

Qt –Dialog-based Networking Application

Yih-Chuan Lin

CSIE Windows Programming Class

National Formosa University

Qt- Networking

- 學習目的
 - 認識Qt網路連線類別功能
 - 練習使用Qt TcpSocket類別
 - 學習如何整合網路連線功能於對話盒視窗應用程式



Qt- Networking

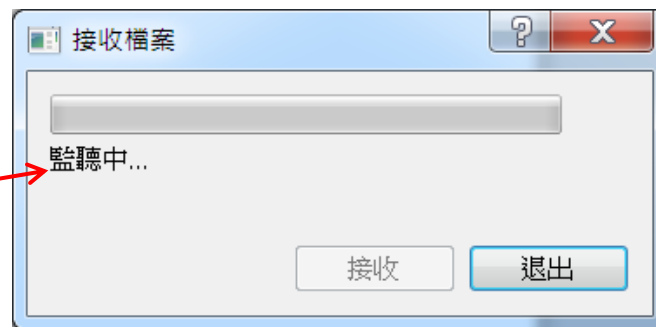
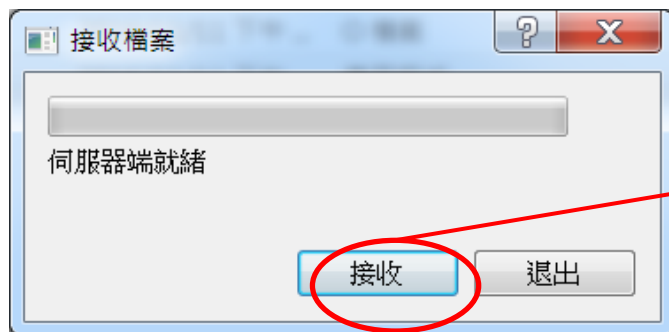
- **QTcpSocket Class is used for TCP communication end-point:**

Header:	<code>#include <QtNetwork></code>
qmake:	<code>QT += network</code>
Inherits:	<code>QAbstractSocket</code>
Inherited By:	<code>QSslSocket.</code>

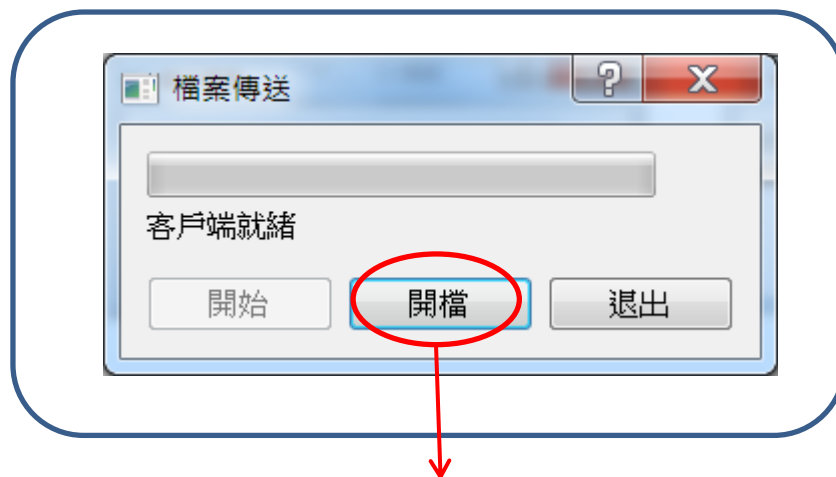


Qt- Networking

- **QTcpSocket Class : end-points of TCP connection**



Server

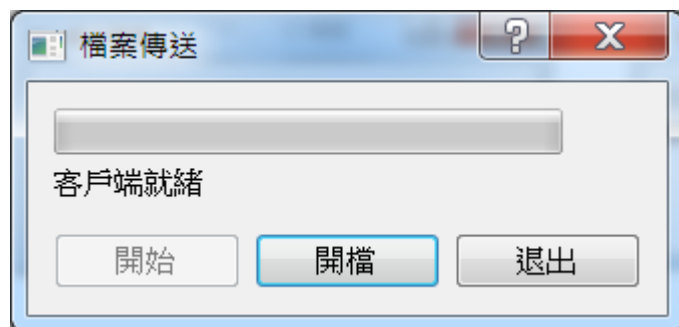


Client

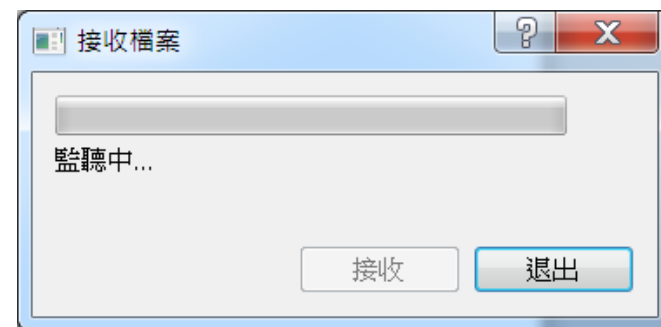
Qt- Networking

- The application data format (user-defined)

Client

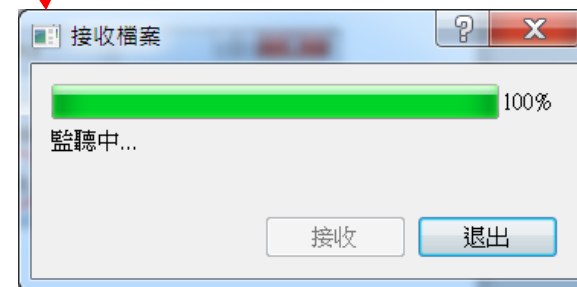
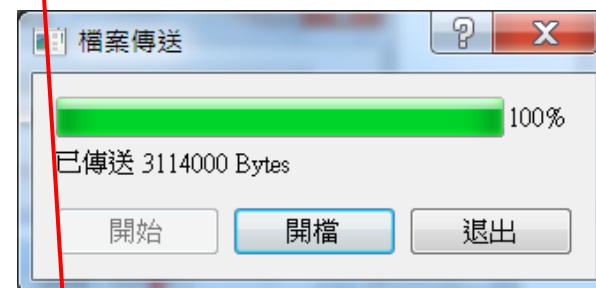
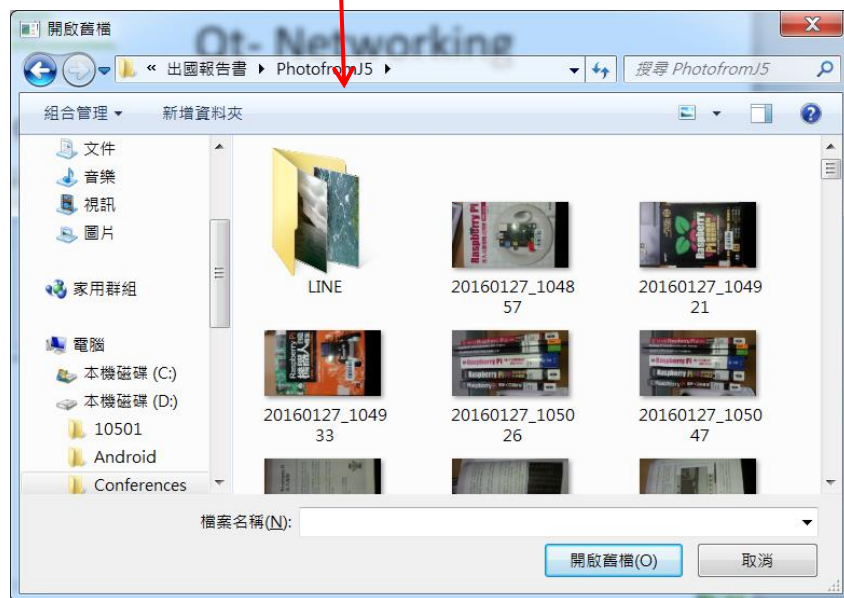
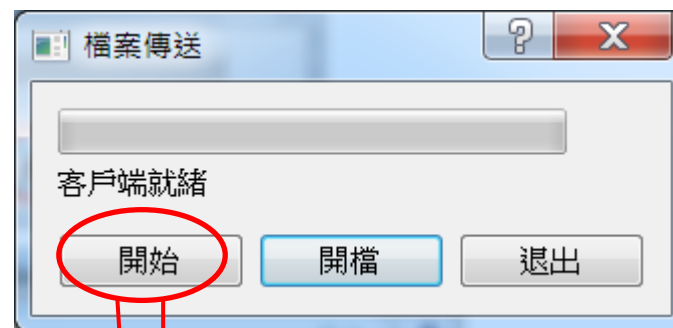
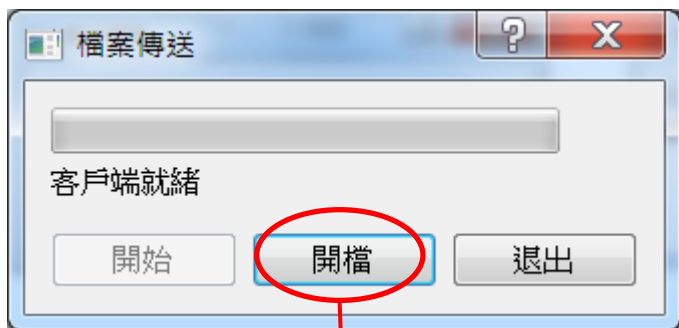


Server



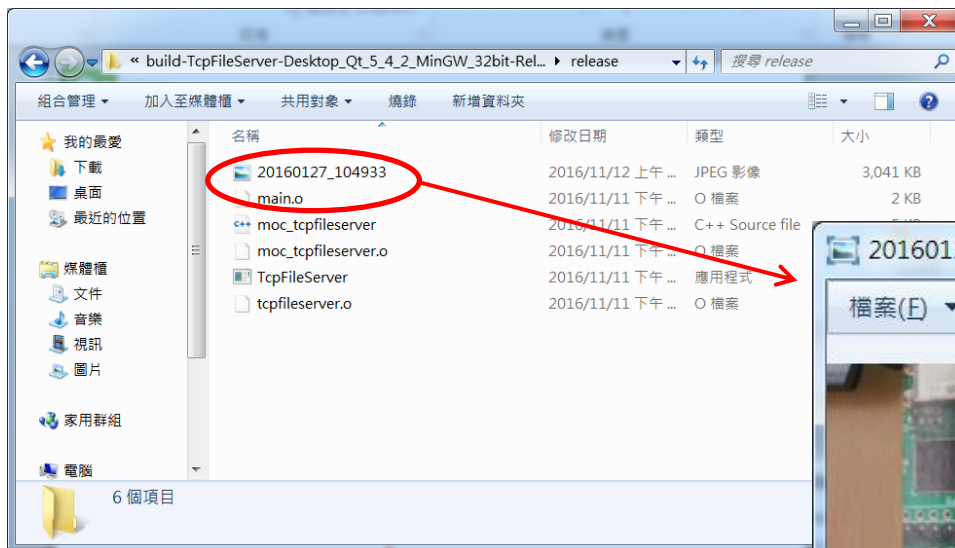
Qt- Networking

- **QTcpSocket Class : end-points of TCP connection**



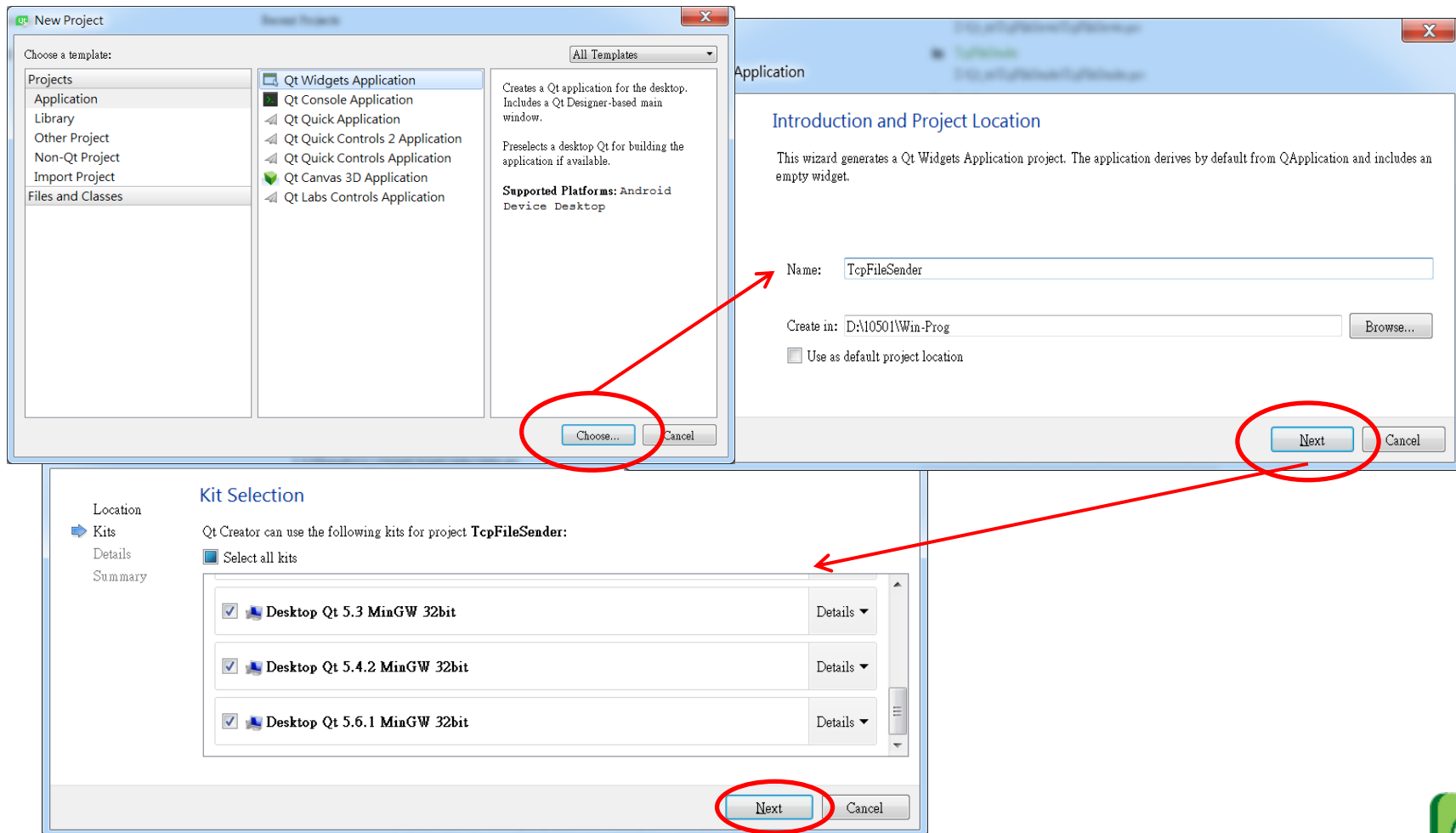
Qt- Networking

● QTcpSocket Class : end-points of TCP connection



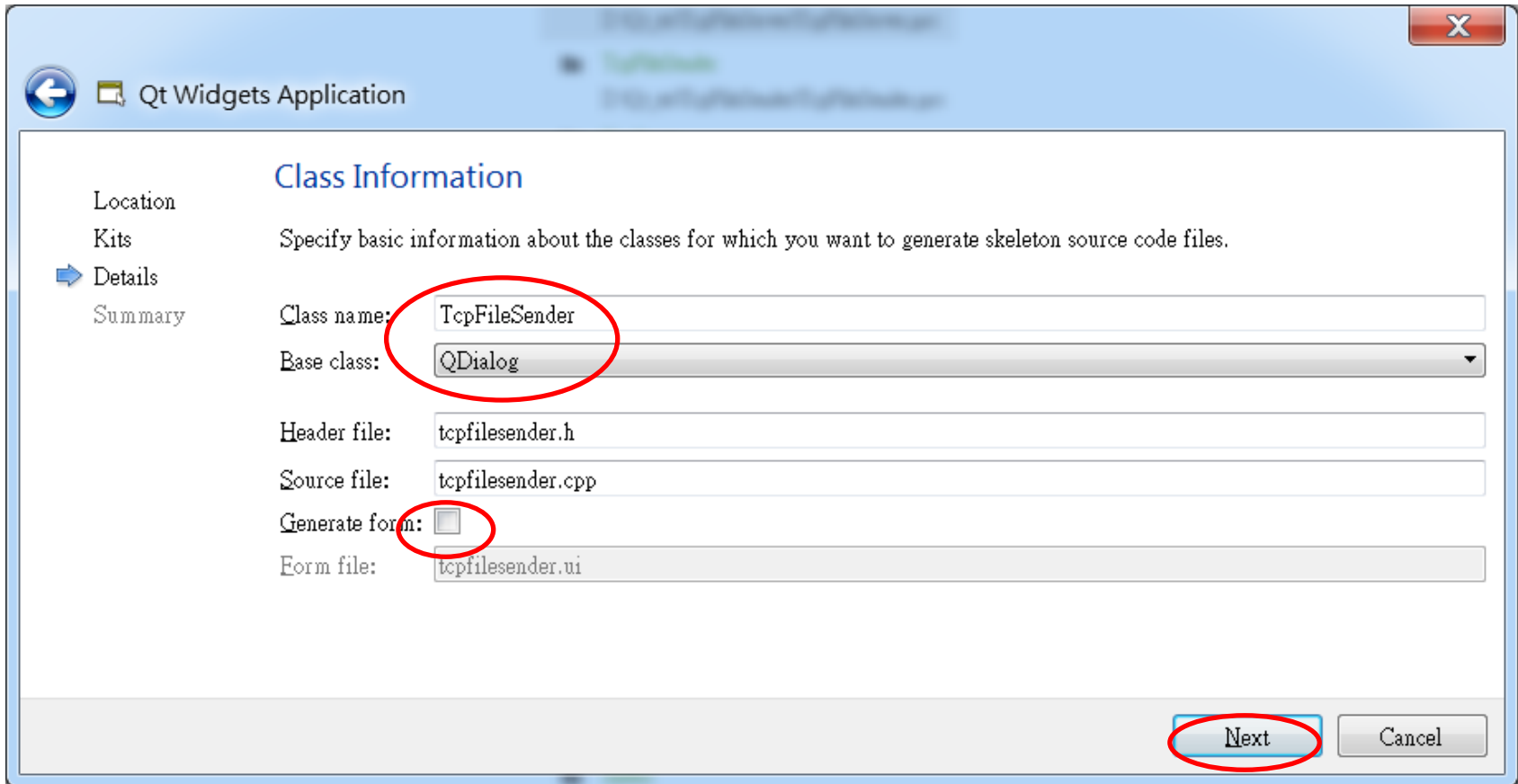
Qt- Networking

● Create a file sender dialog-based application:



Qt- Networking

- Create a file sender dialog-based application:



The image shows the 'Class Information' dialog in Qt Creator. The dialog has a sidebar on the left with 'Details' selected. The main area contains fields for class information. Red circles highlight the 'Class name' field (containing 'TcpFileSender'), the 'Base class' dropdown (showing 'QDialog'), the 'Generate form' checkbox (which is unchecked), and the 'Next' button at the bottom right.

Qt Widgets Application

Location
Kits
Details
Summary

Class Information

Specify basic information about the classes for which you want to generate skeleton source code files.

Class name:

Base class:

Header file:

Source file:

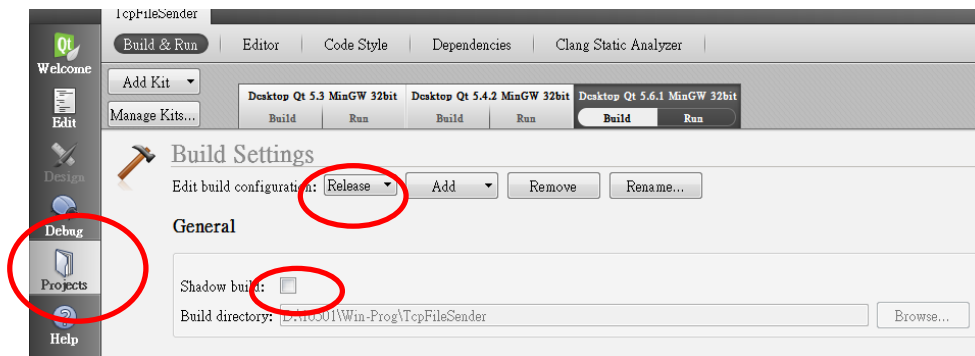
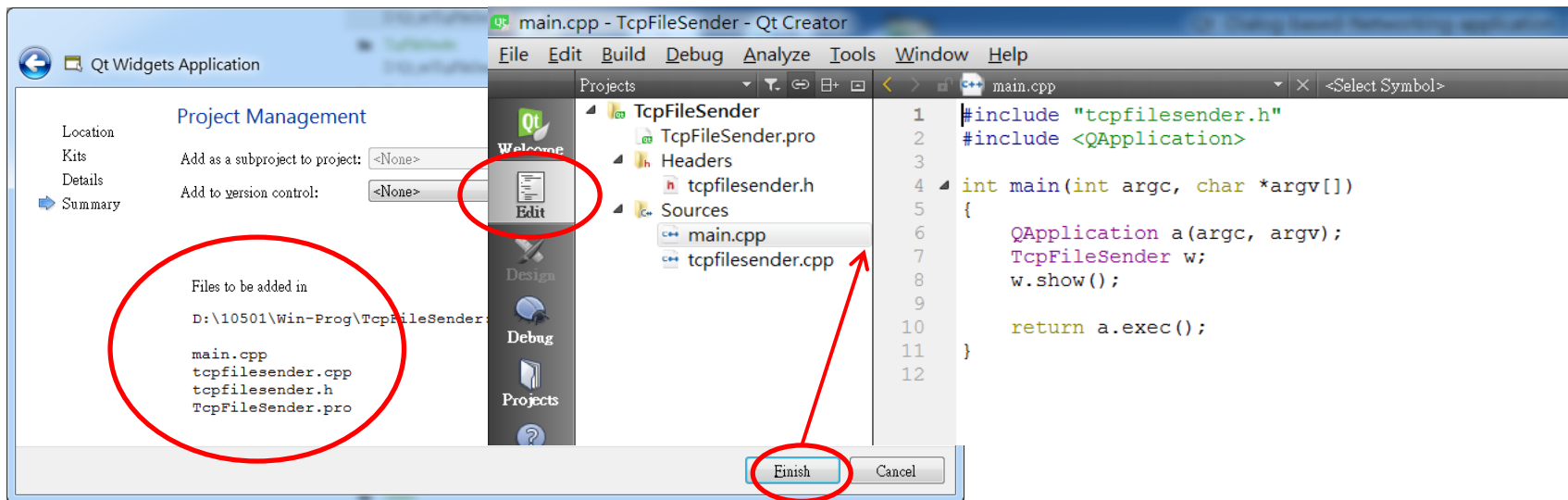
Generate form: ☐

Form file:

Next Cancel

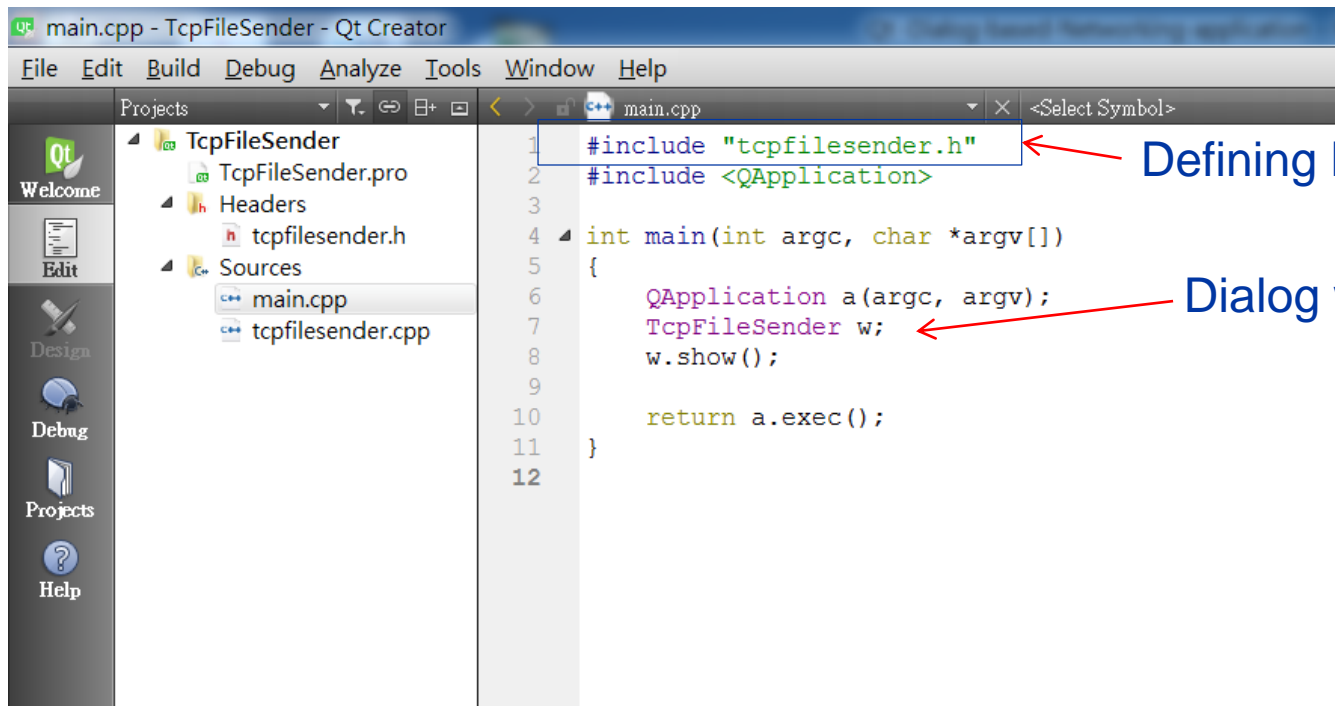
Qt- Networking

● Create a file sender dialog-based application:



Qt- Networking

- Check out the main function:



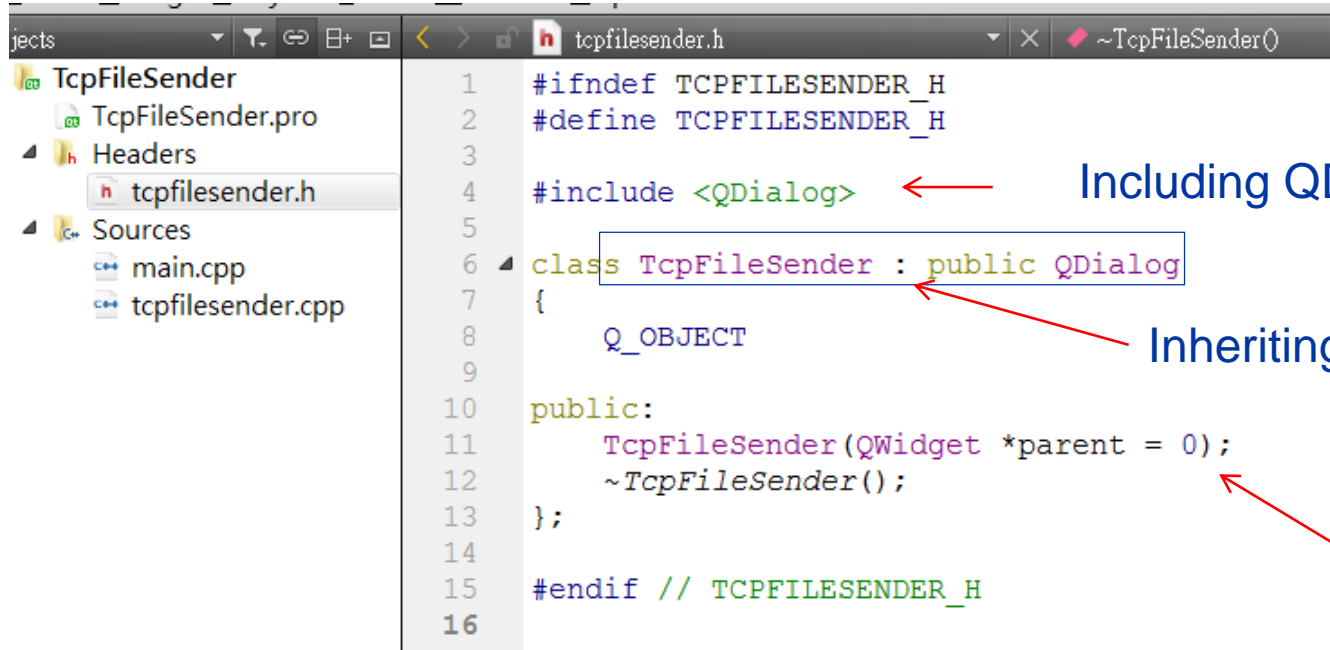
```
1 #include "tcpfilesender.h"
2 #include <QApplication>
3
4 int main(int argc, char *argv[])
5 {
6     QApplication a(argc, argv);
7     TcpFileSender w;
8     w.show();
9
10    return a.exec();
11 }
12
```

Defining Dialog window!

Dialog window object!

Qt-Networking

- Check out the definition of TcpFileSender class:



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

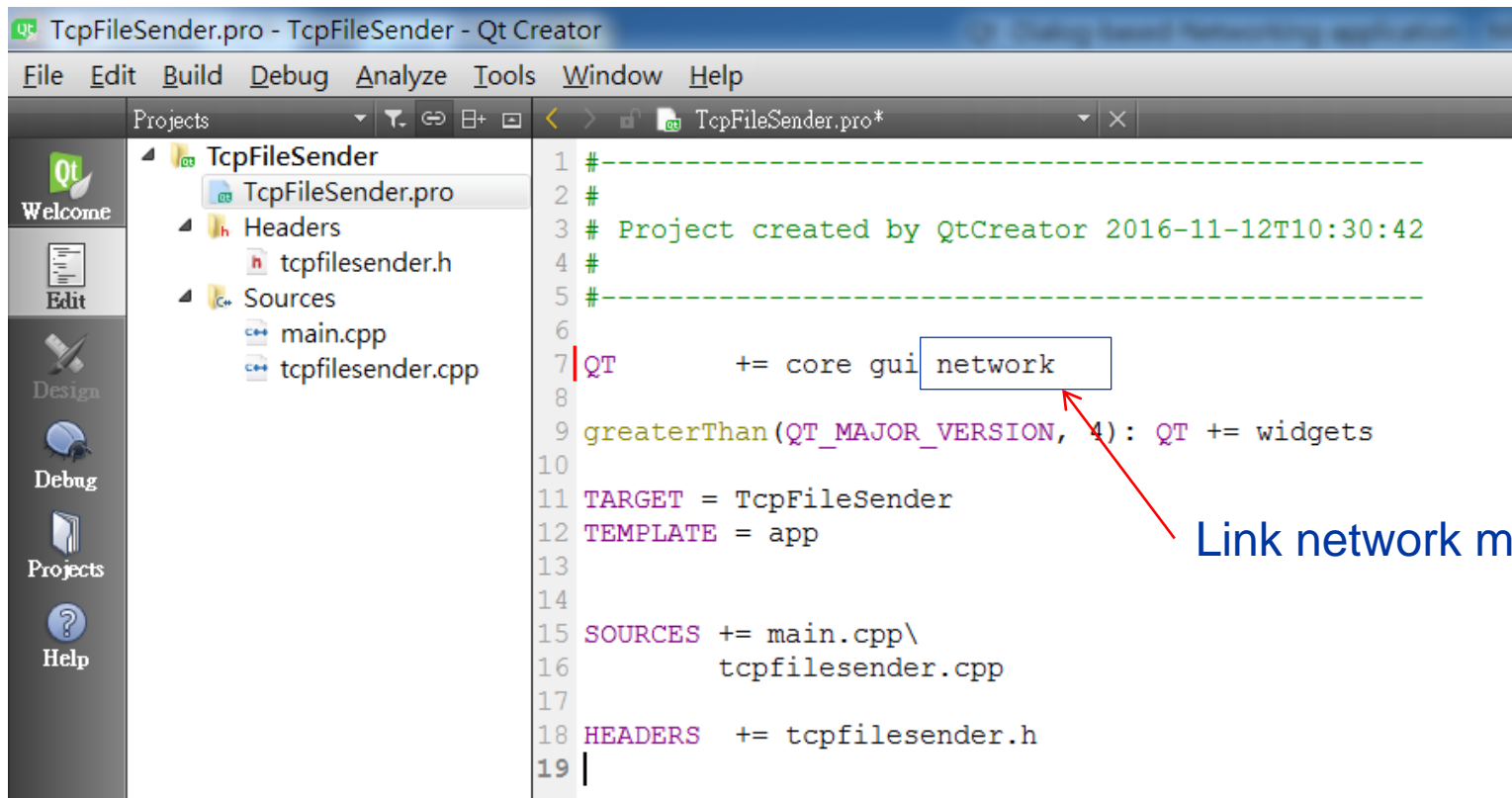
Including QDialog header file!

Inheriting from QDialog Class!

Constructor!

Networking

- Check out the qmake project file:



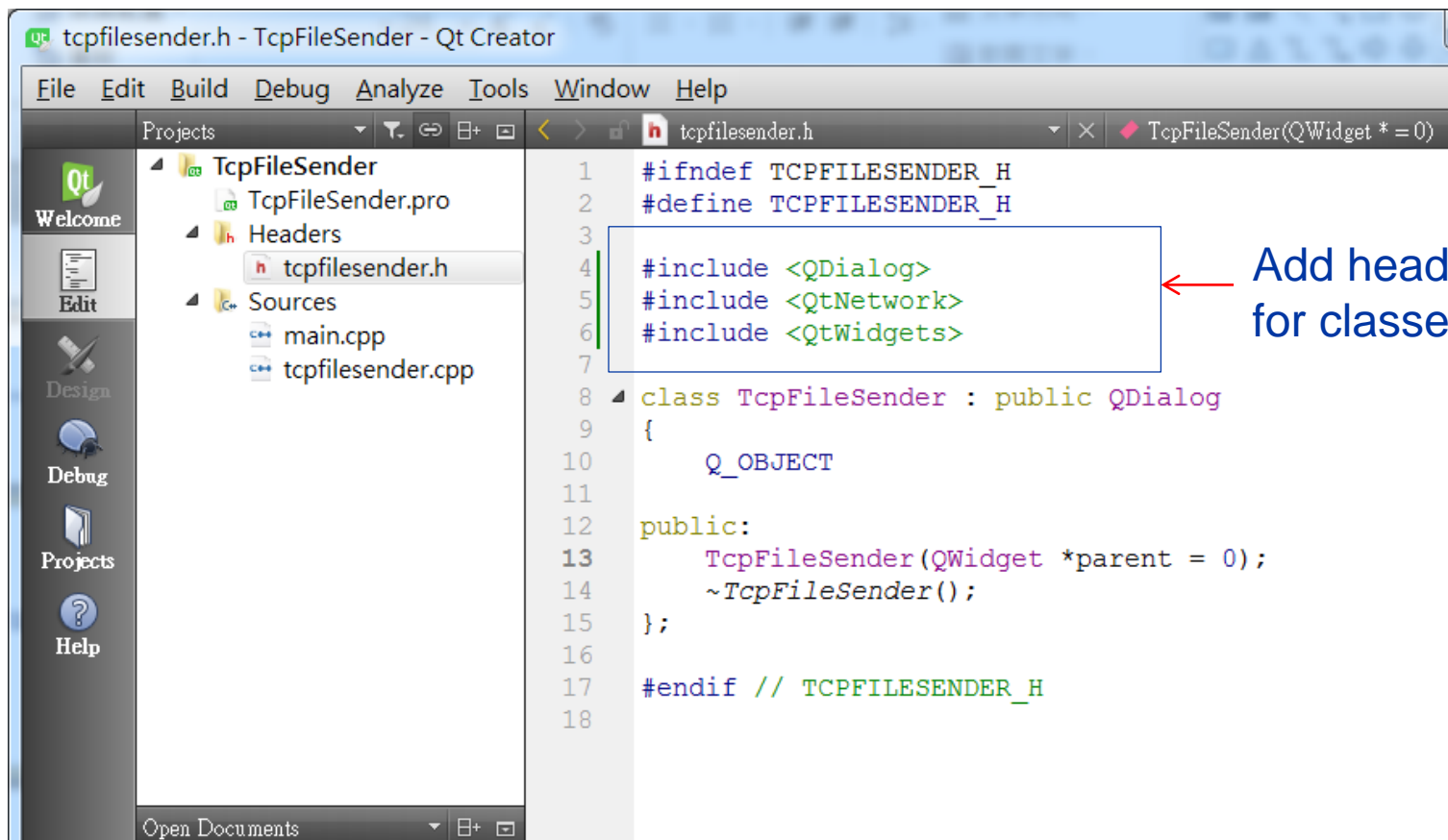
The screenshot shows the Qt Creator interface with the project file `TcpFileSender.pro` open. The left sidebar shows the project structure with `TcpFileSender` containing `TcpFileSender.pro`, `Headers` (with `tcpfilesender.h`), and `Sources` (with `main.cpp` and `tcpfilesender.cpp`). The main editor displays the content of `TcpFileSender.pro`, which is a qmake project file. A red arrow points to the line `QT += core gui network`, highlighting the `network` module being linked.

```
1 #-----
2 #
3 # Project created by QtCreator 2016-11-12T10:30:42
4 #
5 #-----
6
7 QT       += core gui network
8
9 greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
10
11 TARGET = TcpFileSender
12 TEMPLATE = app
13
14
15 SOURCES += main.cpp\
16          tcpfilesender.cpp
17
18 HEADERS  += tcpfilesender.h
19 |
```

Link network module!

Qt- Networking

- **Modify the definition of TcpFileSender class:**



```
1  #ifndef TCPFILESENDER_H
2  #define TCPFILESENDER_H
3
4  #include <QDialog>
5  #include <QtNetwork>
6  #include <QtWidgets>
7
8  class TcpFileSender : public QDialog
9  {
10     Q_OBJECT
11
12     public:
13         TcpFileSender(QWidget *parent = 0);
14         ~TcpFileSender();
15     };
16
17 #endif // TCPFILESENDER_H
18
```

← Add header files for classes used.

Qt- Networking

- **Modify the definition of TcpFileSender class:**

```

1  #ifndef TCPFILESENDER_H
2  #define TCPFILESENDER_H
3
4  #include <QDialog>
5  #include <QtNetwork>
6  #include <QtWidgets>
7
8  class TcpFileSender : public QDialog
9  {
10     Q_OBJECT
11
12     public:
13         TcpFileSender(QWidget *parent = 0);
14         ~TcpFileSender();
15
16     public slots:
17         void start();
18         void startTransfer();
19         void updateClientProgress(qint64 numBytes);
20         void openFile();
21     };

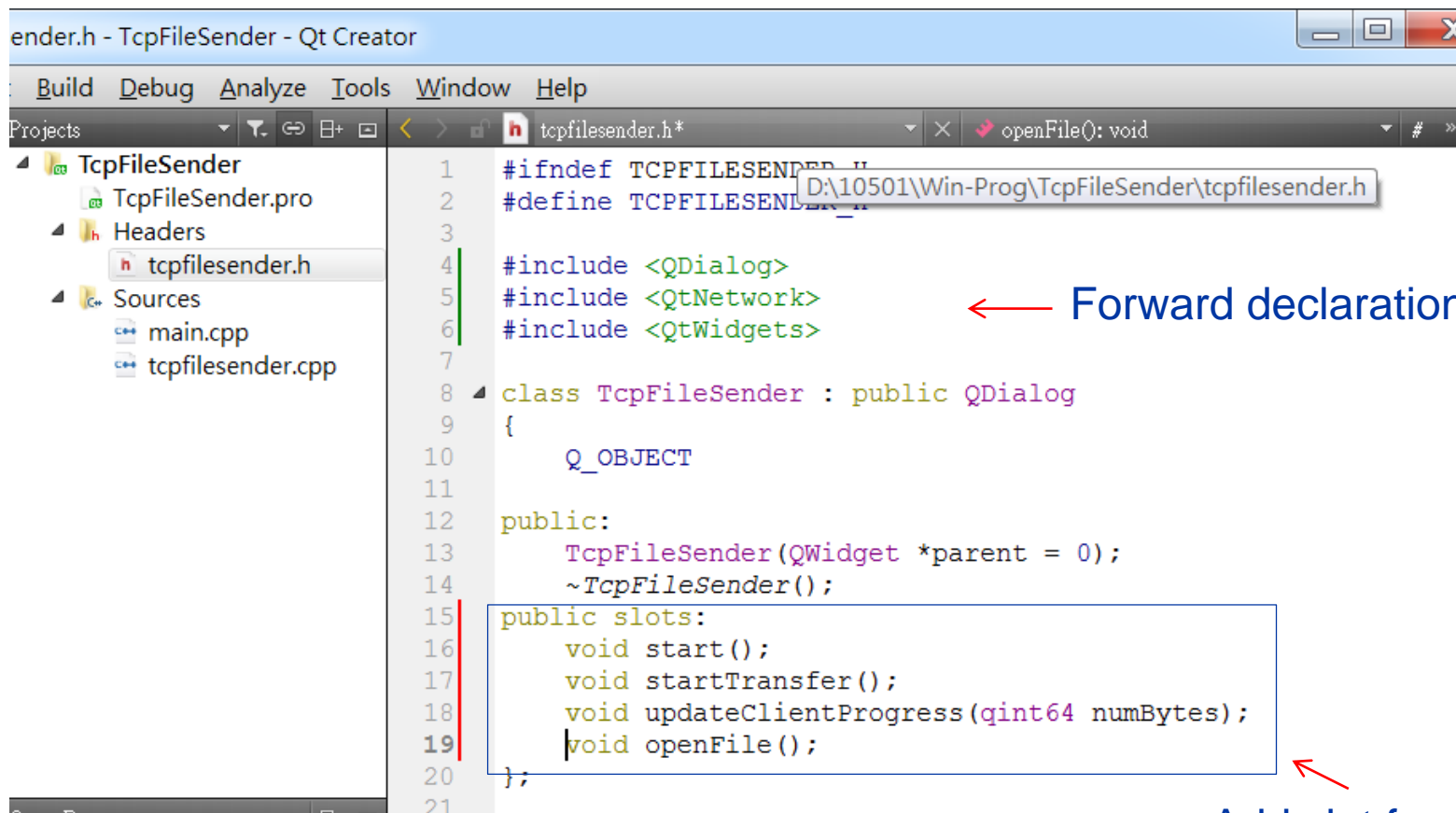
```

Forward declaration.

Add slot functions.

Qt- Networking

- **Modify the definition of TcpFileSender class:**



```

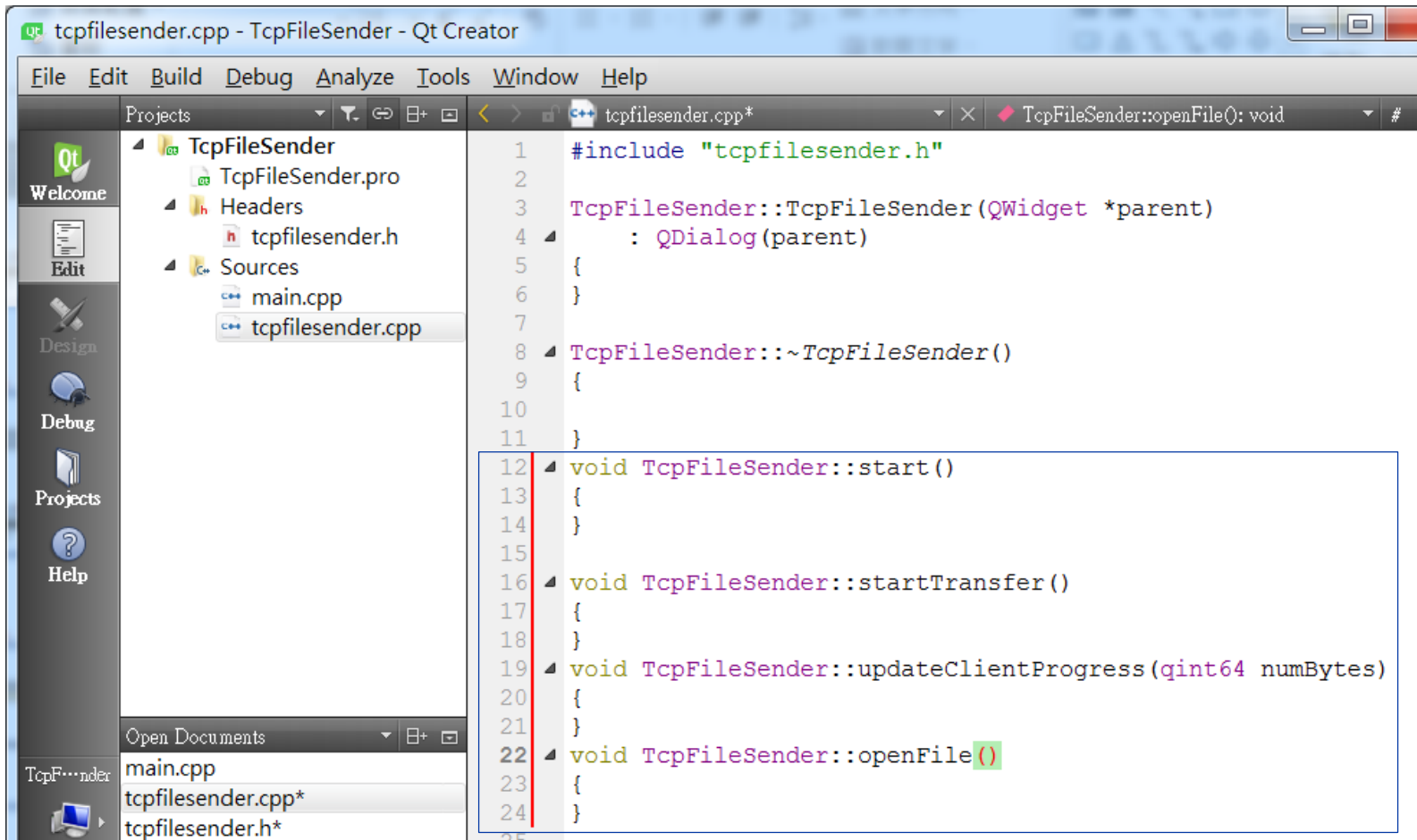
1  #ifndef TCPFILESENDER_H
2  #define TCPFILESENDER_H
3
4  #include <QDialog>
5  #include <QtNetwork>
6  #include <QtWidgets>
7
8  class TcpFileSender : public QDialog
9  {
10     Q_OBJECT
11
12     public:
13         TcpFileSender(QWidget *parent = 0);
14         ~TcpFileSender();
15
16     public slots:
17         void start();
18         void startTransfer();
19         void updateClientProgress(qint64 numBytes);
20         void openFile();
21     };

```

Add slot functions.

Qt- Networking

- Add slot function implementation to tcpfilesender.cpp:



The screenshot shows the Qt Creator IDE with the project 'TcpFileSender' open. The left sidebar displays the project structure, including 'Headers' and 'Sources' folders. The 'Sources' folder contains 'main.cpp' and 'tcpfilesender.cpp'. The 'tcpfilesender.cpp' file is selected, and its code is displayed in the main editor. The code includes the 'tcpfilesender.h' header and implements several functions: 'TcpFileSender::TcpFileSender(QWidget *parent)', 'TcpFileSender::~~TcpFileSender()', 'void TcpFileSender::start()', 'void TcpFileSender::startTransfer()', 'void TcpFileSender::updateClientProgress(qint64 numBytes)', and 'void TcpFileSender::openFile()'. The 'openFile()' function is highlighted with a red vertical line, indicating it is the focus of the current task.

```
1  #include "tcpfilesender.h"
2
3  TcpFileSender::TcpFileSender(QWidget *parent)
4      : QDialog(parent)
5  {
6  }
7
8  TcpFileSender::~~TcpFileSender()
9  {
10 }
11
12 void TcpFileSender::start()
13 {
14 }
15
16 void TcpFileSender::startTransfer()
17 {
18 }
19 void TcpFileSender::updateClientProgress(qint64 numBytes)
20 {
21 }
22 void TcpFileSender::openFile()
23 {
24 }
```

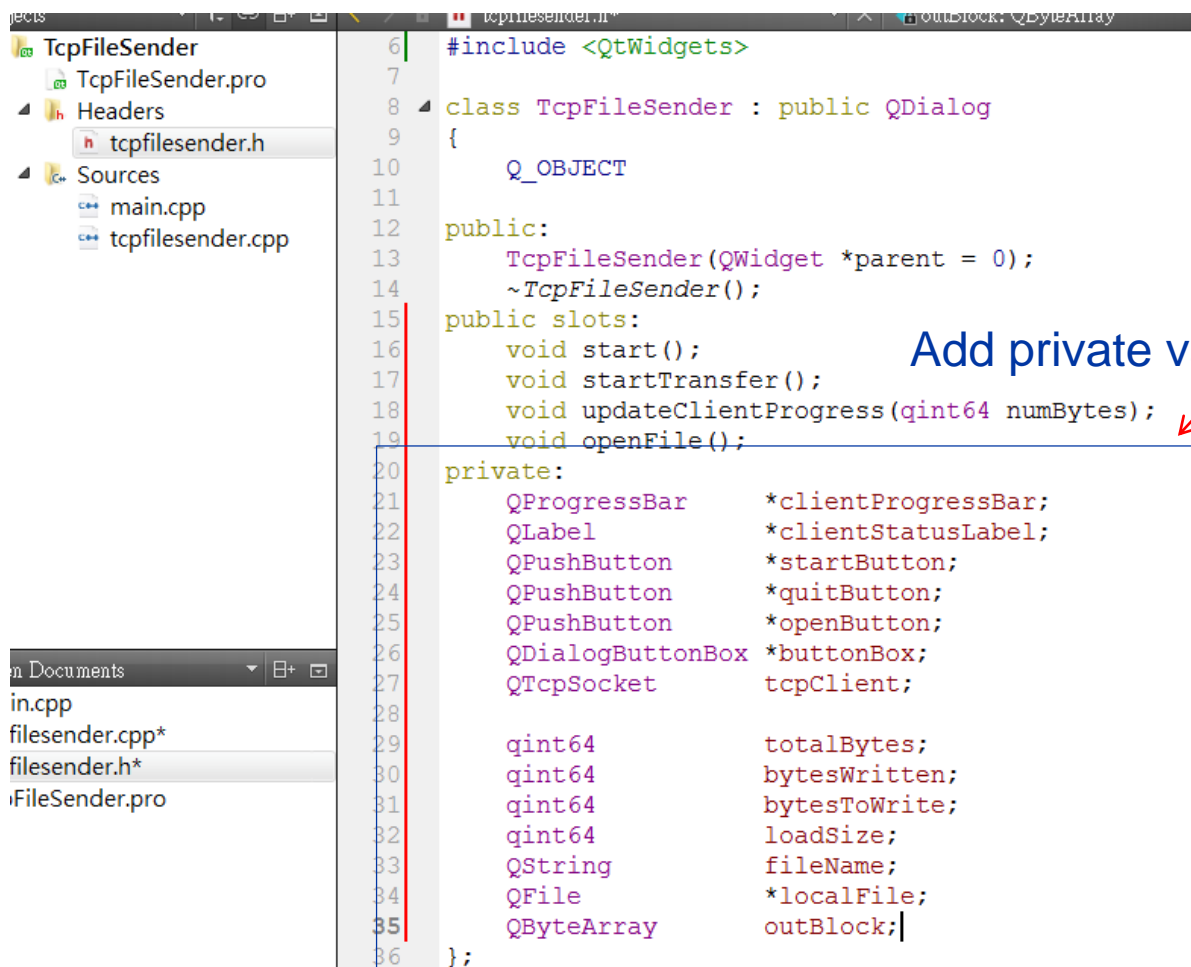
Add slot function
implementations.



Code less.
Create more.
Deploy everywhere.

Qt- Networking

- Modify the definition of TcpFileSender class:



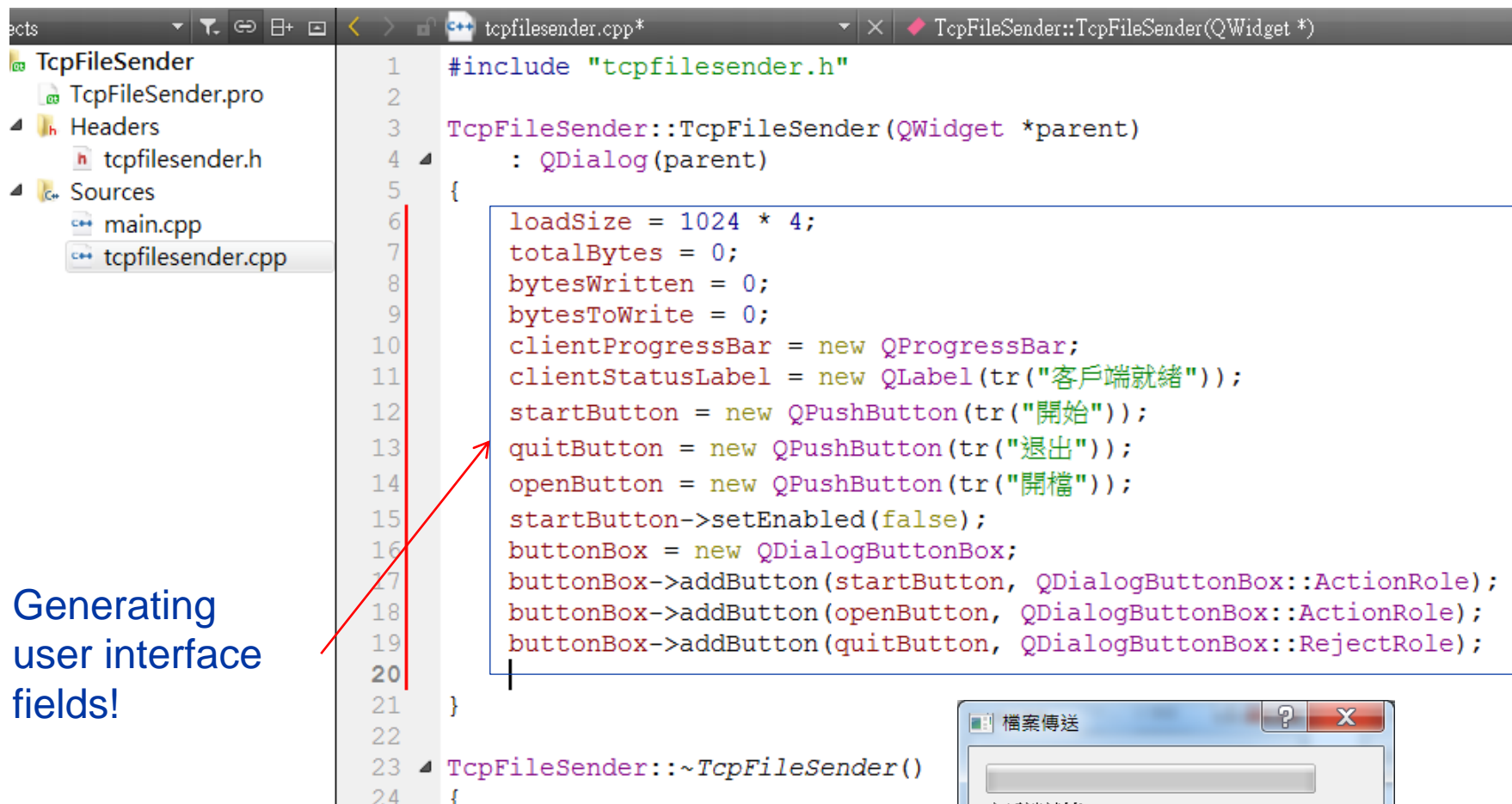
```
1 #include <QtWidgets>
2
3 class TcpFileSender : public QDialog
4 {
5     Q_OBJECT
6
7 public:
8     TcpFileSender(QWidget *parent = 0);
9     ~TcpFileSender();
10
11 public slots:
12     void start();
13     void startTransfer();
14     void updateClientProgress(qint64 numBytes);
15     void openFile();
16
17 private:
18     QProgressBar *clientProgressBar;
19     QLabel *clientStatusLabel;
20     QPushButton *startButton;
21     QPushButton *quitButton;
22     QPushButton *openButton;
23     QDialogButtonBox *buttonBox;
24     QTcpSocket tcpClient;
25
26     qint64 totalBytes;
27     qint64 bytesWritten;
28     qint64 bytesToWrite;
29     qint64 loadSize;
30     QString fileName;
31     QFile *localFile;
32     QByteArray outBlock;
```

Add private variables.



Qt- Networking

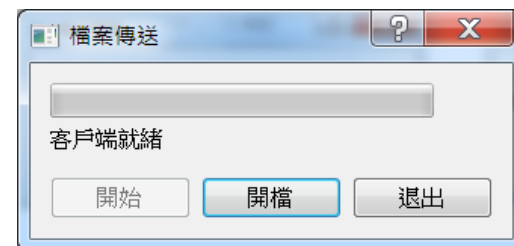
- **Modify the constructor of TcpFileSender class:**



```

1  #include "tcpfilesender.h"
2
3  TcpFileSender::TcpFileSender(QWidget *parent)
4      : QDialog(parent)
5  {
6      loadSize = 1024 * 4;
7      totalBytes = 0;
8      bytesWritten = 0;
9      bytesToWrite = 0;
10     clientProgressBar = new QProgressBar;
11     clientStatusLabel = new QLabel(tr("客戶端就緒"));
12     startButton = new QPushButton(tr("開始"));
13     quitButton = new QPushButton(tr("退出"));
14     openButton = new QPushButton(tr("開檔"));
15     startButton->setEnabled(false);
16     buttonBox = new QDialogButtonBox;
17     buttonBox->addButton(startButton, QDialogButtonBox::ActionRole);
18     buttonBox->addButton(openButton, QDialogButtonBox::ActionRole);
19     buttonBox->addButton(quitButton, QDialogButtonBox::RejectRole);
20 }
21
22
23 TcpFileSender::~TcpFileSender()
24 {
  
```

Generating user interface fields!



Qt- Networking

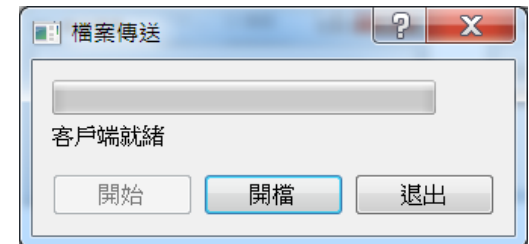
● Modify the constructor of TcpFileSender class:

Establish layout managers!

```

1  #include "tcpfilesender.h"
2
3  TcpFileSender::TcpFileSender(QWidget *parent)
4      : QDialog(parent)
5  {
6      loadSize = 1024 * 4;
7      totalBytes = 0;
8      bytesWritten = 0;
9      bytesToWrite = 0;
10     clientProgressBar = new QProgressBar;
11     clientStatusLabel = new QLabel(tr("客戶端就緒"));
12     startButton = new QPushButton(tr("開始"));
13     quitButton = new QPushButton(tr("退出"));
14     openButton = new QPushButton(tr("開檔"));
15     startButton->setEnabled(false);
16     buttonBox = new QDialogButtonBox;
17     buttonBox->addButton(startButton, QDialogButtonBox::ActionRole);
18     buttonBox->addButton(openButton, QDialogButtonBox::ActionRole);
19     buttonBox->addButton(quitButton, QDialogButtonBox::RejectRole);
20
21     QVBoxLayout *mainLayout = new QVBoxLayout;
22     mainLayout->addWidget(clientProgressBar);
23     mainLayout->addWidget(clientStatusLabel);
24     mainLayout->addStretch(1);
25     mainLayout->addSpacing(10);
26     mainLayout->addWidget(buttonBox);
27     setLayout(mainLayout);
28     setWindowTitle(tr("檔案傳送"));
29
30 }
31

```



Qt- Networking

- **Modify the constructor of TcpFileSender class:**

```

19     buttonBox->addButton(quitButton, QDialogButtonBox::RejectRole);
20
21     QVBoxLayout *mainLayout = new QVBoxLayout;
22     mainLayout->addWidget(clientProgressBar);
23     mainLayout->addWidget(clientStatusLabel);
24     mainLayout->addStretch(1);
25     mainLayout->addSpacing(10);
26     mainLayout->addWidget(buttonBox);
27     setLayout(mainLayout);
28     setWindowTitle(tr("檔案傳送"));
29
30     connect(openButton, SIGNAL(clicked()), this, SLOT(openFile()));
31     connect(startButton, SIGNAL(clicked()), this, SLOT(start()));
32     connect(&tcpClient, SIGNAL(connected()), this, SLOT(startTransfer()));
33     connect(&tcpClient, SIGNAL(bytesWritten(qint64)), this, SLOT(updateClientProgress(qint64)));
34     connect(quitButton, SIGNAL(clicked()), this, SLOT(close()));
35 }
36
37 TcpFileSender::~TcpFileSender()
38 {
39
40 }
41
42 void TcpFileSender::start()
43 {

```

Connect signals to slot functions!

Qt- Networking


- **Build and run the project:**



Try to interact with the dialog window displayed!
Do you see any reaction when you click on the buttons?

Qt- Networking

● Implement slot functions openFile() :



```

22     mainLayout->addWidget(clientProgressBar);
23     mainLayout->addWidget(clientStatusLabel);
24     mainLayout->addStretch(1);
25     mainLayout->addSpacing(10);
26     mainLayout->addWidget(buttonBox);
27     setLayout(mainLayout);
28     setWindowTitle(tr("檔案傳送"));
29     connect(openButton, SIGNAL(clicked()), this, SLOT(openFile()));
30     connect(startButton, SIGNAL(clicked()), this, SLOT(start()));
31     connect(&tcpClient, SIGNAL(connected()), this, SLOT(startTransfer()));
32     connect(&tcpClient, SIGNAL(bytesWritten(qint64)), this, SLOT(updateClientPr
33     connect(quitButton, SIGNAL(clicked()), this, SLOT(close()));
34
35 }
36 void TcpFileSender::openFile()
37 {
38     fileName = QFileDialog::getOpenFileName(this);
39     if (!fileName.isEmpty()) startButton->setEnabled(true);
40 }
41 TcpFileSender::~TcpFileSender()
42 {
43 }
44
45 void TcpFileSender::start()

```

Get the filename the user selects from the QFileDialog.

Check if the filename is gotten successfully.

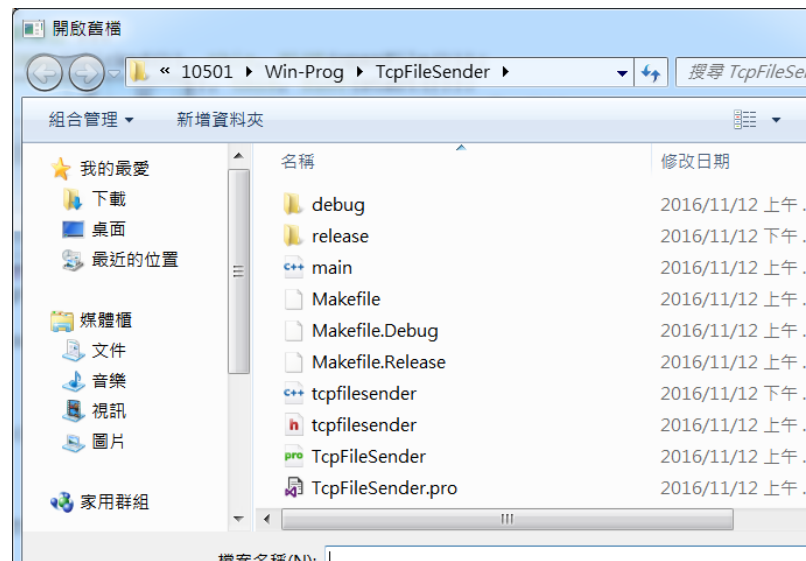
If yes, set the start button enabled!

Qt- Networking

- Build and run the project:



Try to interact with the “開檔” button!



Qt- Networking

● Implement slot function start():



```

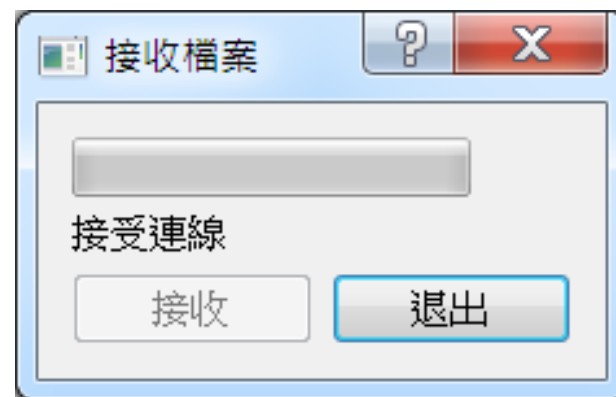
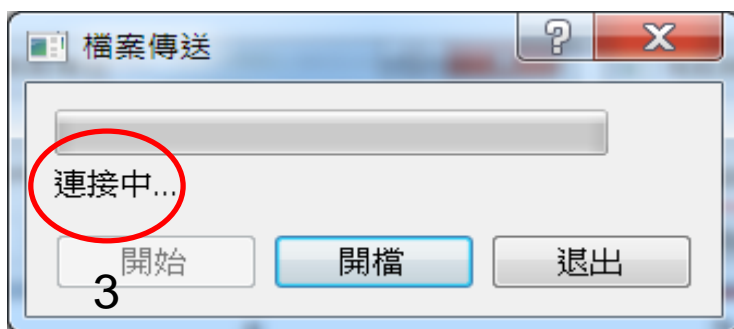
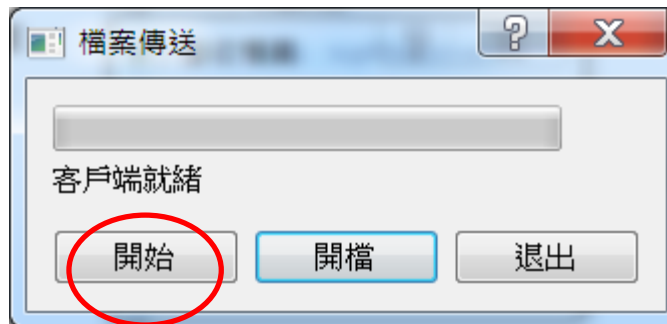
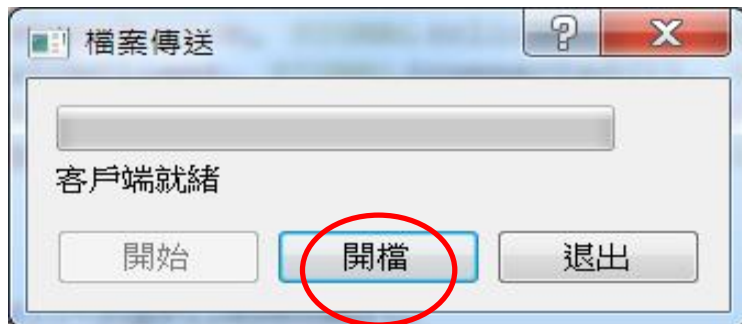
23   mainLayout->addWidget(clientStatusLabel);
24   mainLayout->addStretch(1);
25   mainLayout->addSpacing(10);
26   mainLayout->addWidget(buttonBox);
27   setLayout(mainLayout);
28   setWindowTitle(tr("檔案傳送"));
29   connect(openButton, SIGNAL(clicked()), this, SLOT(openFile()));
30   connect(startButton, SIGNAL(clicked()), this, SLOT(start()));
31   connect(&tcpClient, SIGNAL(connected()), this, SLOT(startTransfer()));
32   connect(&tcpClient, SIGNAL(bytesWritten(qint64)), this, SLOT(updateClientProgress()));
33   connect(quitButton, SIGNAL(clicked()), this, SLOT(close()));
34
35 }
36 void TcpFileSender::openFile()
37 {
38     fileName = QFileDialog::getOpenFileName(this);
39     if (!fileName.isEmpty()) startButton->setEnabled(true);
40 }
41 void TcpFileSender::start()
42 {
43     startButton->setEnabled(false);
44     bytesWritten = 0;
45     clientStatusLabel->setText(tr("連接中..."));
46     tcpClient.connectToHost(QHostAddress::LocalHost, 16689);
47 }
48

```

Use connectToHost() to initiate a tcp connection to a host with ip (127.0.0.1) running a server listen on port (16689).

Qt- Networking

- Build and run the project:



Server has accepted the connection.



Code less.
Create more.
Deploy everywhere.

Qt- Networking

- Implement slot function startTransfer():

```
er.cpp - TcpFileSender - Qt Creator
Build Debug Analyze Tools Window Help
tcpfilesender.cpp*
TcpFileSender
  TcpFileSender.pro
  Headers
    tcpfilesender.h
  Sources
    main.cpp
    tcpfilesender.cpp
47 }
48 void TcpFileSender::startTransfer()
49 {
50     localFile = new QFile(fileName);
51     if (!localFile->open(QFile::ReadOnly))
52     {
53         QMessageBox::warning(this, tr("應用程式"),
54                               tr("無法讀取 %1:\n%2.") .arg(fileName)
55                               .arg(localFile->errorString()));
56         return;
57     }
58
59
60
61
```

Step 1: Open the file the user has selected.



Qt- Networking

- Implement slot function startTransfer():

```


Sender.pro
rs
fileSender.h
as
in.cpp
fileSender.cpp
48 void TcpFileSender::startTransfer()
49 {
50     localFile = new QFile(fileName);
51     if (!localFile->open(QFile::ReadOnly))
52     {
53         QMessageBox::warning(this, tr("應用程式"),
54                               tr("無法讀取 %1:\n%2.").arg(fileName)
55                               .arg(localFile->errorString()));
56         return;
57     }
58
59     totalBytes = localFile->size();
60     QDataStream sendOut(&outBlock, QIODevice::WriteOnly);
61     sendOut.setVersion(QDataStream::Qt_4_6);
62     QString currentFile = fileName.right(fileName.size() -
63                                         fileName.lastIndexOf("/") - 1);
64     sendOut << qint64(0) << qint64(0) << currentFile;
65     totalBytes += outBlock.size();
66

```

Step 2: Fill the filename field in the header.

Qt- Networking

- Implement slot function startTransfer():



```

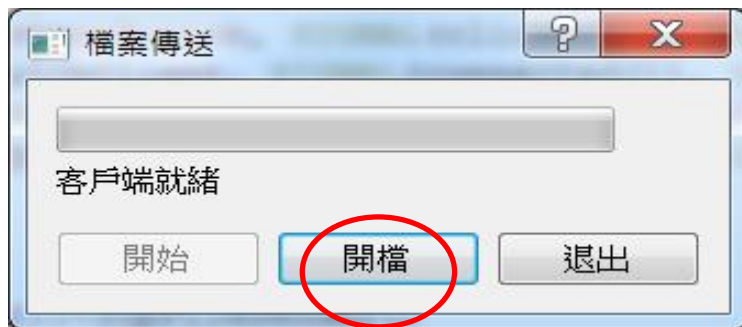
57     }
58
59     totalBytes = localFile->size();
60     QDataStream sendOut(&outBlock, QIODevice::WriteOnly);
61     sendOut.setVersion(QDataStream::Qt_4_6);
62     QString currentFile = fileName.right(fileName.size() -
63                                     fileName.lastIndexOf("/")-1);
64     sendOut << qint64(0) << qint64(0) << currentFile;
65     totalBytes += outBlock.size();
66
67     sendOut.device()->seek(0);
68     sendOut << totalBytes << qint64((outBlock.size() - sizeof(qint64)*2));
69     bytesToWrite = totalBytes - tcpClient.write(outBlock);
70     clientStatusLabel->setText(tr("已連接"));
71     qDebug() << currentFile << totalBytes;
72     outBlock.resize(0);
73 }

```

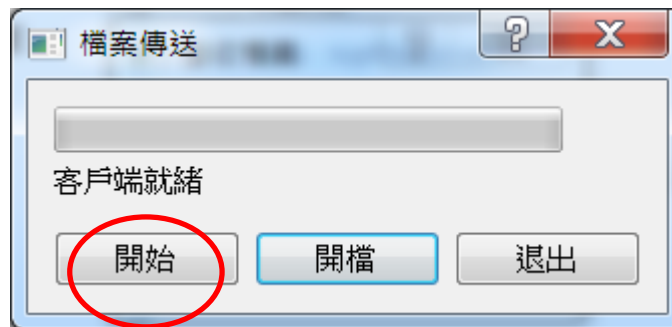
Step 3: Fill the **total size** and **filename size** fields in the header.
And send the header first using write() method.

Qt- Networking

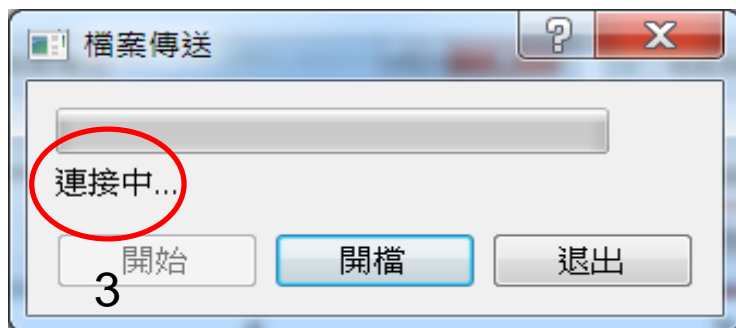
- **Build and run the project:**



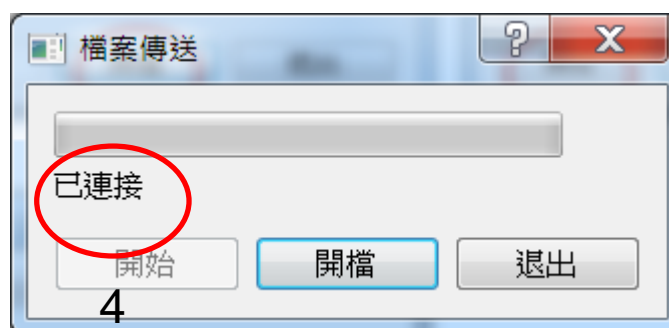
1



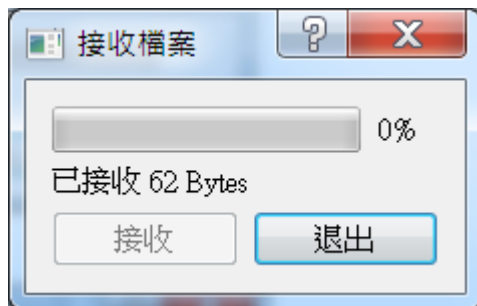
2



3



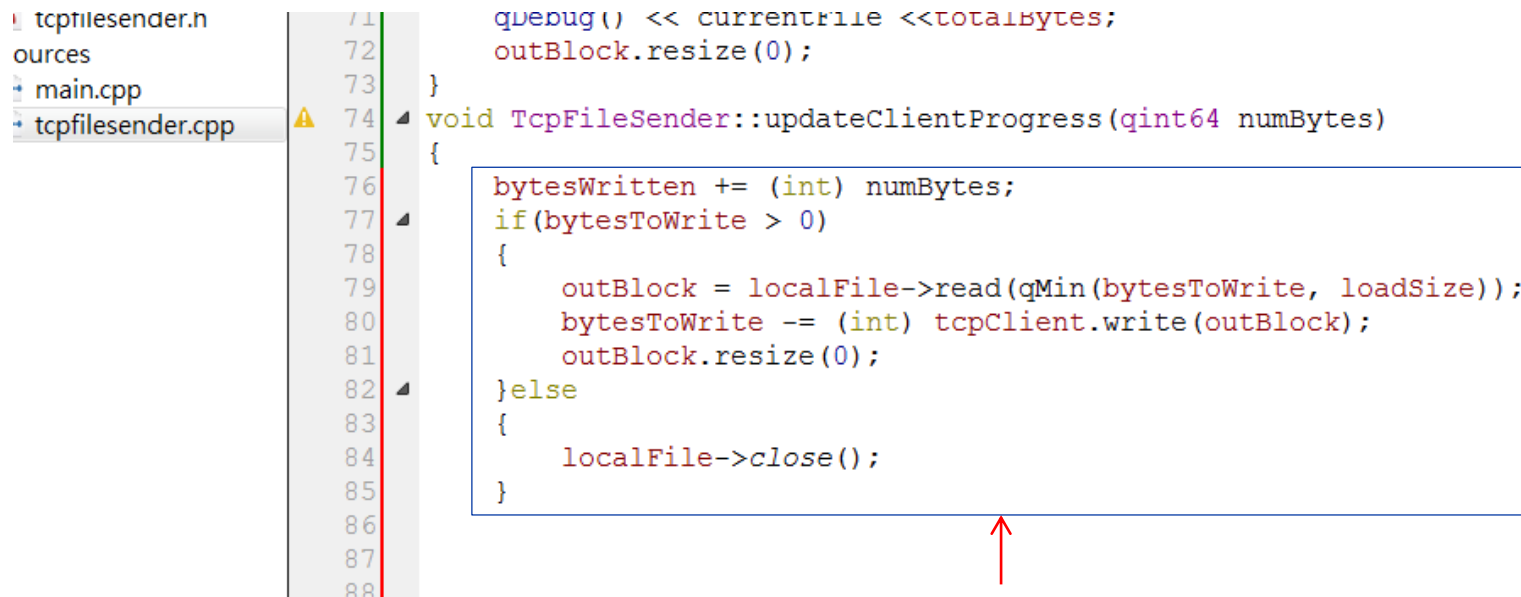
4



Server has received the header part.

Qt- Networking

- Implement slot function `updateClientProgress()`:



```
71  qDebug() << currentFile << totalBytes;
72  outBlock.resize(0);
73  }
74  void TcpFileSender::updateClientProgress(qint64 numBytes)
75  {
76      bytesWritten += (int) numBytes;
77      if(bytesToWrite > 0)
78      {
79          outBlock = localFile->read(qMin(bytesToWrite, loadSize));
80          bytesToWrite -= (int) tcpClient.write(outBlock);
81          outBlock.resize(0);
82      }else
83      {
84          localFile->close();
85      }
86
87
88
```

Step 1: Read payload (at most 4 KB) from the file.
Send the payload (in outBlock) using write().

Qt- Networking

- Implement slot function `startTransfer()`:

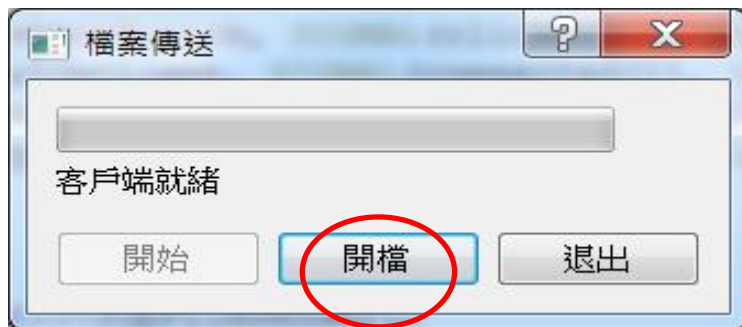


```
72     outBlock.resize(0);
73 }
74 void TcpFileSender::updateClientProgress(qint64 numBytes)
75 {
76     bytesWritten += (int) numBytes;
77     if(bytesToWrite > 0)
78     {
79         outBlock = localFile->read(qMin(bytesToWrite, loadSize));
80         bytesToWrite -= (int) tcpClient.write(outBlock);
81         outBlock.resize(0);
82     }else
83     {
84         localFile->close();
85     }
86
87     clientProgressBar->setMaximum(totalBytes);
88     clientProgressBar->setValue(bytesWritten);
89     clientStatusLabel->setText(tr("已傳送 %1 Bytes").arg(bytesWritten));
90 }
```

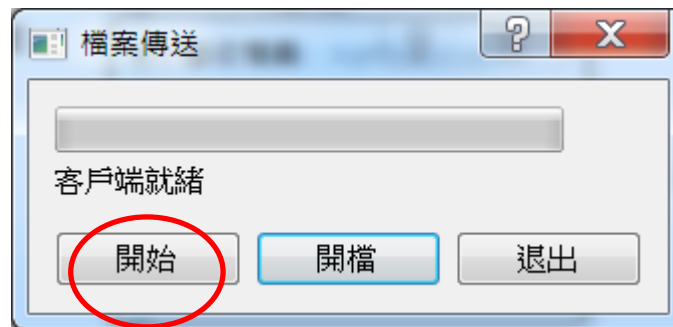
Step 2: Update the progress bar and the status label shown in the window.

Qt- Networking

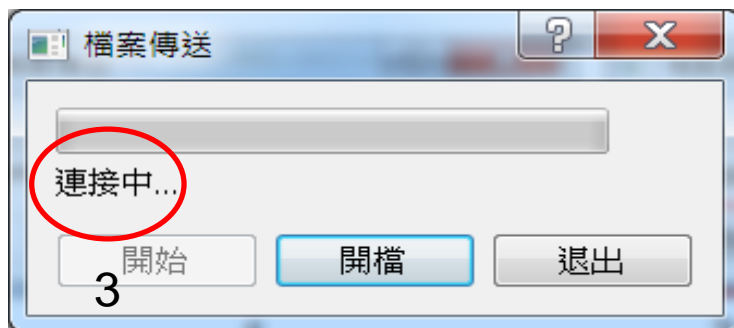
- Build and run the project:



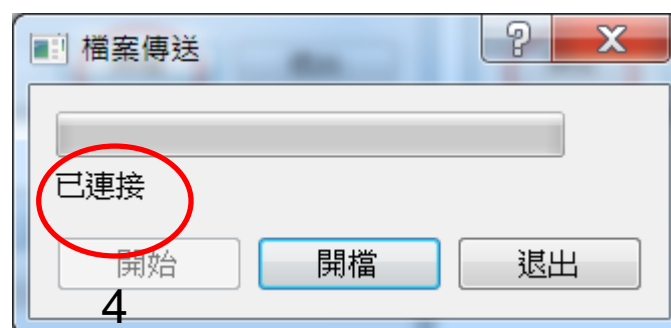
1



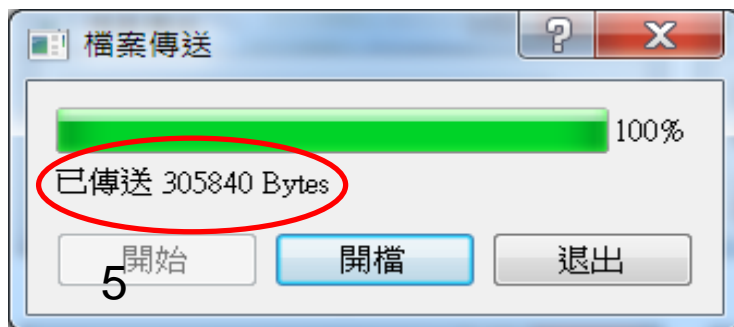
2



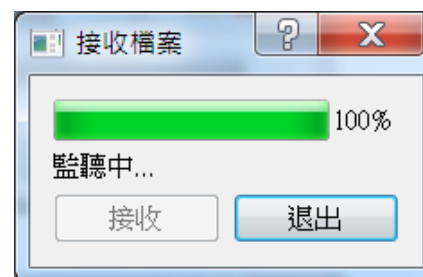
3



4



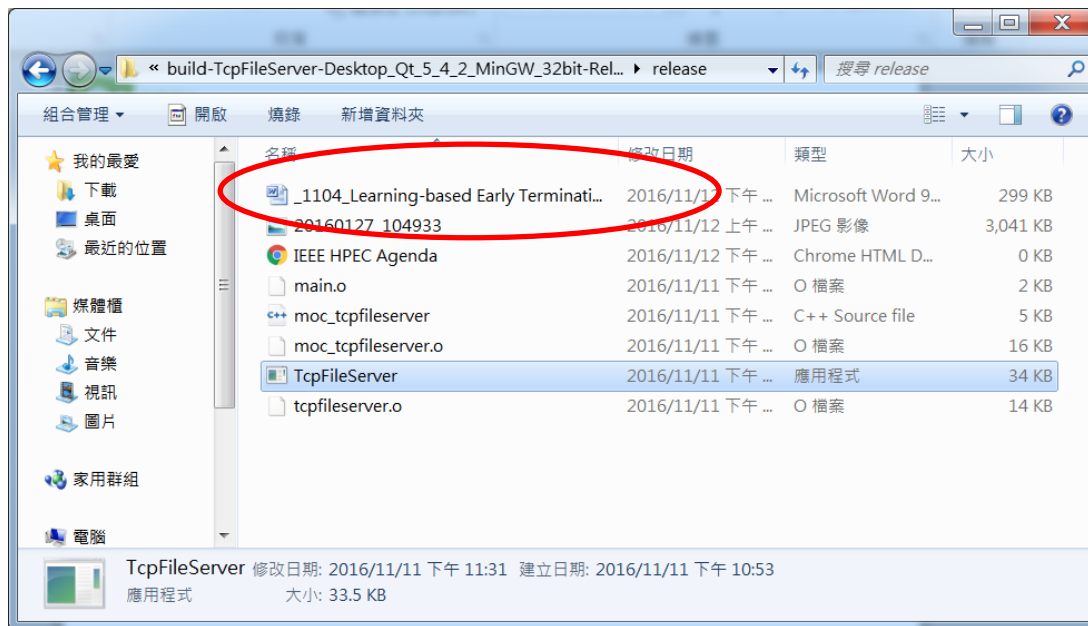
5



Server has received the header part.

Qt- Networking

- Build and run the project:



Open the File Explore to see if the file has been received successfully.

Qt- Networking

- **Signals of TcpSocket used in the project:**

