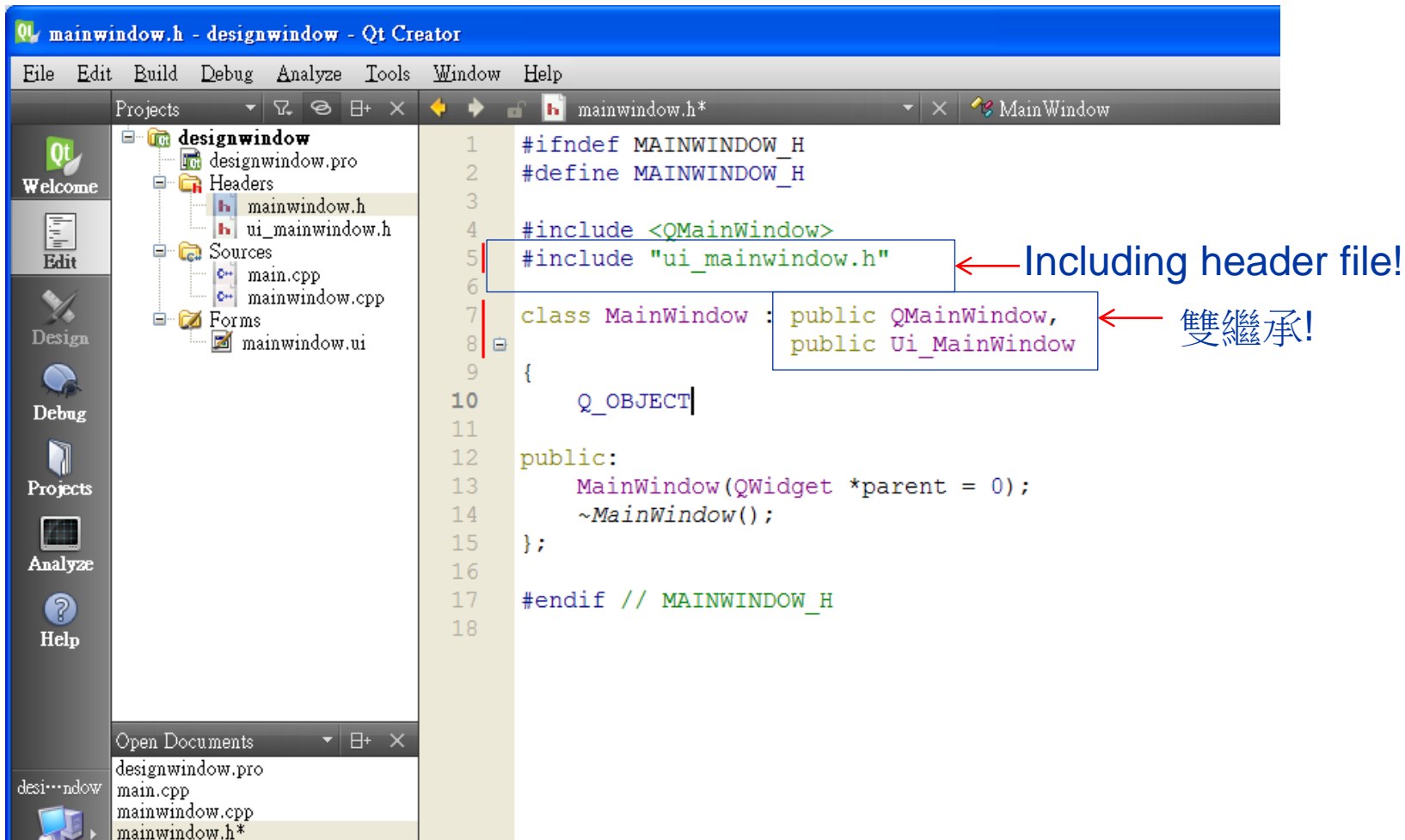
1  /*****
8
9  #ifndef UI_MAINWINDOW_H
10 #define UI_MAINWINDOW_H
11
12 #include <QtCore/QVariant>
13 #include <QtWidgets/QAction>
14 #include <QtWidgets/QApplication>
15 #include <QtWidgets/QButtonGroup>
16 #include <QtWidgets/QHeaderView>
17 #include <QtWidgets/QMainWindow>
18 #include <QtWidgets/QMenuBar>
19 #include <QtWidgets/QStatusBar>
20 #include <QtWidgets/QWidget>
21
22 QT_BEGIN_NAMESPACE
23
24 class Ui_MainWindow
25 {
26 public:
27     QWidget *centralwidget;
28     QMenuBar *menubar;
29     QStatusBar *statusbar;
30
31     void setupUi(QMainWindow *MainWindow)
32     {
33         if (MainWindow->objectName().isEmpty())
34             MainWindow->setObjectName(QStringLiteral("MainWi
35         MainWindow->resize(609, 488);

```

The line `class Ui_MainWindow` is circled in red in the original image.

Qt- MainWindow

變更mainwindow類別的定義：多重繼承 (加入人機界面)



The screenshot shows the Qt Creator IDE with the `mainwindow.h` file open. The code defines the `MainWindow` class, which inherits from `QMainWindow` and `Ui_MainWindow`. Annotations highlight the inclusion of the `ui_mainwindow.h` header file and the multiple inheritance setup.

```

1  #ifndef MAINWINDOW_H
2  #define MAINWINDOW_H
3
4  #include <QMainWindow>
5  #include "ui_mainwindow.h"
6
7  class MainWindow : public QMainWindow,
8                   public Ui_MainWindow
9  {
10     Q_OBJECT
11
12 public:
13     MainWindow(QWidget *parent = 0);
14     ~MainWindow();
15 };
16
17 #endif // MAINWINDOW_H
18

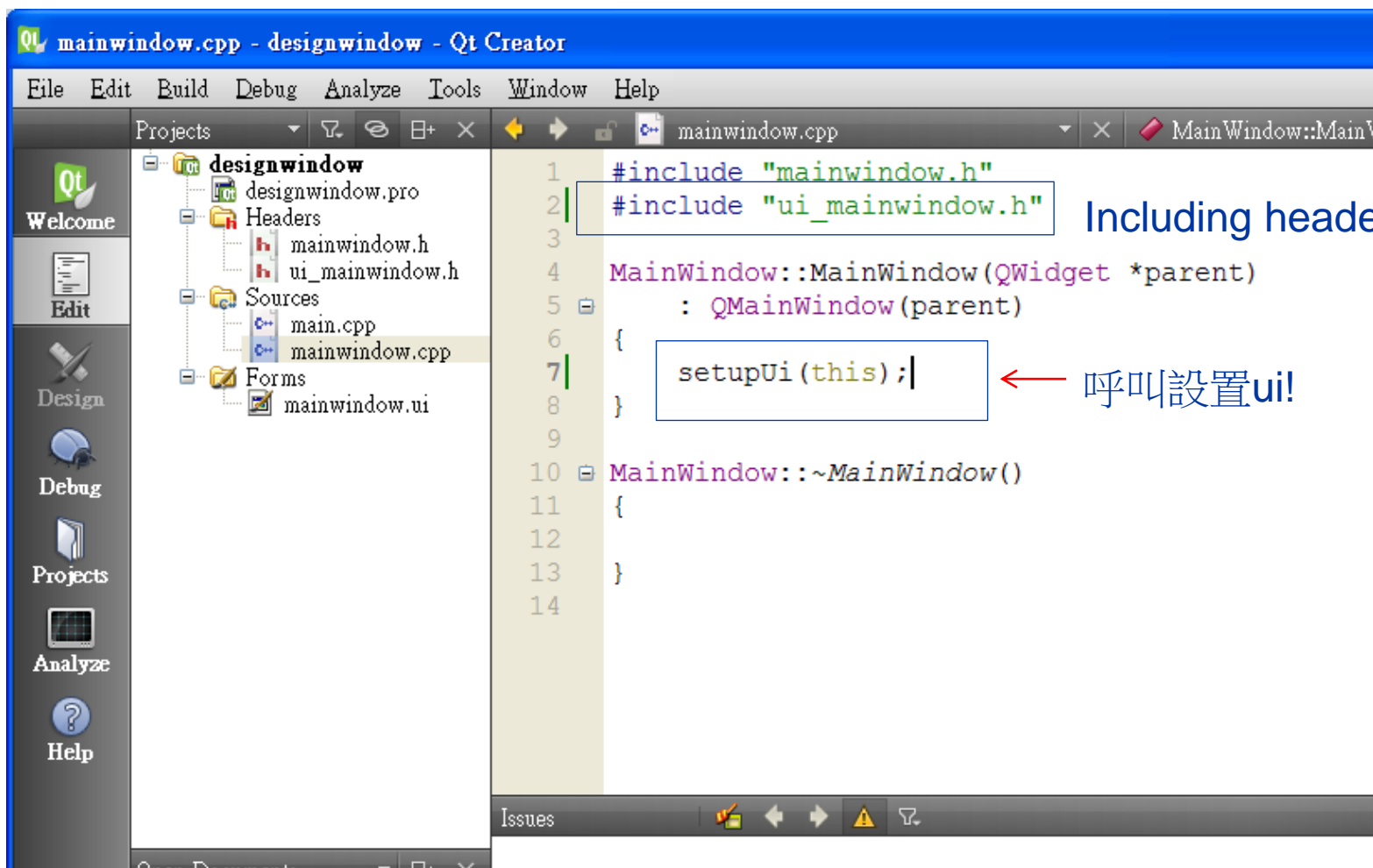
```

Annotations in the image:

- Line 5: `#include "ui_mainwindow.h"` is annotated with "Including header file!"
- Lines 7-8: `class MainWindow : public QMainWindow, public Ui_MainWindow` is annotated with "雙繼承!" (Multiple Inheritance!).

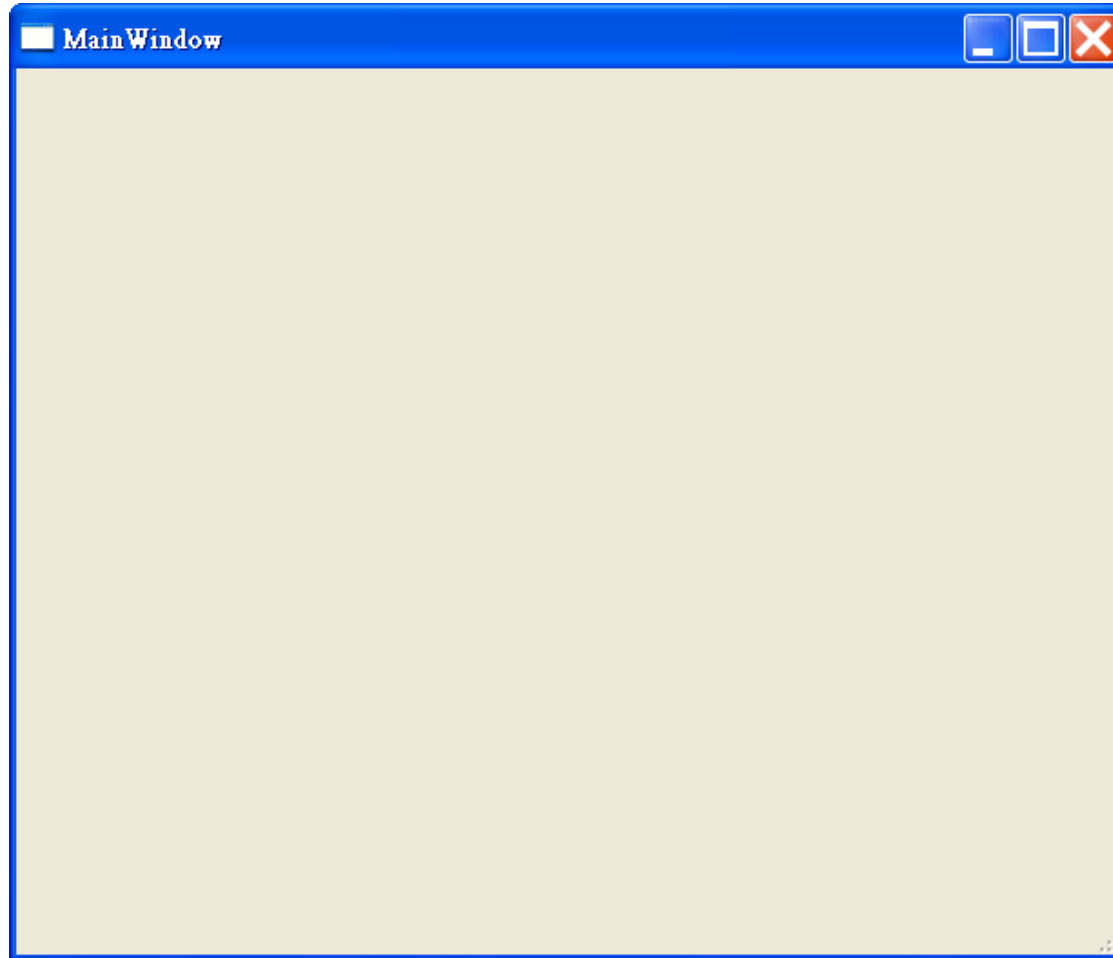
Qt- MainWindow

變更mainwindow類別的建構函數：設置人機界面



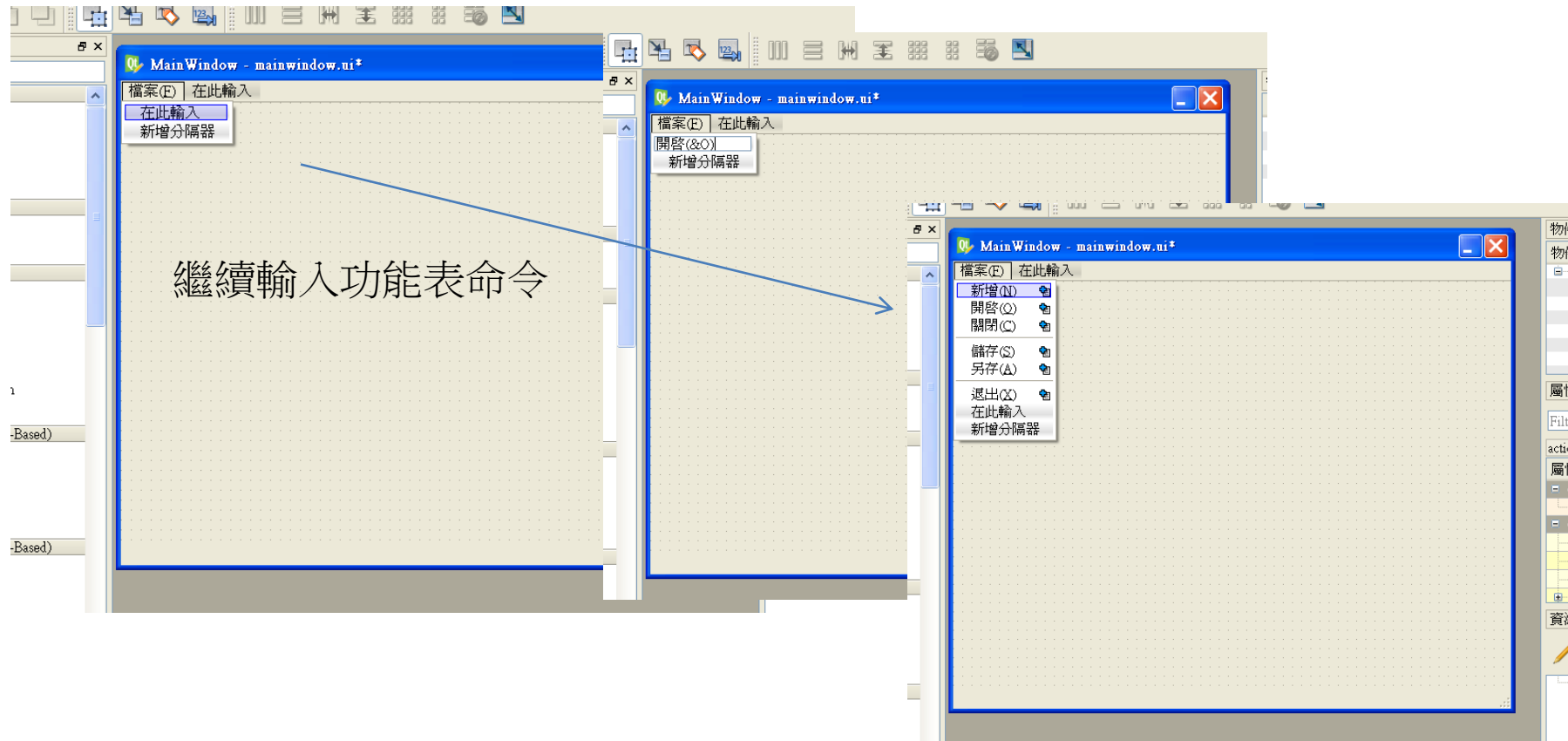
Qt- MainWindow

建置、執行專案



Qt- MainWindow

Qt Designer: double-clicking [在此輸入]，輸入第一個功能表名字



Qt- MainWindow

Qt Designer: 修改功能表命令物件名(動作物件：QAction objects)

雙擊

持續修改

動作編輯器

名稱	已使用	文字	捷徑	可勾選
action_O	✓	新增(&N)		
action_O_2	✓	開啟(&O)		
action_C	✓	關閉(&C)		
action_S	✓	儲存(&S)		
action_A	✓	另存(&A)		
action_X	✓	退出(&X)		

編輯動作

文字(I): 新增(&N)

物件名稱(N): actionNew

工具提示(o): 新增(N)

主題圖示(a):

圖示(I): 正常時關閉

可勾選的(C): ☐

捷徑(S): Ctrl+Z

OK Cancel

屬性編輯器

Filter

action_O : QAction

屬性 數值

QObject

objectName action_O

QAction

checkable ☐

checked ☐

enabled ☒

icon

資源瀏覽器

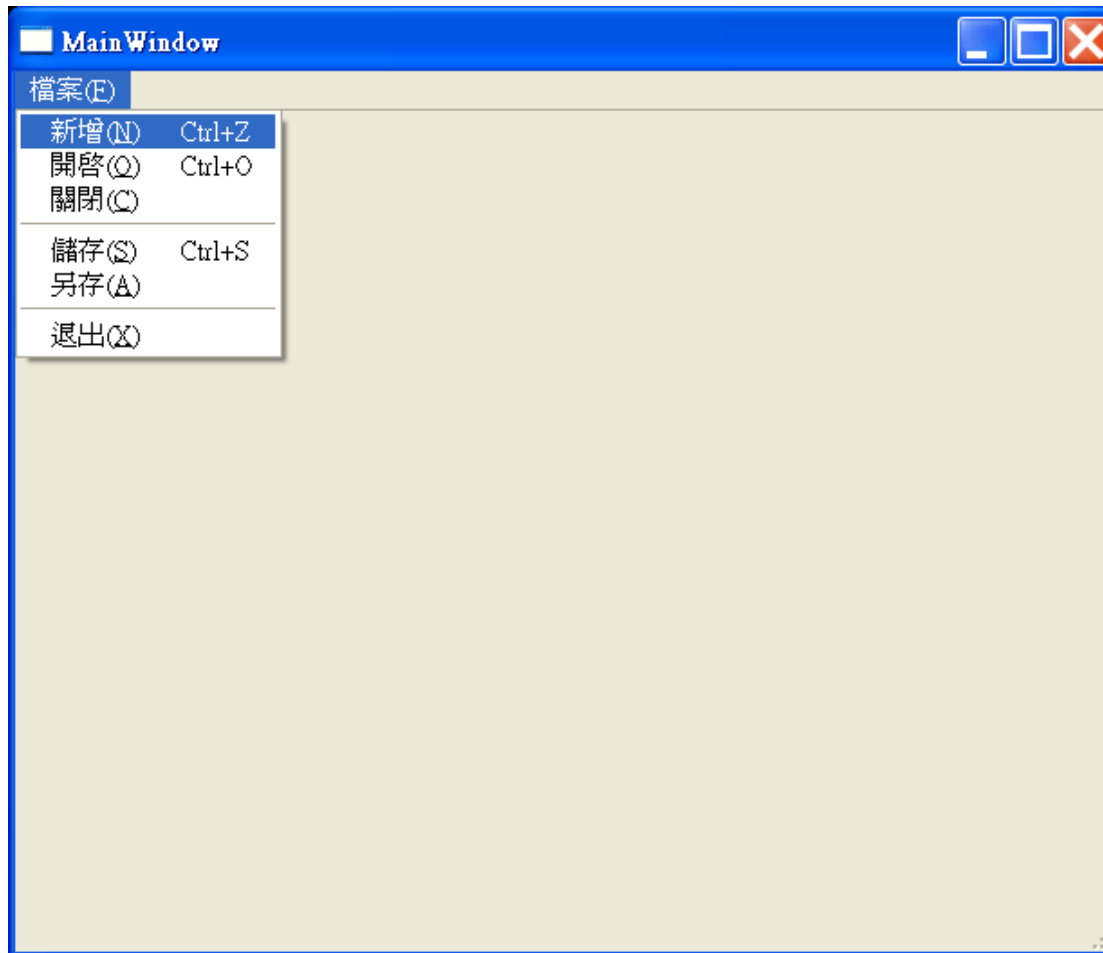
<resource root>

動作編輯器

名稱	已使用	文字	捷徑	可勾選
actionNew	✓	新增(&N)	Ctrl+Z	
actionOpen	✓	開啟(&O)	Ctrl+O	
actionClose	✓	關閉(&C)		
actionSave	✓	儲存(&S)	Ctrl+S	
actionASave	✓	另存(&A)		
actionQuit	✓	退出(&X)		

Qt- MainWindow

Qt Designer存檔後；Qt Creator 建置、執行專案



Qt- MainWindow

Qt Designer: 繼續新增“編輯”功能表及其功能表命令

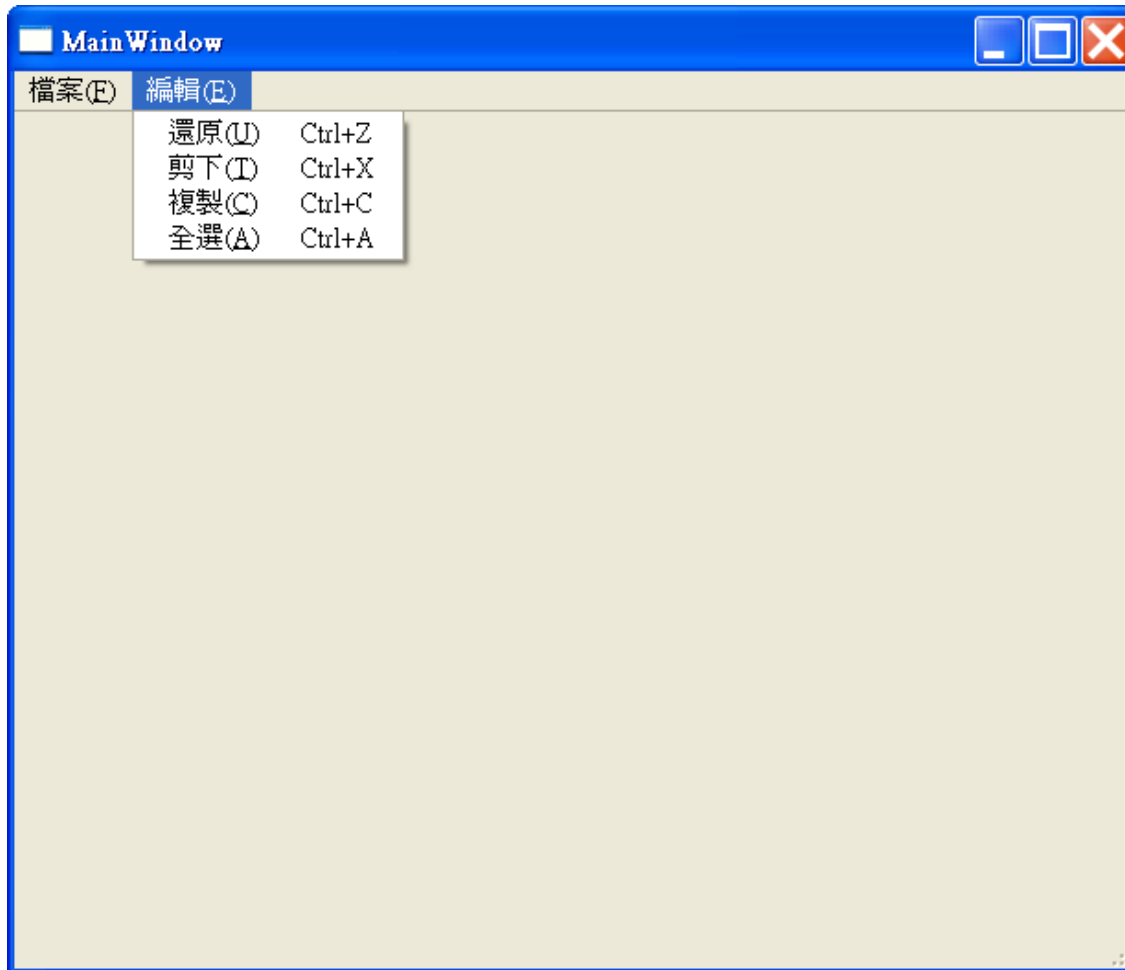
The screenshot shows the Qt Designer interface with a context menu open over the main window. The '動作編輯器' (Action Editor) dialog is also open, displaying a list of actions. The 'actionCopy' row is selected, highlighting its details.

名稱	已使用	文字	捷徑	可勾
actionNew	✓	新增(&N)	Ctrl+Z	
actionOpen	✓	開啓(&O)	Ctrl+O	
actionClose	✓	關閉(&C)		
actionSave	✓	儲存(&S)	Ctrl+S	
actionASave	✓	另存(&A)		
actionQuit	✓	退出(&X)		
actionUndo	✓	還原(&U)	Ctrl+Z	
actionCuT	✓	剪下(&T)	Ctrl+X	
actionAll	✓	全選(&A)	Ctrl+A	
actionCopy	✓	複製(&C)	Ctrl+C	

編輯功能表命令動作物件

Qt- MainWindow

Qt Designer存檔後；Qt Creator 建置、執行專案



Qt- MainWindow

Qt Creator 檢視 ui_mainwindow.h

The screenshot shows the Qt Creator IDE with the file `ui_mainwindow.h` open. The code defines a class `Ui_MainWindow` with the following members:

```

class Ui_MainWindow
{
public:
    QAction *actionNew;
    QAction *actionOpen;
    QAction *actionClose;
    QAction *actionSave;
    QAction *actionASave;
    QAction *actionQuit;
    QAction *actionUndo;
    QAction *actionCut;
    QAction *actionAll;
    QAction *actionCopy;
    QWidget *centralwidget;
    QMenuBar *menubar;
    QMenu *menu_F;
    QMenu *menu_E;
    QStatusBar *statusbar;

```

Annotations in Chinese explain the purpose of different groups of actions:

- 檔案功能表 命令物件!** (File menu command objects!): Points to the actions `actionNew` through `actionQuit`, which are enclosed in a blue box.
- 編輯功能表 命令物件!** (Edit menu command objects!): Points to the actions `actionUndo` through `actionCopy`, which are enclosed in a red box.
- 功能表列!** (Menu bar!): Points to the `menubar` member.
- 檔案功能表!** (File menu!): Points to the `menu_F` member.
- 編輯功能表!** (Edit menu!): Points to the `menu_E` member.

Qt- MainWindow

Qt Creator 檢視 ui_mainwindow.h

mainwindow.h - designwindow - Qt Creator

Build Debug Analyze Tools Window Help

projects ui_mainwindow.h menu_F: QMenu *

designwindow

- designwindow.pro
- Headers
 - mainwindow.h
 - ui_mainwindow.h
- Sources
 - main.cpp
 - mainwindow.cpp
- Forms
 - mainwindow.ui

```

71 MainWindow->setCentralWidget(centralwidget);
72 menubar = new QMenuBar(MainWindow);
73 menubar->setObjectName(QStringLiteral("menubar"));
74 menubar->setGeometry(QRect(0, 0, 609, 22));
75 menu_F = new QMenu(menubar);
76 menu_F->setObjectName(QStringLiteral("menu_F"));
77 menu_E = new QMenu(menubar);
78 menu_E->setObjectName(QStringLiteral("menu_E"));
79 MainWindow->setMenuBar(menubar);
80 statusbar = new QStatusBar(MainWindow);
81 statusbar->setObjectName(QStringLiteral("statusbar"));
82 MainWindow->setStatusBar(statusbar);
83
84 menubar->addAction(menu_F->menuAction());
85 menubar->addAction(menu_E->menuAction());
86 menu_F->addAction(actionNew);
87 menu_F->addAction(actionOpen);
88 menu_F->addAction(actionClose);
89 menu_F->addSeparator();
90 menu_F->addAction(actionSave);
91 menu_F->addAction(actionASave);
92 menu_F->addSeparator();
93 menu_F->addAction(actionQuit);
94 menu_E->addAction(actionUndo);
95 menu_E->addAction(actionCut);
96 menu_E->addAction(actionCopy);
97 menu_E->addAction(actionAll);

```

新增功能表!

新增功能表!

功能表加入功能表列!

建立檔案功能表
命令物件!

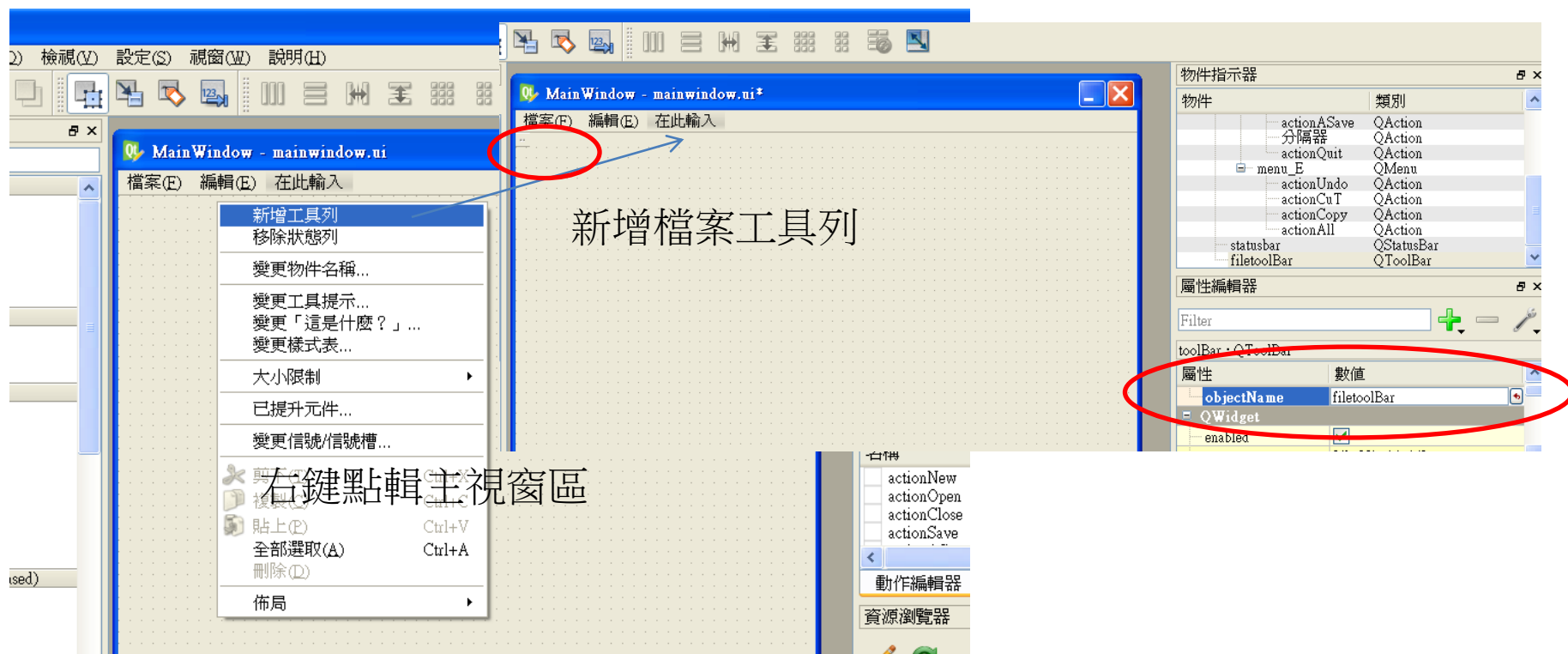
建立編輯功能表命
令物件!

open Documents

- designwindow.pro
- main.cpp
- mainwindow.cpp
- mainwindow.h
- ui_mainwindow.h

Qt- MainWindow

Qt Designer新增工具列：檔案工具列、編輯工具列



Qt- MainWindow

Qt Designer新增檔案動作物件至檔案工具列

Qt 設計師

檔案(F) 編輯(E) 在此輸入

新增(N) 開啟(O) 儲存(S) 另存(A)

左鍵拖曳動作物件至工具列

動作編輯器

名稱	已使用	文字	捷徑	可勾選	工
actionNew	✓	新增(&N)	Ctrl+Z		新
actionOpen	✓	開啟(&O)	Ctrl+O		開
actionClose	✓	關閉(&C)			關
actionSave	✓	儲存(&S)	Ctrl+S		儲
actionASave	✓	另存(&A)			另
actionQuit	✓	退出(&X)			退
actionUndo	✓	還原(&U)	Ctrl+Z		還
actionCuT	✓	剪下(&T)	Ctrl+X		剪
actionAll	✓	全選(&A)	Ctrl+A		全
actionCopy	✓	複製(&C)	Ctrl+C		複

物件指示器

物件	類別
actionUndo	QAction
actionCuT	QAction
actionCopy	QAction
actionAll	QAction
statusbar	QStatusBar
fileToolBar	QToolBar
actionNew	QAction
actionOpen	QAction
actionSave	QAction
actionASave	QAction

屬性編輯器

Filter

actionASave : QAction

屬性 數值

Object

objectName actionASave

QAction

checkable

checked

資源瀏覽器

Filter

<resource root>

信號/信號槽編輯器 資源瀏覽器

Qt- MainWindow

Qt Designer新增檔案動作物件至編輯工具列

Qt 設計師

檔案(E) 編輯(E) 表單(O) 檢視(V) 設定(S) 視窗(W) 說明(H)

元件盒

Filter

Layouts

- Vertical Layout
- Horizontal Layout
- Grid Layout
- Form Layout

Spacers

- Horizontal Spacer
- Vertical Spacer

Buttons

- Push Button
- Tool Button
- Radio Button
- Check Box
- Command Link Button
- Button Box

Item Views (Model-Based)

- List View
- Tree View
- Table View
- Column View

Item Widgets (Item-Based)

- List Widget
- Tree Widget
- Table Widget

Containers

- Group Box
- Scroll Area

MainWindow - mainwindow.ui*

檔案(E) 編輯(E) 在此輸入

新增(N) 開啟(O) 儲存(S) 另存(A) 還原(U) 剪下(T) 複製(C) 貼上(P)

左鍵拖曳動作物件至工具列

動作編輯器

名稱	已使用	文字	捷徑	可
actionNew	✓	新增(&N)	Ctrl+Z	
actionOpen	✓	開啟(&O)	Ctrl+O	
actionClose	✓	關閉(&C)		
actionSave	✓	儲存(&S)	Ctrl+S	
actionASave	✓	另存(&A)		
actionQuit	✓	退出(&X)		
actionUndo	✓	還原(&U)	Ctrl+Z	
actionCuT	✓	剪下(&T)	Ctrl+X	
actionAll	✓	全選(&A)	Ctrl+A	
actionCopy	✓	複製(&C)	Ctrl+C	
actionPaste	✓	貼上(&P)	Ctrl+V	

物件指示器

物件	類別
fileToolBar	QToolBar
actionNew	QAction
actionOpen	QAction
actionSave	QAction
actionASave	QAction
toolBar	QToolBar
actionUndo	QAction
actionCuT	QAction
actionCopy	QAction
actionPaste	QAction

屬性編輯器

Filter

actionPaste : QAction

屬性 數值

- QObject
 - objectName actionPaste
- QAction
 - checkable ☐
 - checked ☐

資源瀏覽器

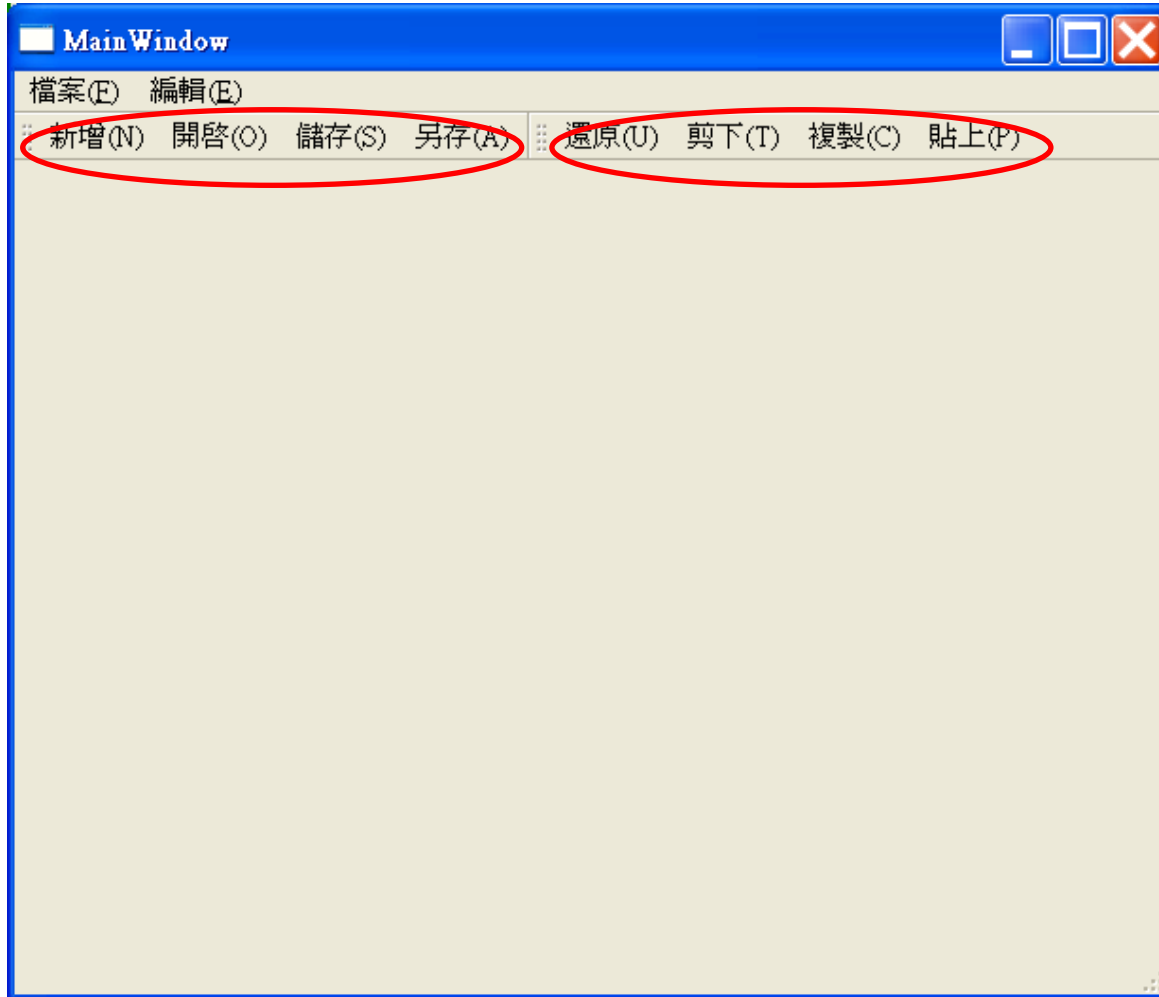
Filter

<resource root>

信號/信號槽編輯器 資源瀏覽器

Qt- MainWindow

Qt Designer存檔後；Qt Creator 建置、執行專案



Qt- MainWindow

Qt Creator 檢視 ui_mainwindow.h

```

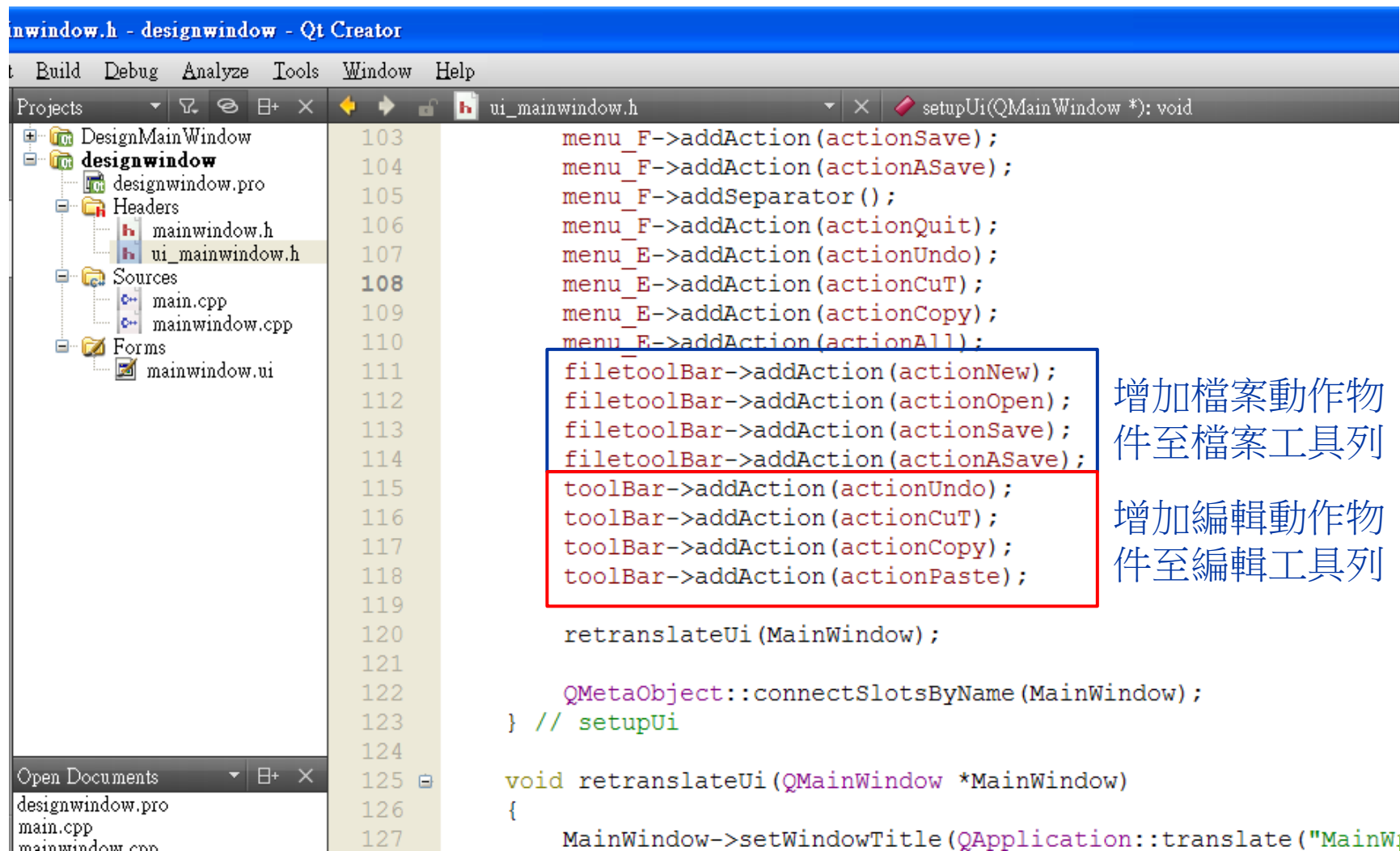
73  actionPaste = new QAction(MainWindow);
74  actionPaste->setObjectName(QStringLiteral("actionPaste"));
75  centralwidget = new QWidget(MainWindow);
76  centralwidget->setObjectName(QStringLiteral("centralwidget"));
77  MainWindow->setCentralWidget(centralwidget);
78  menubar = new QMenuBar(MainWindow);
79  menubar->setObjectName(QStringLiteral("menubar"));
80  menubar->setGeometry(QRect(0, 0, 609, 22));
81  menu_F = new QMenu(menubar);
82  menu_F->setObjectName(QStringLiteral("menu_F"));
83  menu_E = new QMenu(menubar);
84  menu_E->setObjectName(QStringLiteral("menu_E"));
85  MainWindow->setMenuBar(menubar);
86  statusbar = new QStatusBar(MainWindow);
87  statusbar->setObjectName(QStringLiteral("statusbar"));
88  MainWindow->setStatusBar(statusbar);
89  filetoolBar = new QToolBar(MainWindow);
90  filetoolBar->setObjectName(QStringLiteral("filetoolBar"));
91  MainWindow->addToolBar(Qt::TopToolBarArea, filetoolBar);
92  MainWindow->insertToolBarBreak(filetoolBar);
93  edittoolBar = new QToolBar(MainWindow);
94  edittoolBar->setObjectName(QStringLiteral("edittoolBar"));
95  MainWindow->addToolBar(Qt::TopToolBarArea, edittoolBar);
96
97  menubar->addAction(menu_F->menuAction());
98  menubar->addAction(menu_E->menuAction());
99  menu_F->addAction(actionNew);
  
```

增加檔案工具列

增加編輯工具列

Qt- MainWindow

Qt Creator 檢視 ui_mainwindow.h



mainwindow.h - designwindow - Qt Creator

Build Debug Analyze Tools Window Help

Projects

- DesignMainWindow
 - designwindow
 - designwindow.pro
 - Headers
 - mainwindow.h
 - ui_mainwindow.h
 - Sources
 - main.cpp
 - mainwindow.cpp
 - Forms
 - mainwindow.ui

Open Documents

- designwindow.pro
- main.cpp
- mainwindow.cpp

```

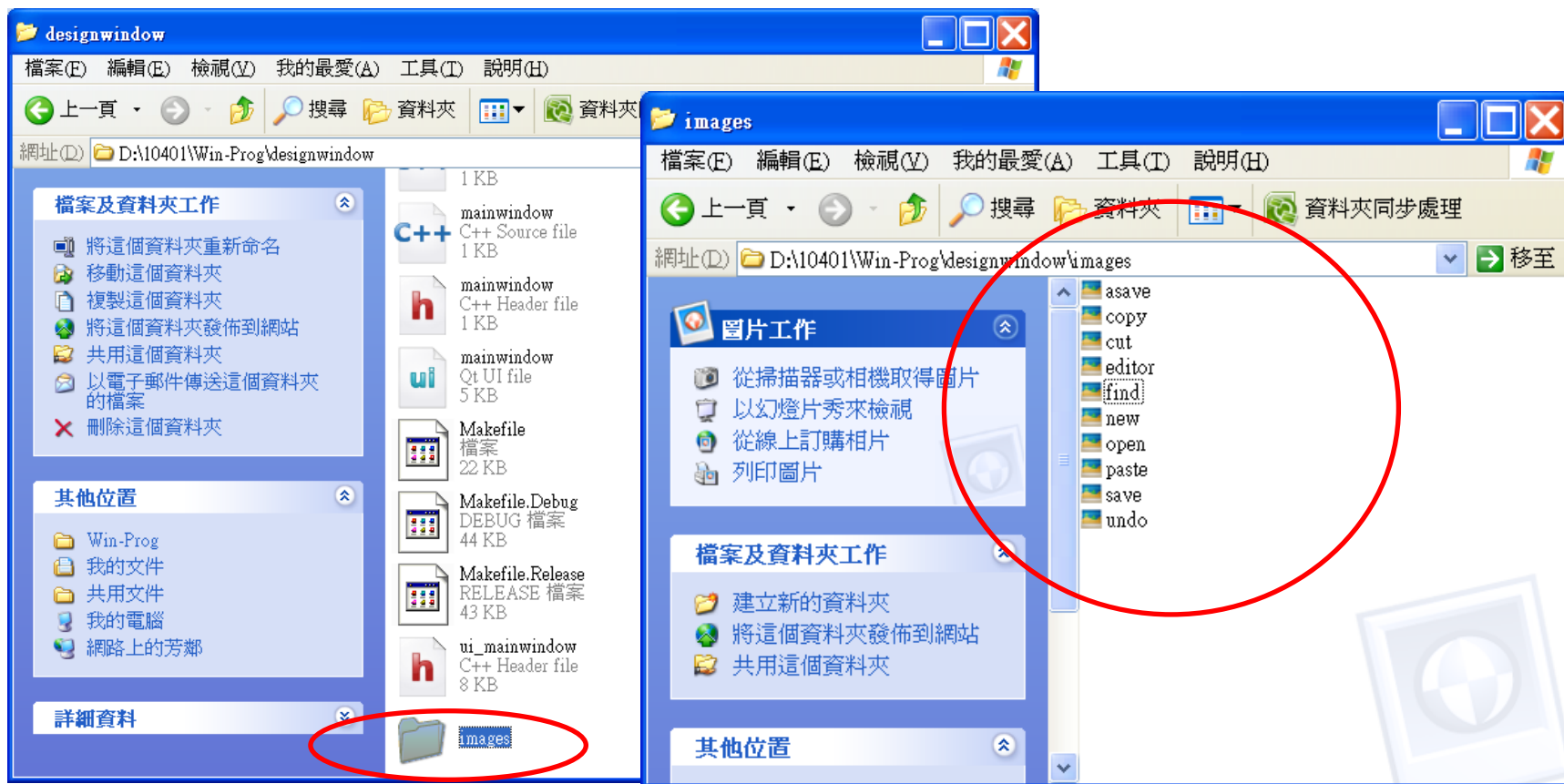
103     menu_F->addAction(actionSave);
104     menu_F->addAction(actionASave);
105     menu_F->addSeparator();
106     menu_F->addAction(actionQuit);
107     menu_E->addAction(actionUndo);
108     menu_E->addAction(actionCuT);
109     menu_E->addAction(actionCopy);
110     menu_E->addAction(actionAll);
111     fileToolBar->addAction(actionNew);
112     fileToolBar->addAction(actionOpen);
113     fileToolBar->addAction(actionSave);
114     fileToolBar->addAction(actionASave);
115     toolBar->addAction(actionUndo);
116     toolBar->addAction(actionCuT);
117     toolBar->addAction(actionCopy);
118     toolBar->addAction(actionPaste);
119
120     retranslateUi(MainWindow);
121
122     QMetaObject::connectSlotsByName(MainWindow);
123 } // setupUi
124
125 void retranslateUi(QMainWindow *MainWindow)
126 {
127     MainWindow->setWindowTitle(QApplication::translate("MainW
  
```

增加檔案動作物件至檔案工具列

增加編輯動作物件至編輯工具列

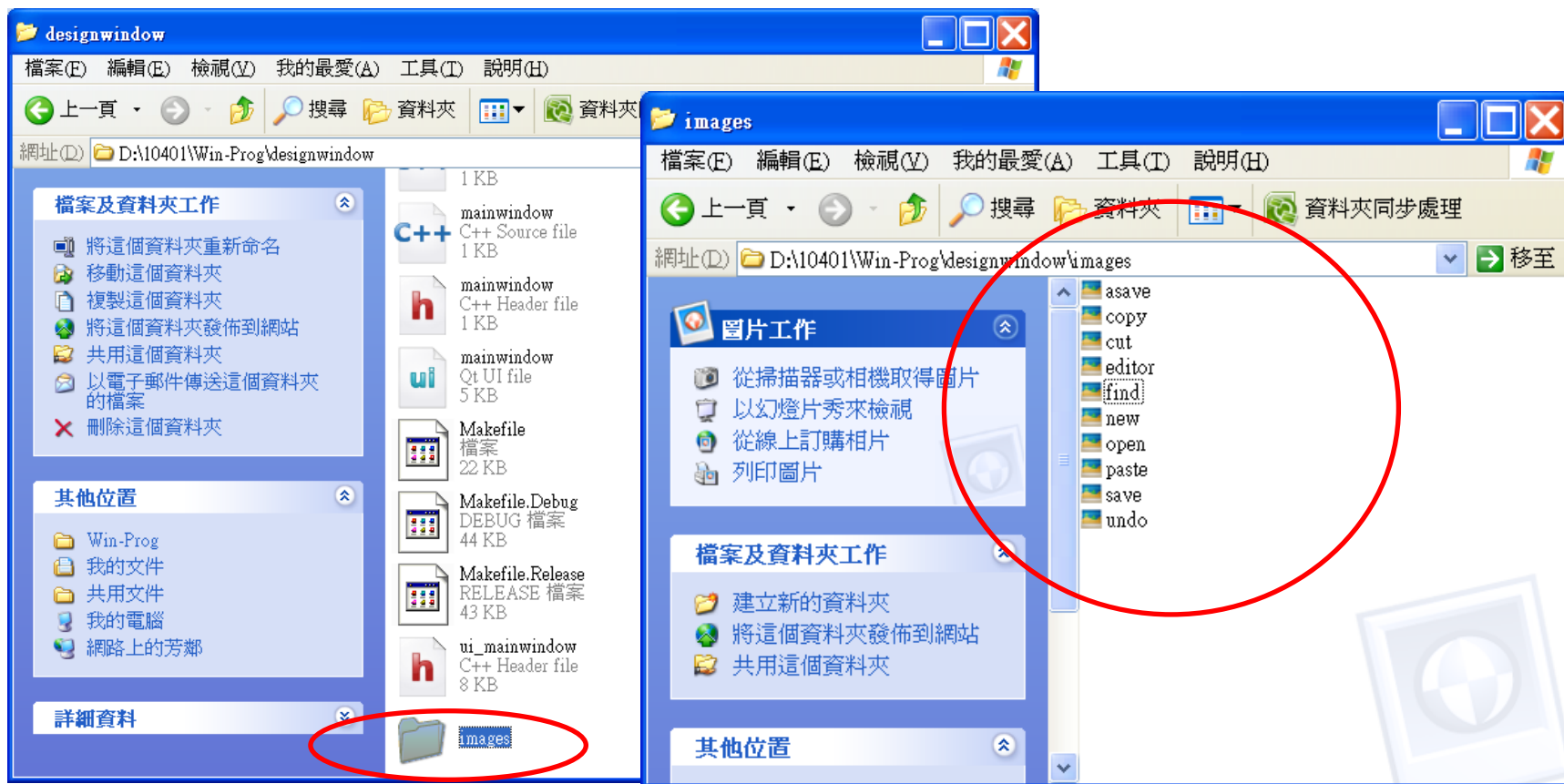
Qt- MainWindow

新增資源檔(resource)：為動作物件加入圖示(icon)
複製圖示檔案至專案資料夾下，置於images子資料夾內



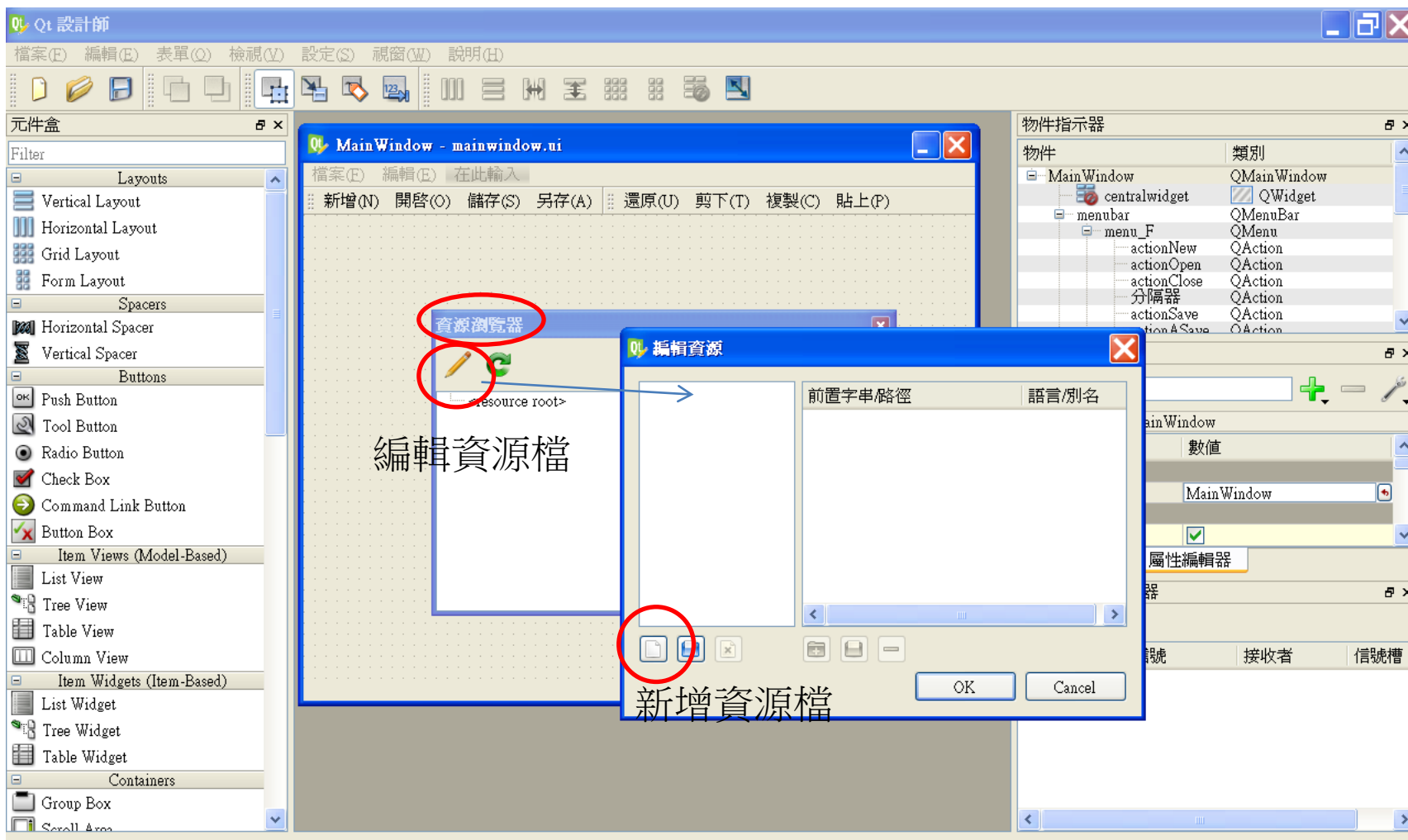
Qt- MainWindow

新增資源檔(resource)：為動作物件加入圖示(icon)
複製圖示檔案至專案資料夾下，置於images子資料夾內



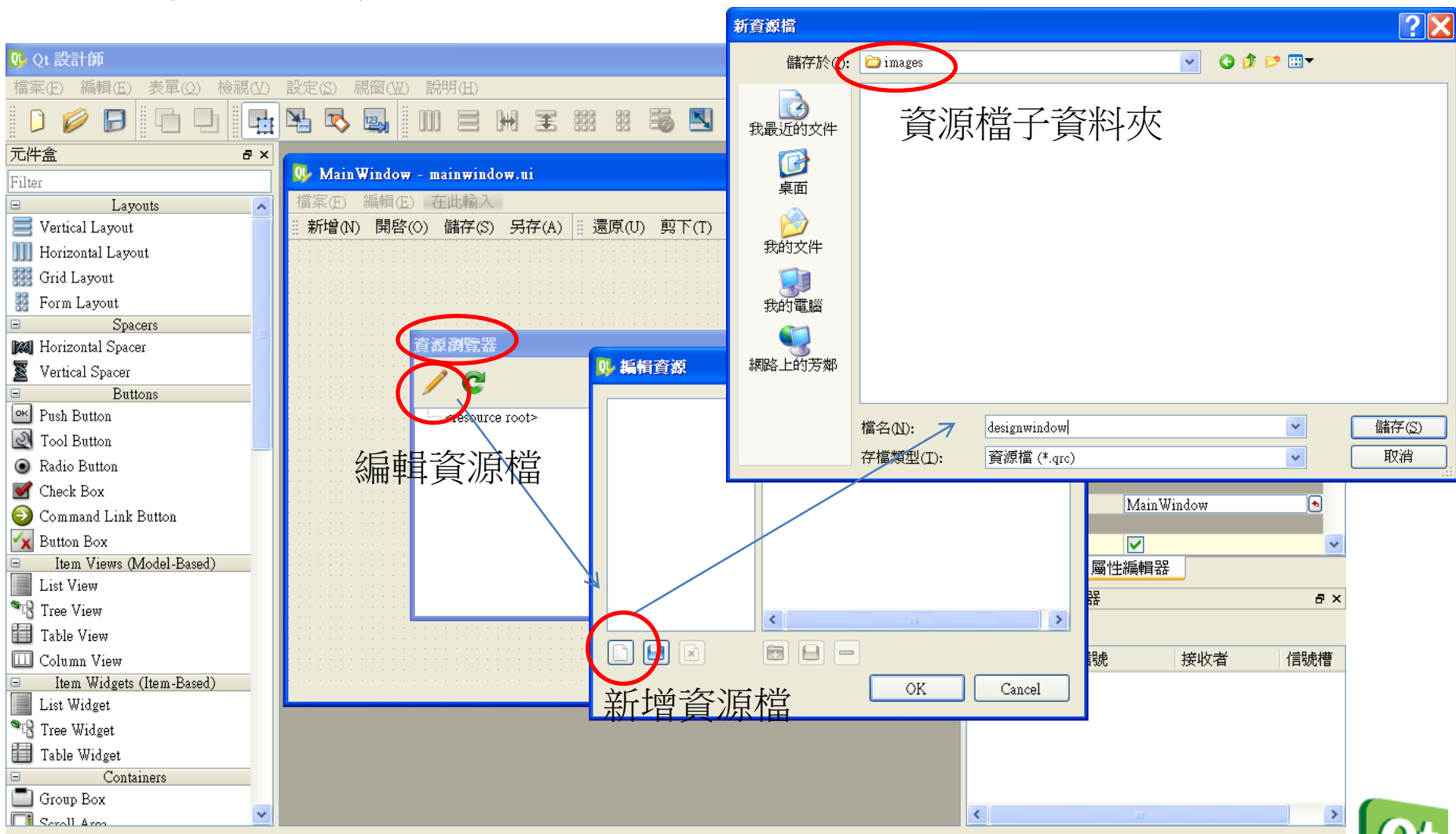
Qt- MainWindow

Qt Designer新增資源檔



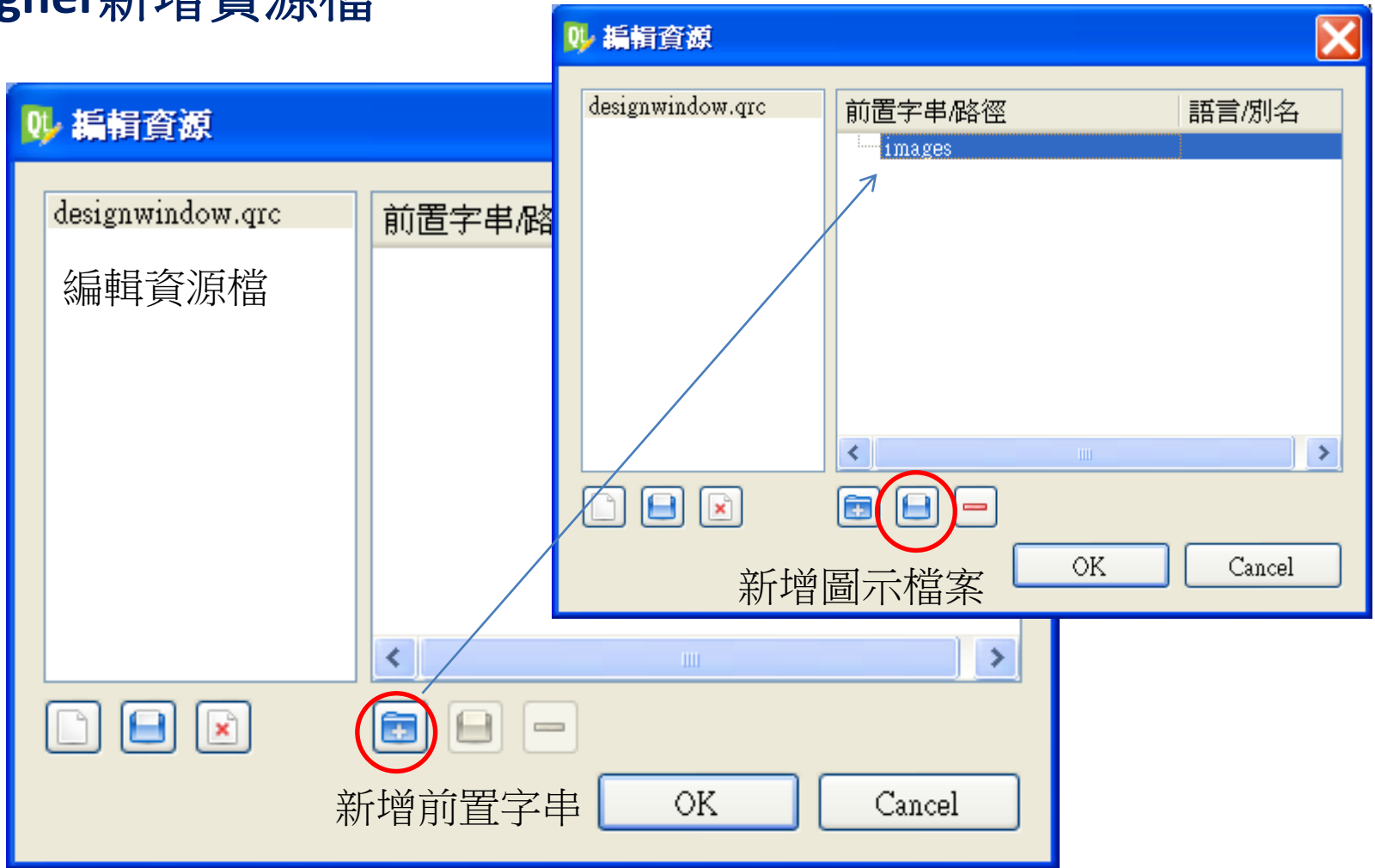
Qt- MainWindow

Qt Designer新增資源檔



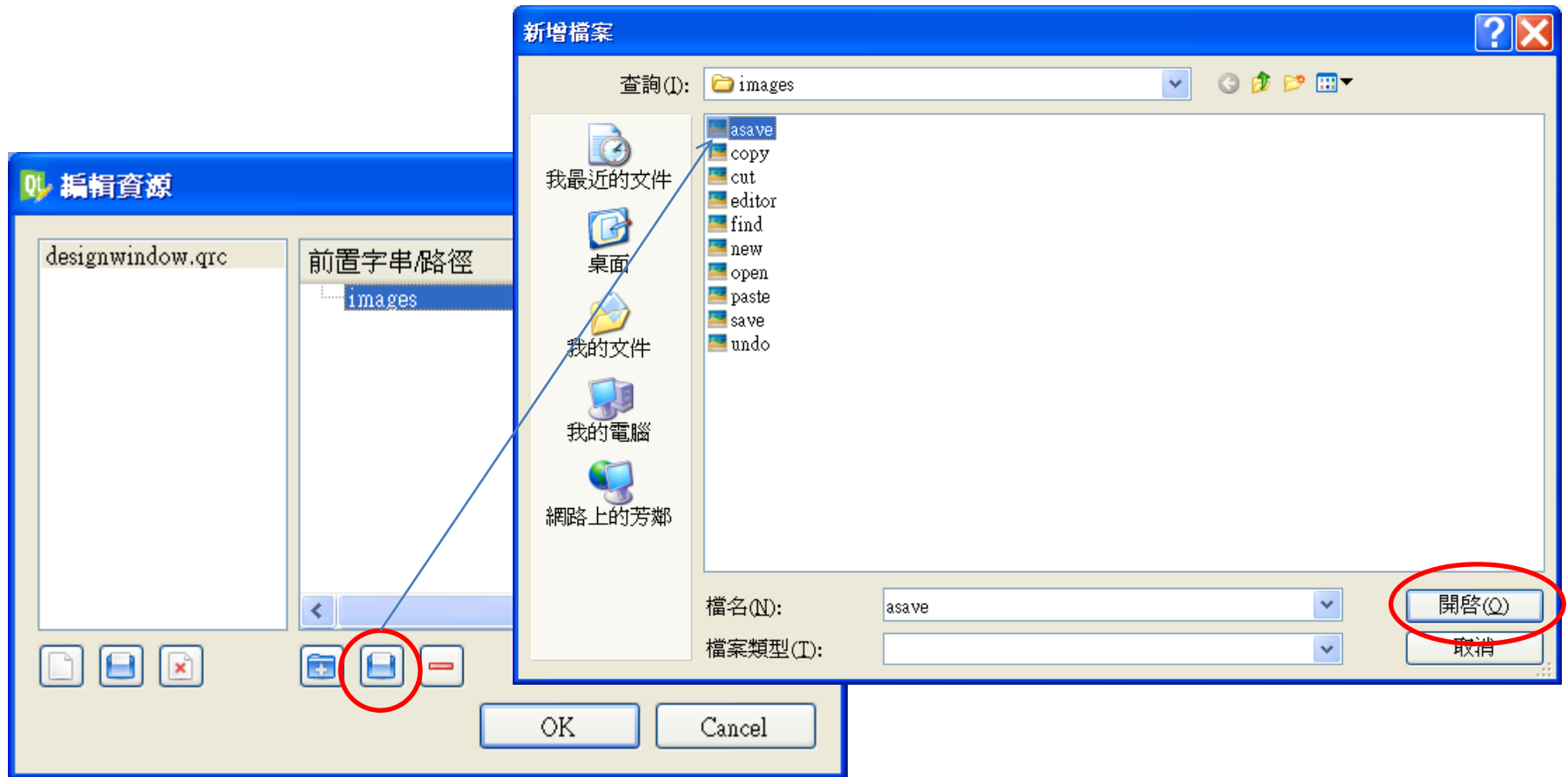
Qt- MainWindow

Qt Designer新增資源檔



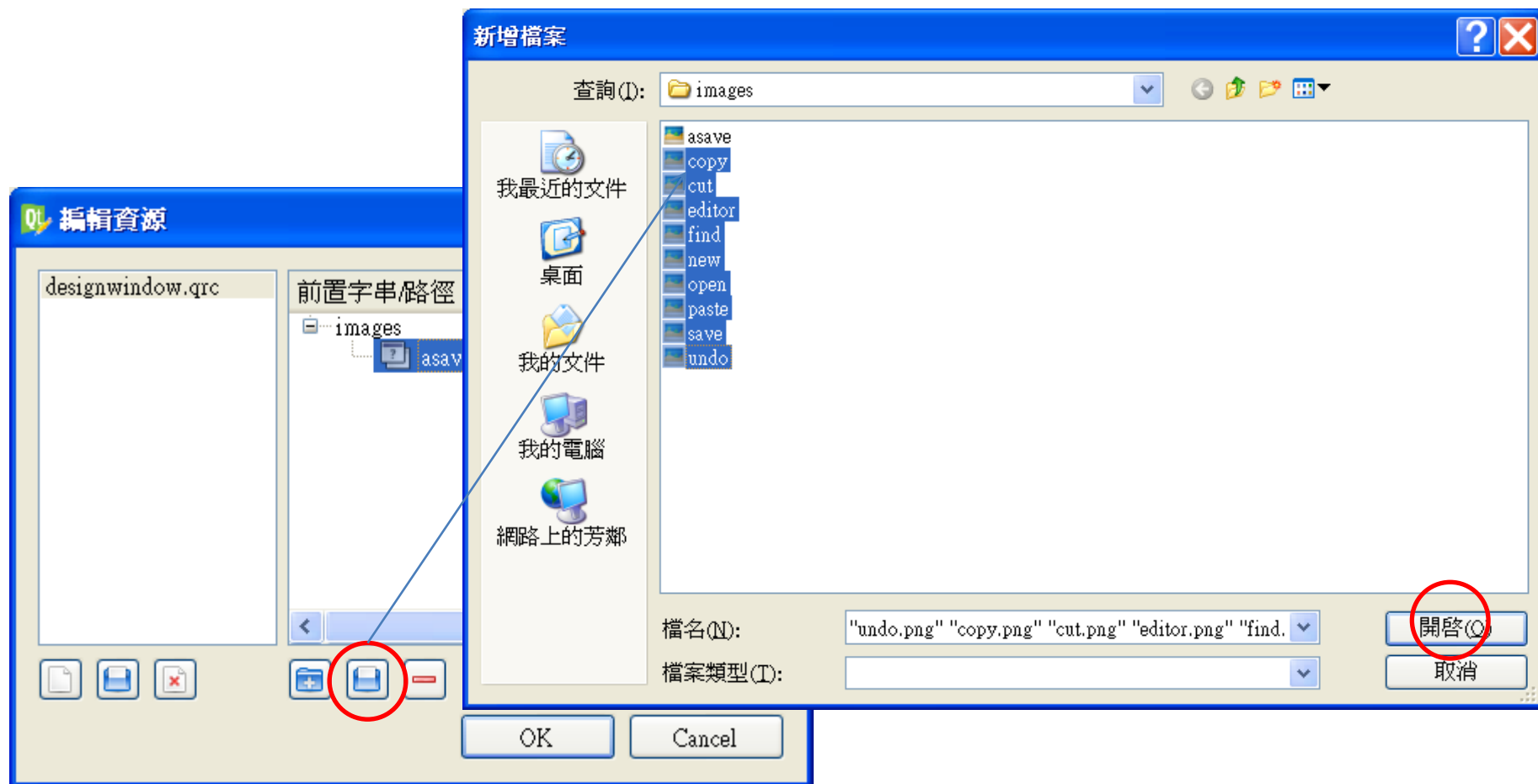
Qt- MainWindow

Qt Designer新增資源圖示檔



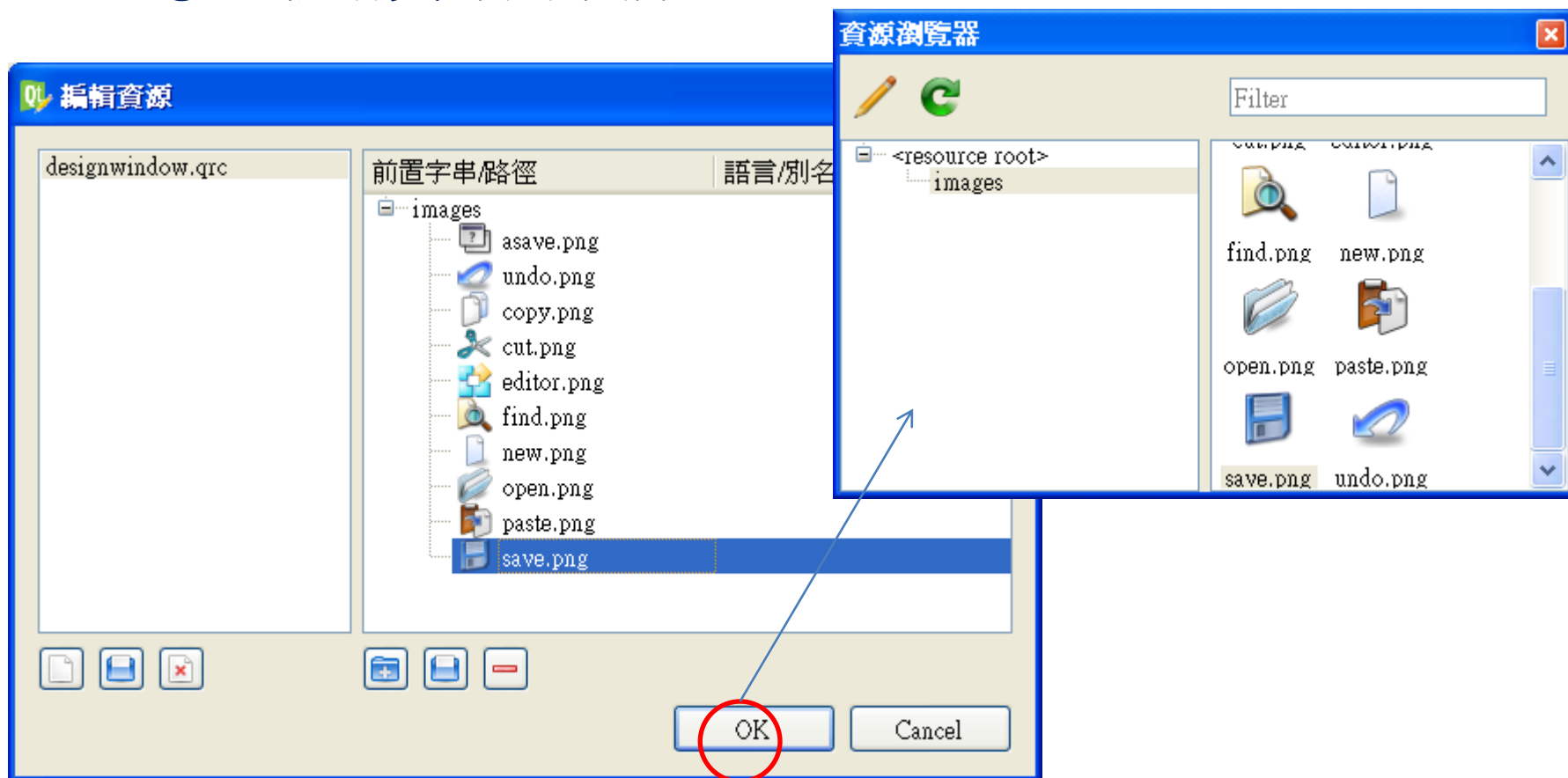
Qt- MainWindow

Qt Designer新增資源圖示檔



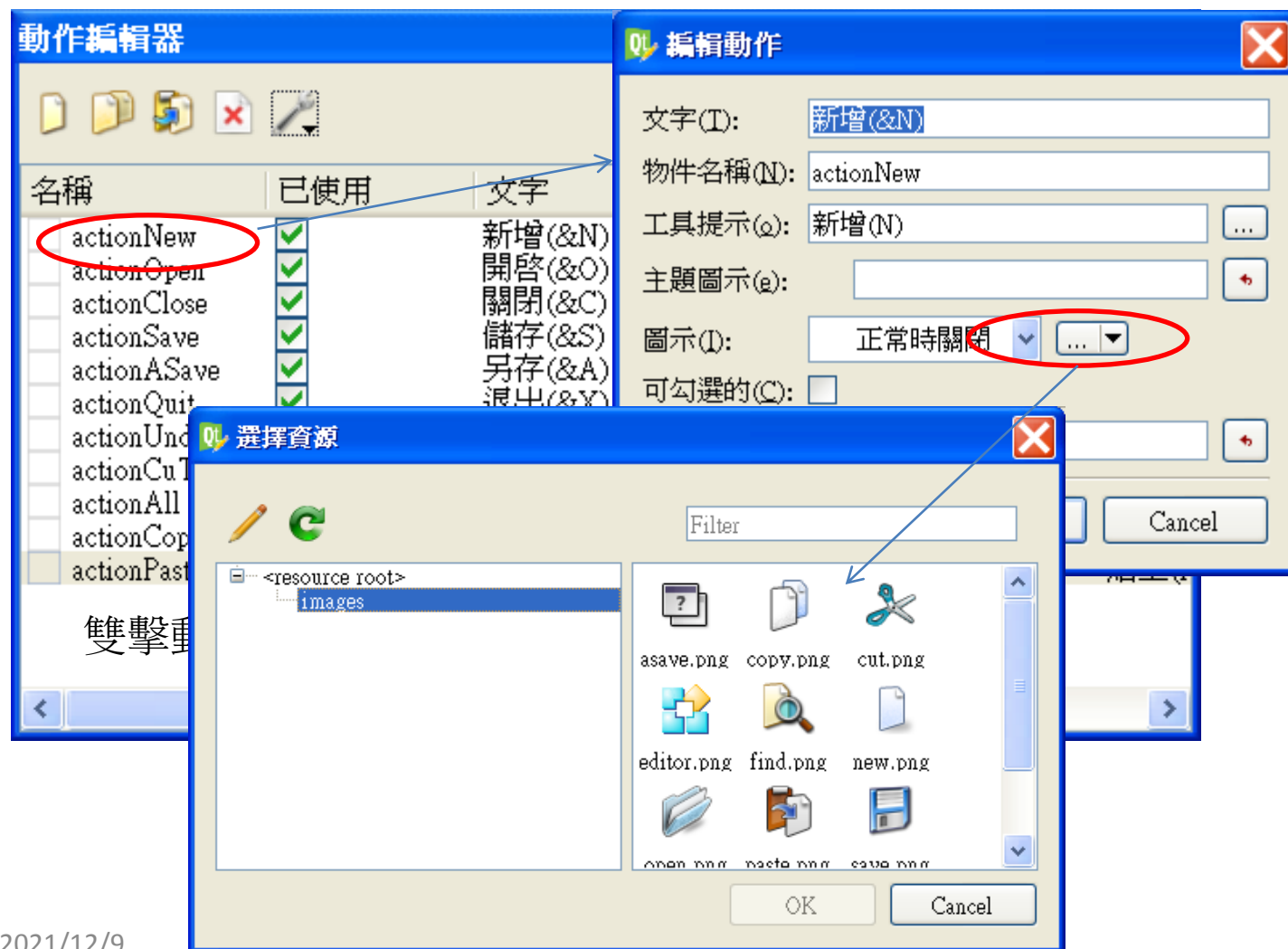
Qt- MainWindow

Qt Designer新增資源圖示檔



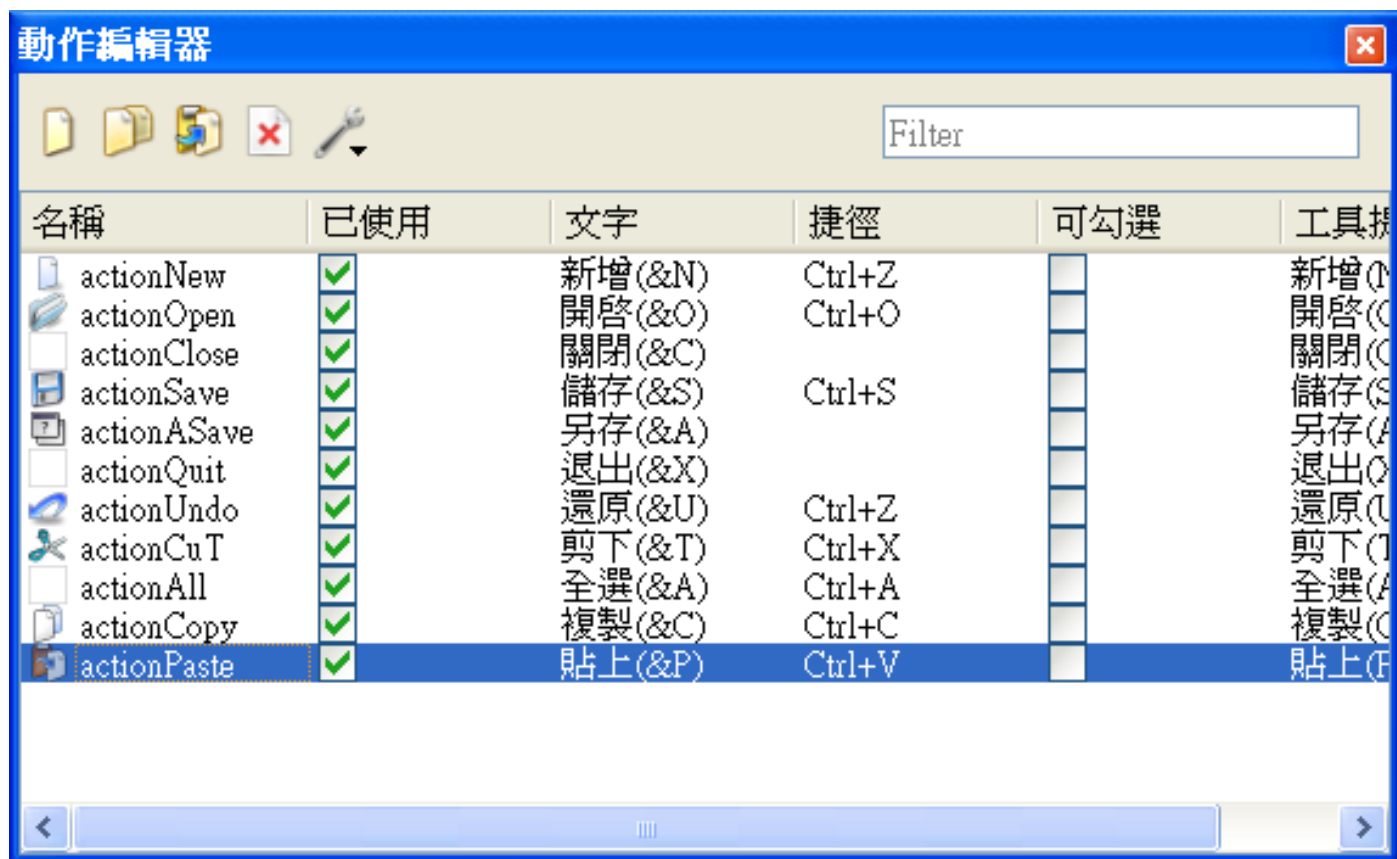
Qt- MainWindow

Qt Designer添加資源圖示給動作物件



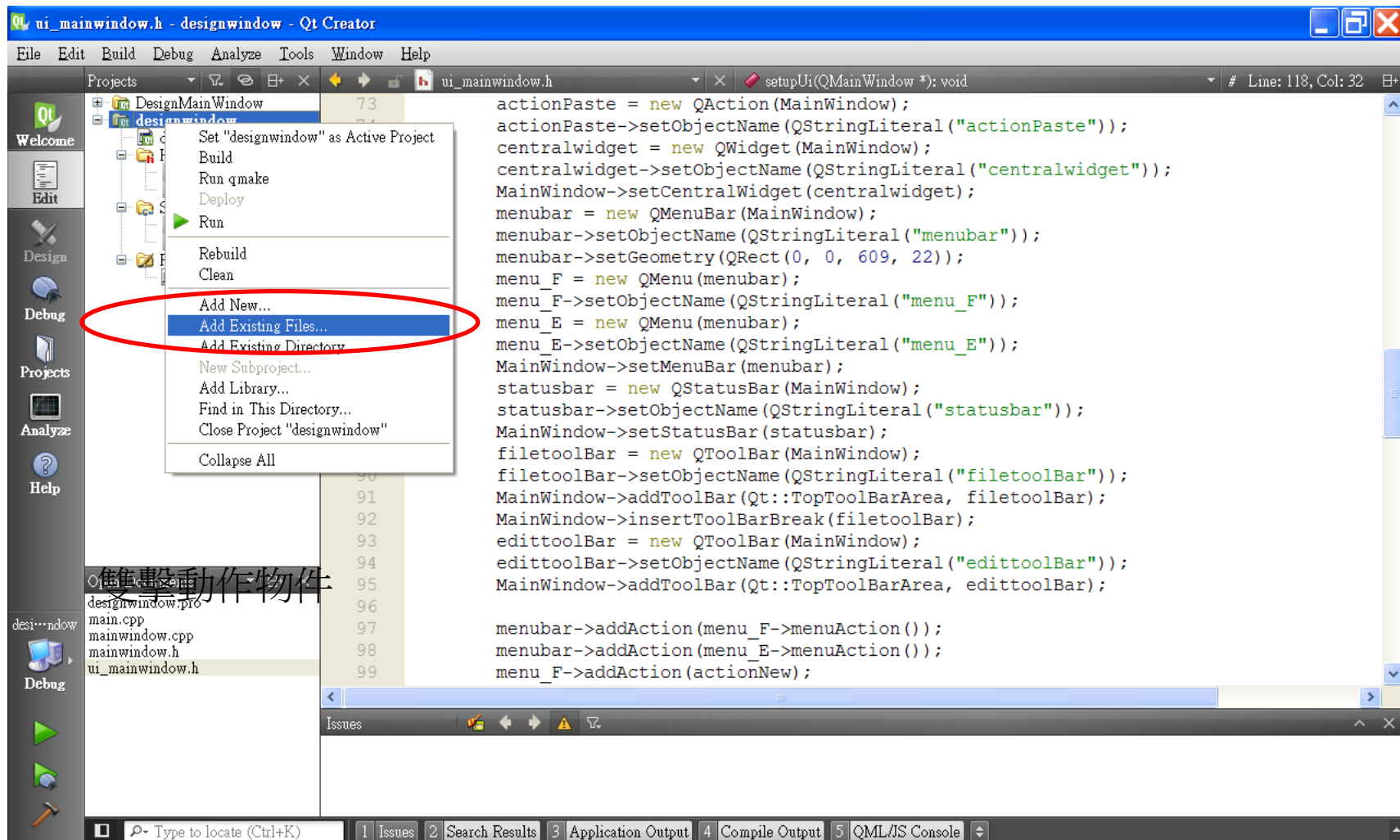
Qt- MainWindow

Qt Designer依下圖完成添加資源圖示給其它動作物件



Qt- MainWindow

Qt Creator: 添加資源檔至專案檔中

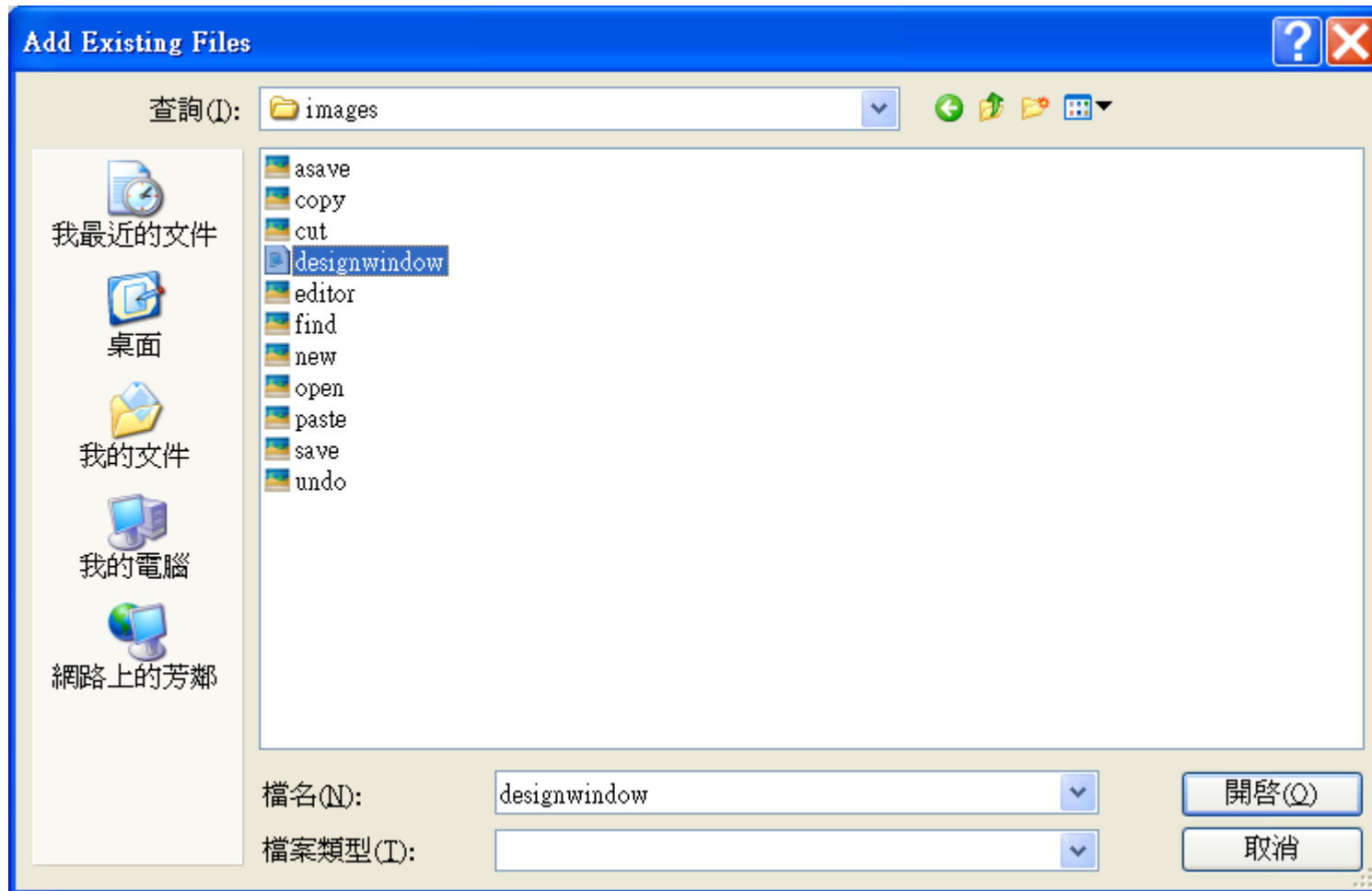




Code less.
Create more.
Deploy everywhere.

Qt- MainWindow

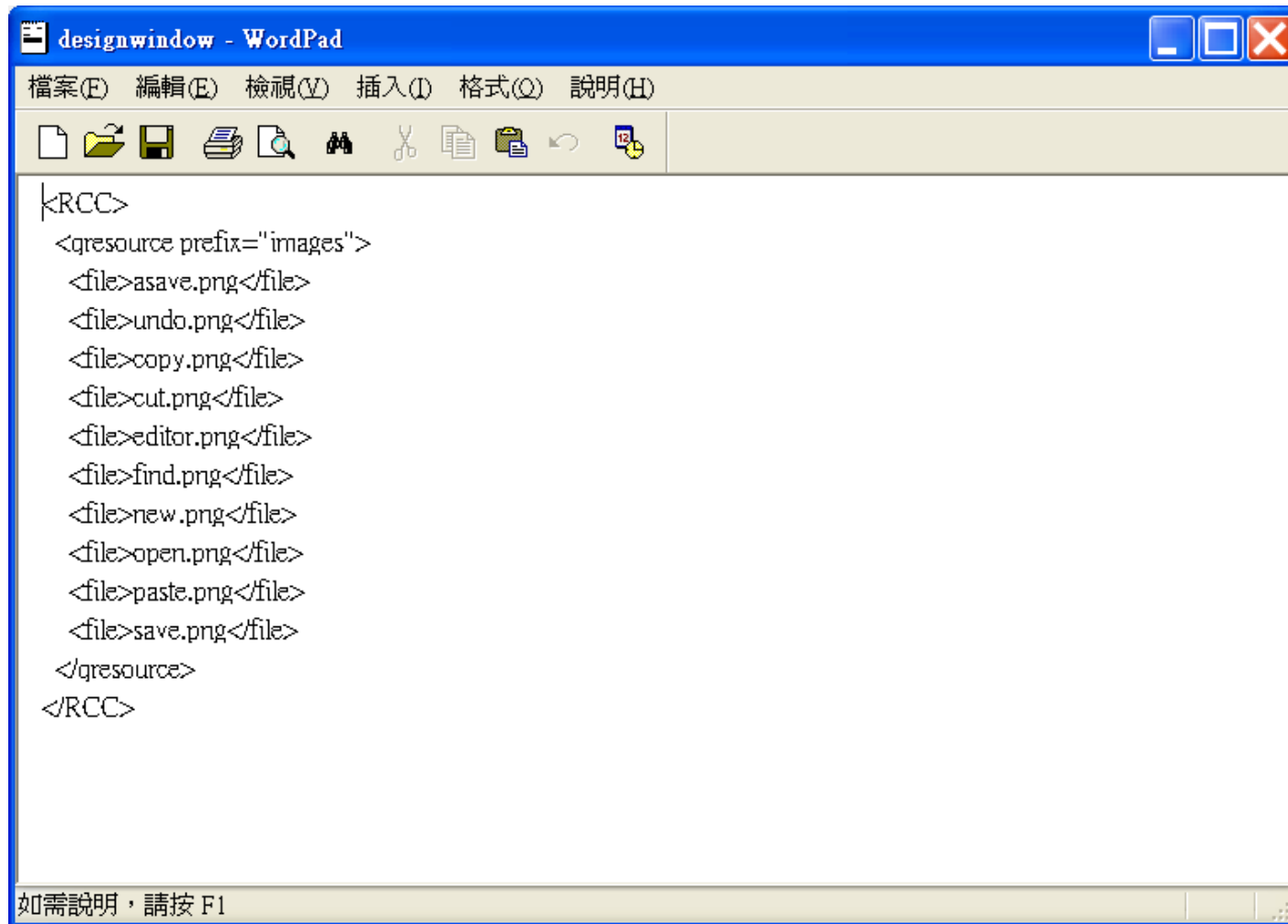
Qt Creator: 添加資源檔至專案檔中





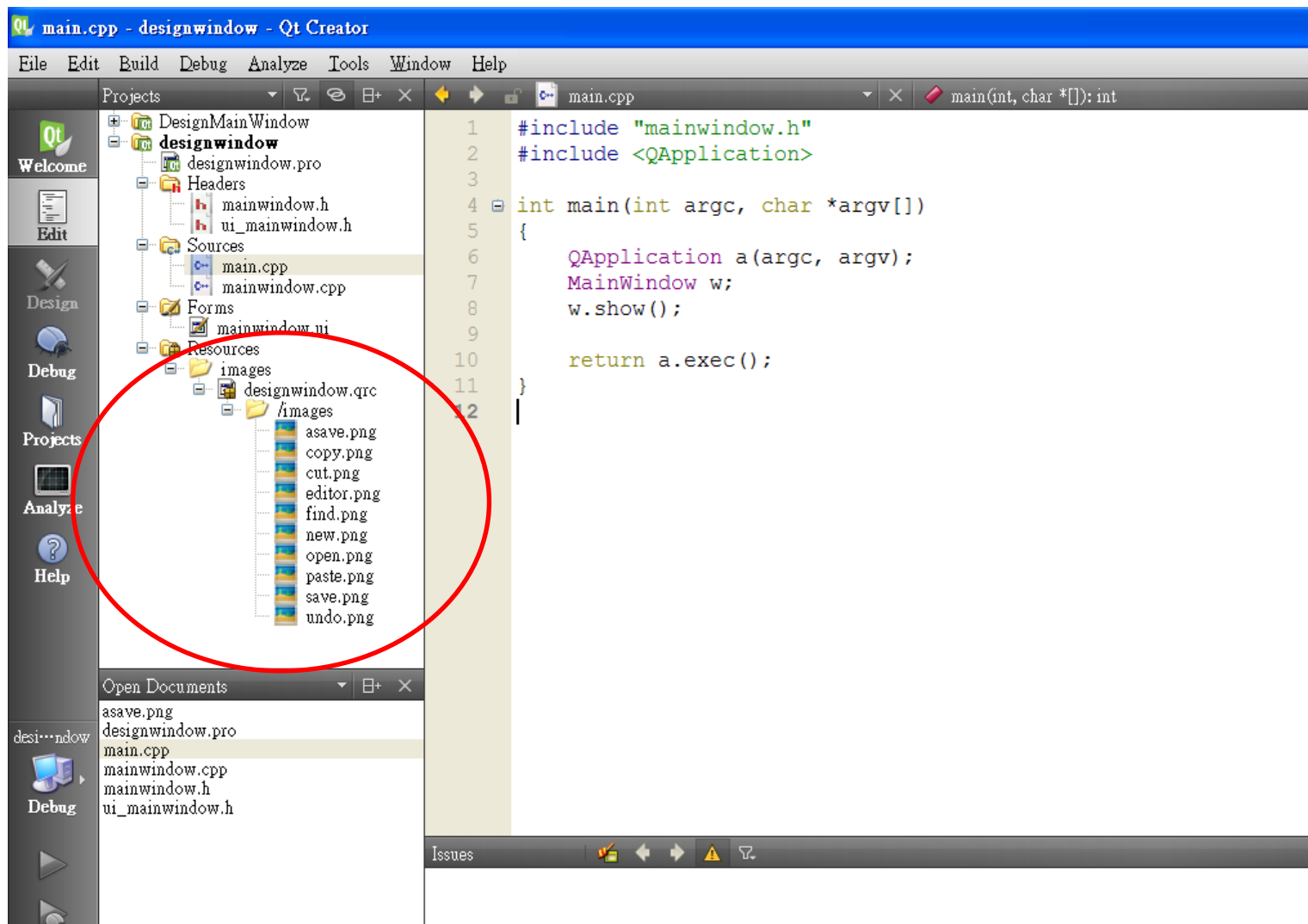
Qt- MainWindow

Qt Creator: 添加資源檔至專案檔中



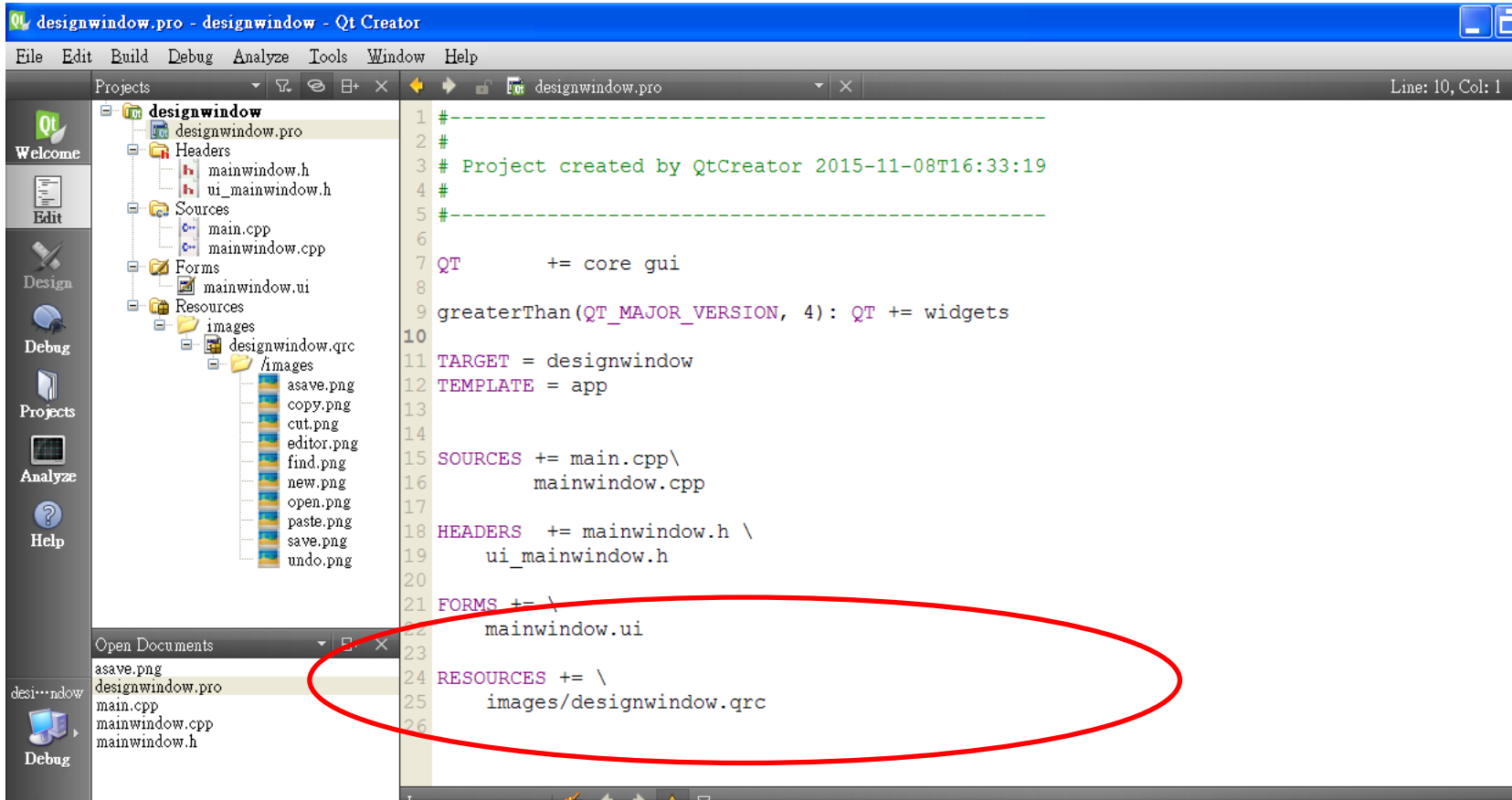
Qt- MainWindow

Qt Creator: 添加資源檔至專案檔中



Qt- MainWindow

Qt Creator: 添加資源檔至專案檔中



The screenshot displays two overlapping windows of a software application. The top window, titled 'MainWindow', features a menu bar with '檔案(F)' and '編輯(E)'. The '編輯(E)' menu is currently open, revealing a list of editing actions: '還原(U)' (Ctrl+Z), '剪下(T)' (Ctrl+X), '複製(C)' (Ctrl+C), and '全選(A)' (Ctrl+A). The bottom window, also titled 'MainWindow', is partially obscured by the top one. Red circles are drawn around the '檔案(F)' menu and the '編輯(E)' menu in the top window.

Qt- MainWindow

Qt Creator 檢視 ui_mainwindow.h

The screenshot shows the Qt Creator IDE with the file `ui_mainwindow.h` open. The code defines a `MainWindow` class that inherits from `QMainWindow`. It includes various actions like `actionNew`, `actionOpen`, `actionClose`, `actionSave`, `actionASave`, and `actionQuit`. It also includes icons like `icon`, `icon1`, `icon2`, and `icon3`. The code is as follows:

```

50 if (MainWindow->objectName().isEmpty())
51     MainWindow->setObjectName(QStringLiteral("MainWindow"));
52 MainWindow->resize(609, 488);
53 actionNew = new QAction(MainWindow);
54 actionNew->setObjectName(QStringLiteral("actionNew"));
55 QIcon icon;
56 icon.addFile(QStringLiteral(":/images/new.png"), QSize(), QIcon::Normal, QIcon::Off);
57 actionNew->setIcon(icon);
58 actionOpen = new QAction(MainWindow);
59 actionOpen->setObjectName(QStringLiteral("actionOpen"));
60 QIcon icon1;
61 icon1.addFile(QStringLiteral(":/images/open.png"), QSize(), QIcon::Normal, QIcon::Off);
62 actionOpen->setIcon(icon1);
63 actionClose = new QAction(MainWindow);
64 actionClose->setObjectName(QStringLiteral("actionClose"));
65 actionSave = new QAction(MainWindow);
66 actionSave->setObjectName(QStringLiteral("actionSave"));
67 QIcon icon2;
68 icon2.addFile(QStringLiteral(":/images/save.png"), QSize(), QIcon::Normal, QIcon::Off);
69 actionSave->setIcon(icon2);
70 actionASave = new QAction(MainWindow);
71 actionASave->setObjectName(QStringLiteral("actionASave"));
72 QIcon icon3;
73 icon3.addFile(QStringLiteral(":/images/asave.png"), QSize(), QIcon::Normal, QIcon::Off);
74 actionASave->setIcon(icon3);
75 actionQuit = new QAction(MainWindow);
76 actionQuit->setObjectName(QStringLiteral("actionQuit"));

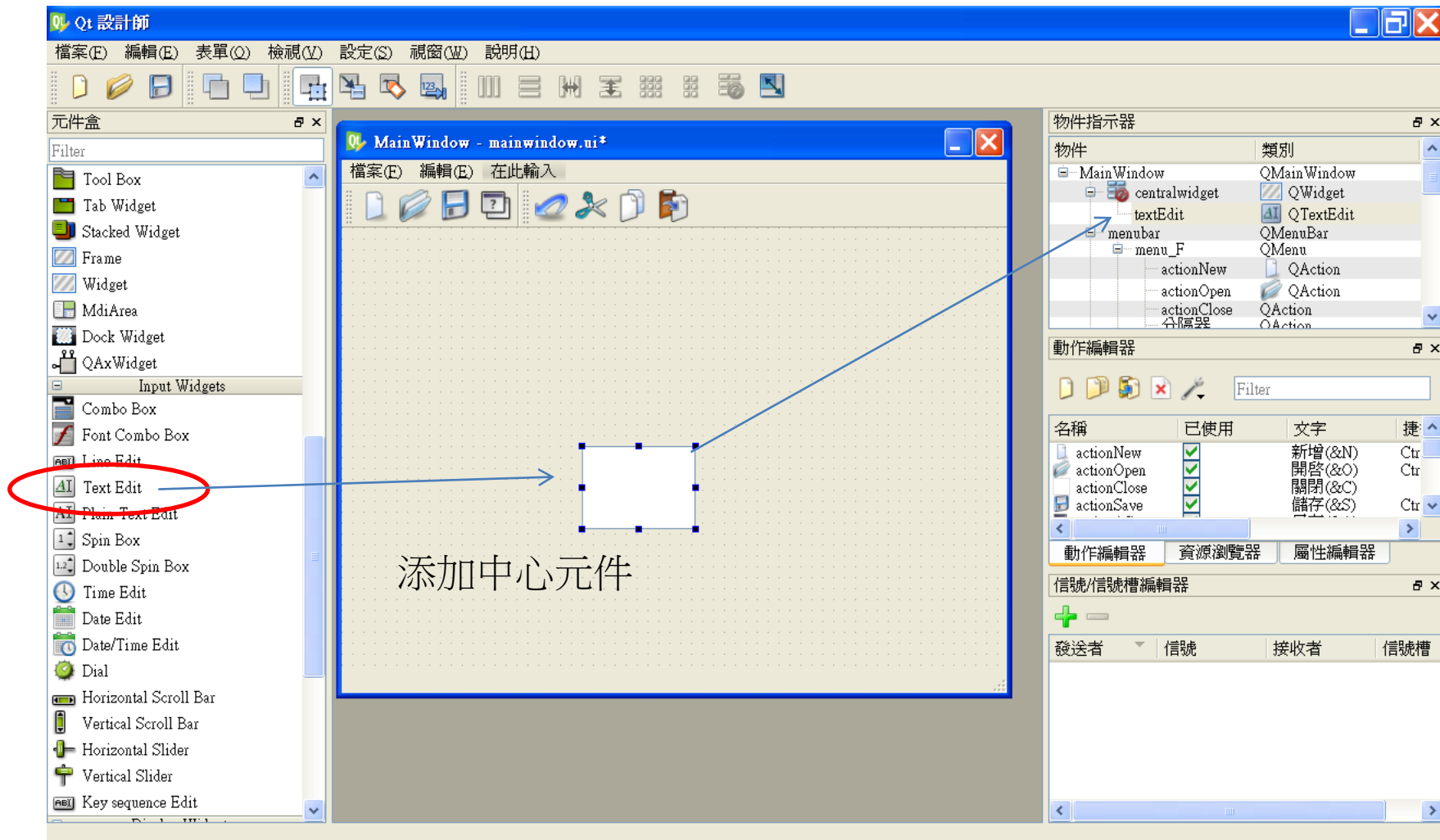
```

Annotations in the image:

- Line 55: `QIcon icon;` is highlighted with a blue box and the text "新增QIcon物件!" (Add QIcon object!).
- Line 59: `actionOpen->setObjectName(QStringLiteral("actionOpen"));` is highlighted with a blue box and the text "設定icon物件!" (Set icon object!).

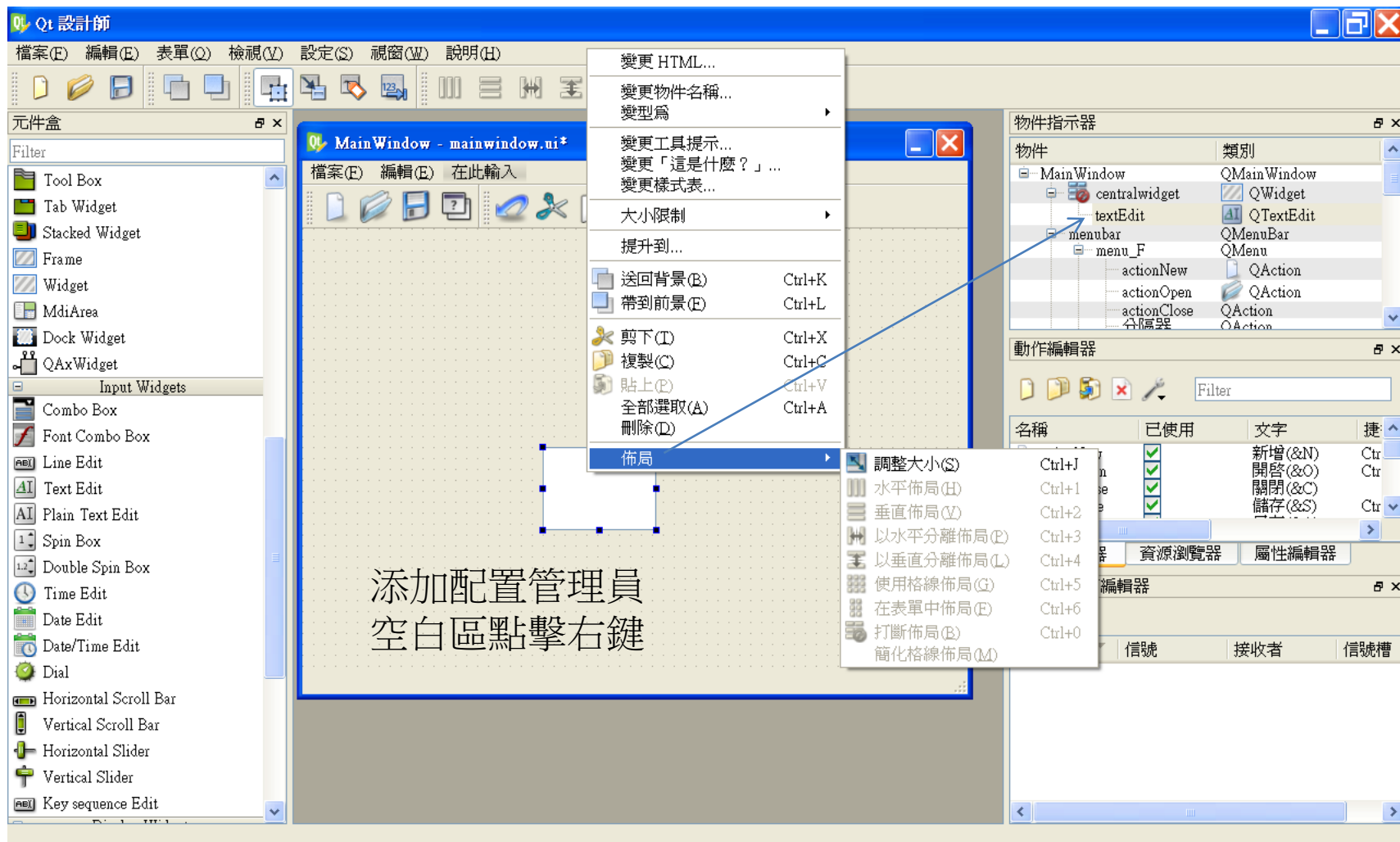
Qt- MainWindow

Qt Designer添加視窗中心元件(centralWidget)



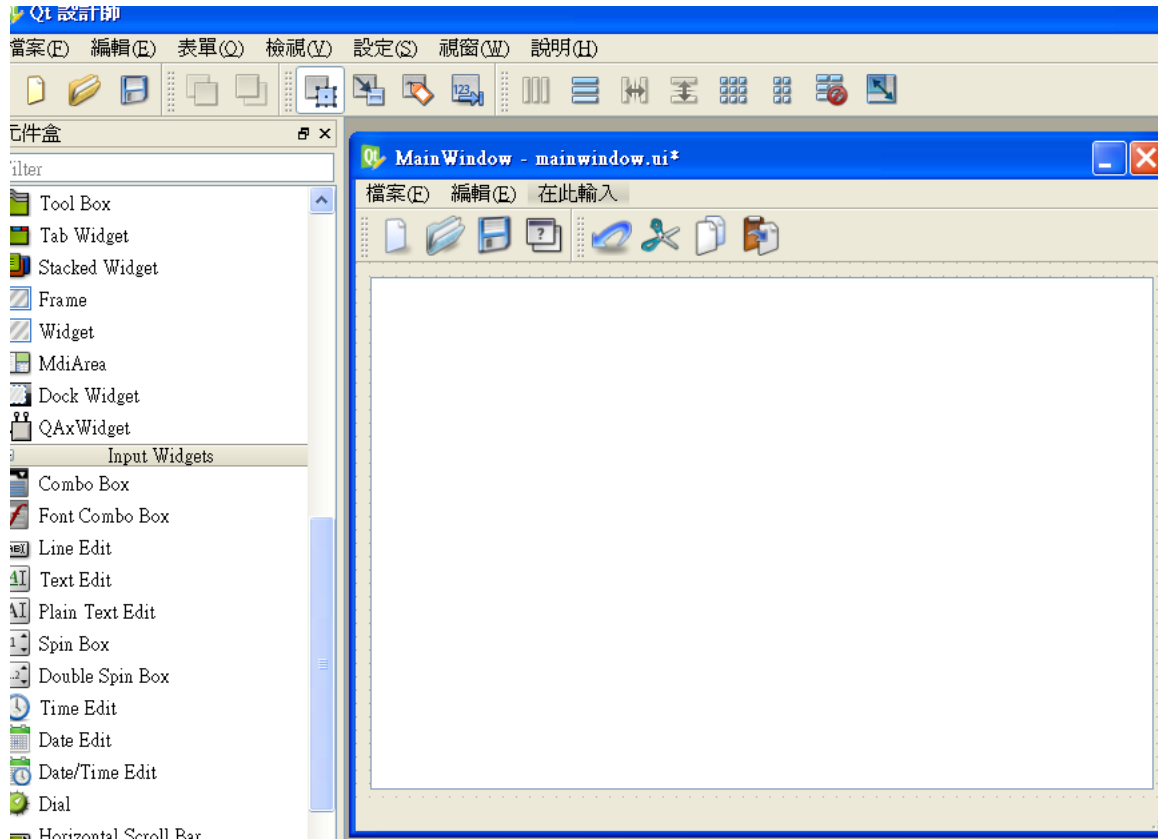
Qt- MainWindow

Qt Designer添加視窗中心元件(centralWidget)



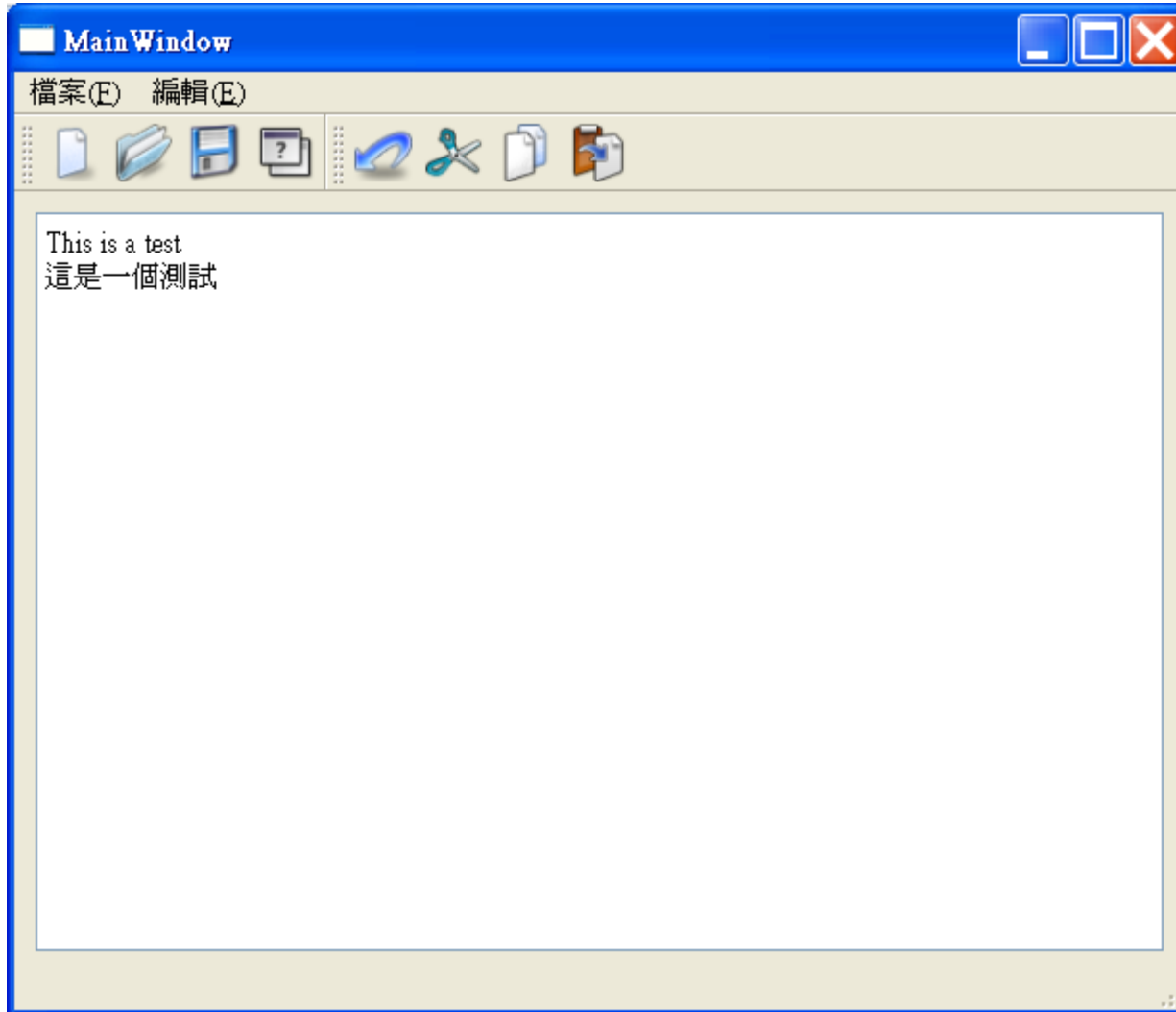
Qt- MainWindow

Qt Designer添加視窗中心元件(centralWidget)



Qt- MainWindow

Qt Designer存檔後；Qt Creator 建置、執行專案



Qt- MainWindow

Qt Creator 檢視 ui_mainwindow.h

