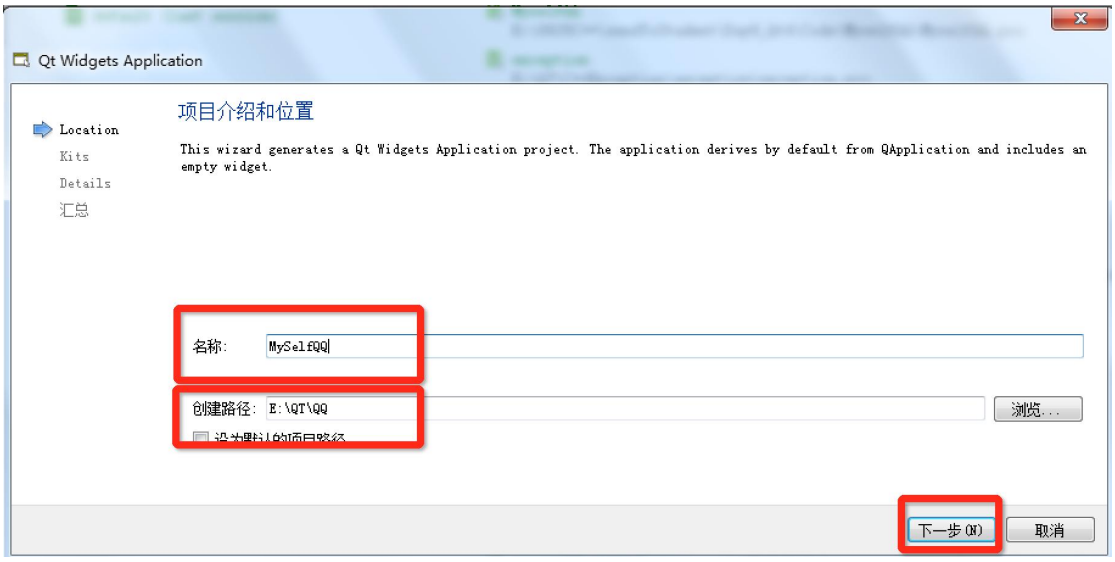
1 新建项目

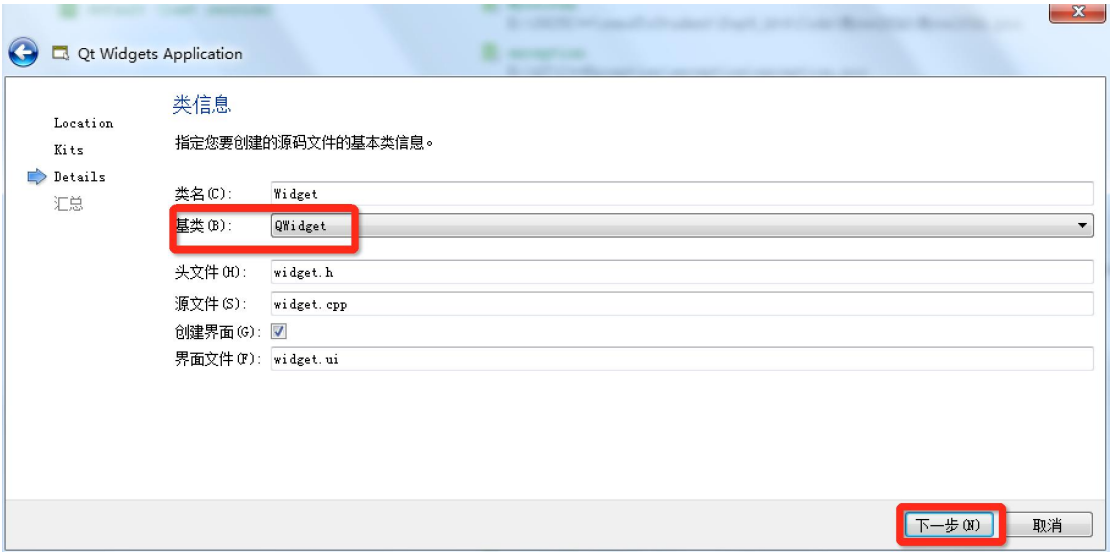
1.1 创建新项目

第一步打开 Qt Creator，点击新建 NewProject
Application –> Qt Widgets Application -> choose

创建项目名称例如： MyselfQQ，路径自己选择，注意不要有空格和中文



选择套件，点击下一步
选择基类 QWidget，然后点击下一步



然后点击完成，至此项目创建完毕。

2 创建对话列表

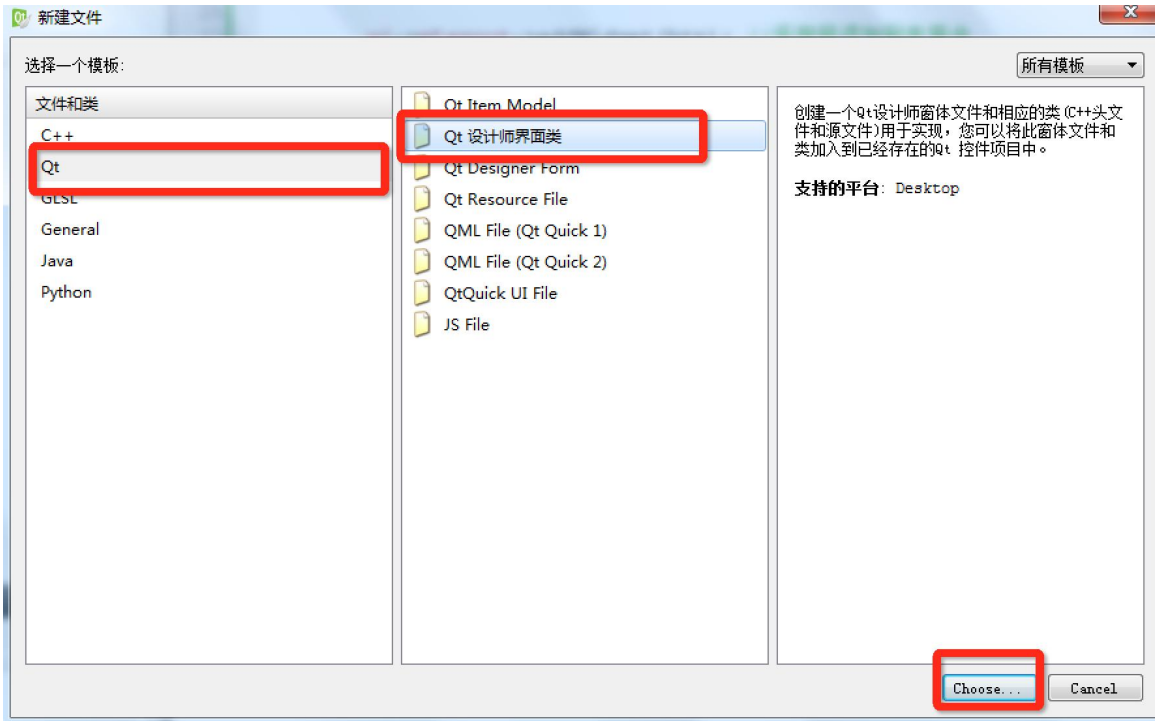
2.1 添加新文件，对话列表类 DialogList

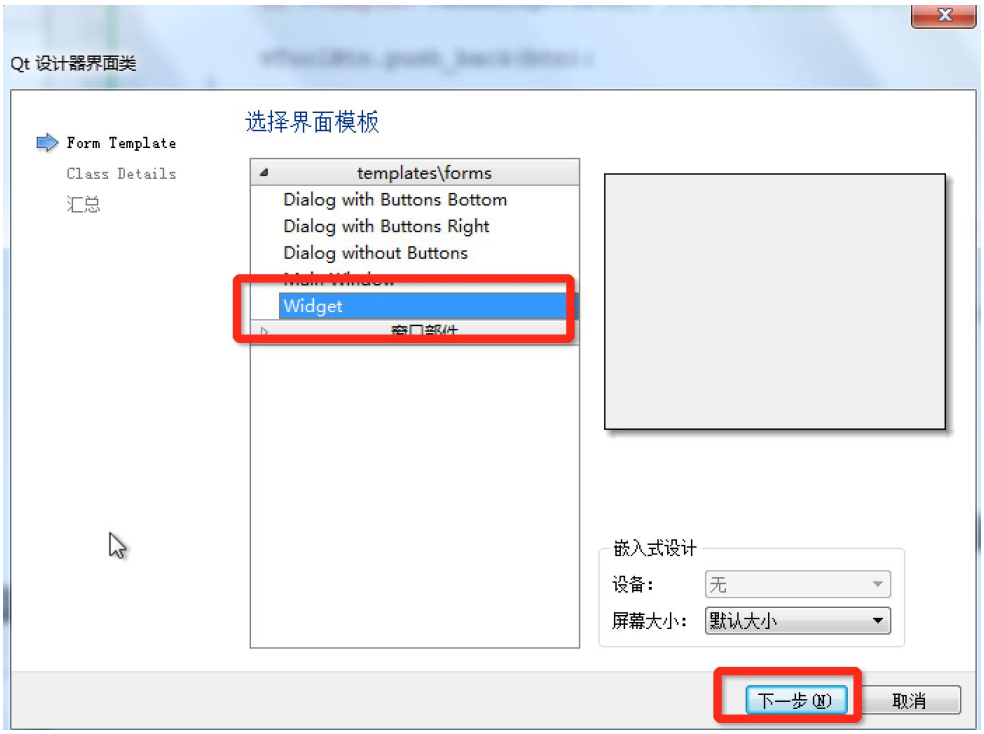
右击项目名，在弹出的快捷菜单中选择“添加新文件...”菜单项，在弹出的对话框中选择“Qt”选项。选择 Qt 设计师界面类，单击“Choose...”按钮；

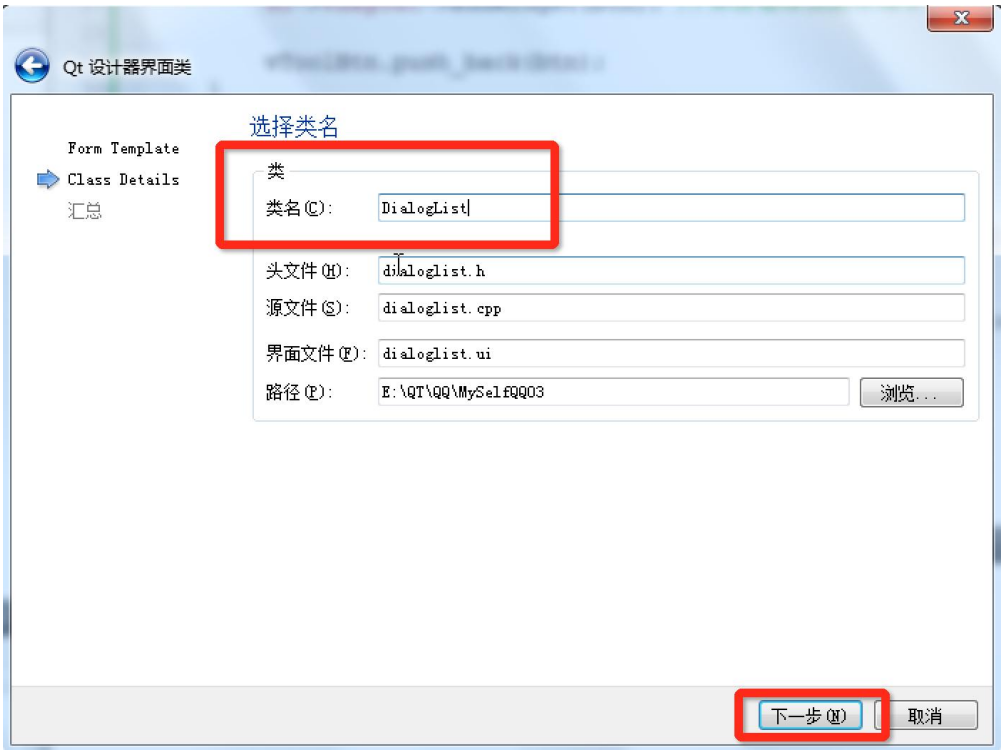
界面模板选择 Widget，点击下一步

类名填写 DialogList （可以起其他名称）点击下一步

在汇总中单击“完成”按钮，系统会为我们添加“dialoglist.h”头文件和“dialoglist.cpp”源文件以及 dialoglist.ui 设计文件

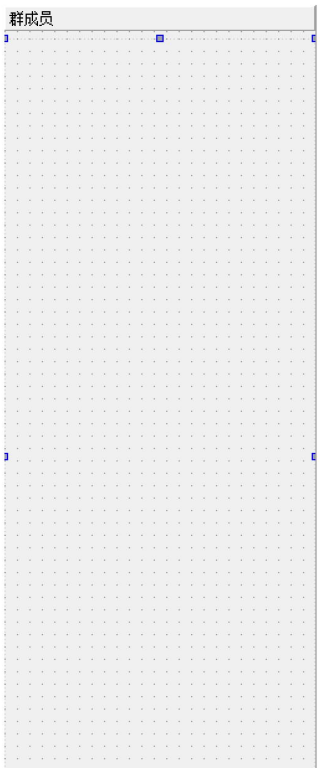



Qt 设计器界面类
选择界面模板
Form Template
Class Details
汇总
templates\forms
Dialog with Buttons Bottom
Dialog with Buttons Right
Dialog without Buttons
Widget
嵌入式设计
设备：
无
屏幕大小：
默认大小
下一步(N)
取消

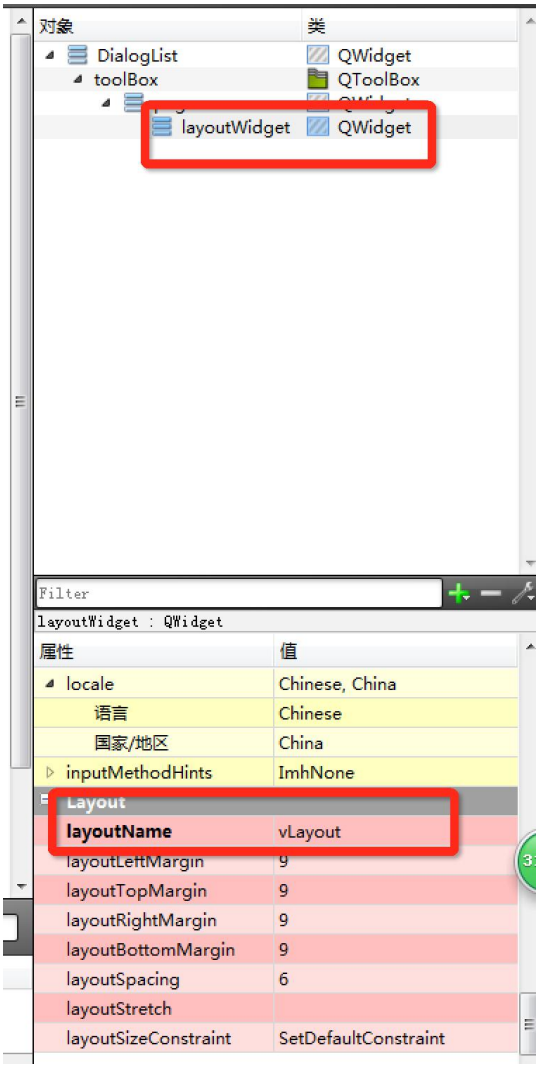

Qt 设计器界面类
选择类名
Form Template
Class Details
汇总
类
类名(C):
DialogList
头文件(H):
dialoglist.h
源文件(S):
dialoglist.cpp
界面文件(F):
dialoglist.ui
路径(P):
E:\QT\QQ\MySelfQQ03
浏览...
下一步(N)
取消

2.2 设计对话框列表 UI

按照下图所示，设计对话框的 UI，该窗口的大小为 250*700，其中的主要控件是 QToolBox，修改该控件的 currentItemText 为“群成员”，QToolBox 默认生成的第二页删除掉



在群成员里我们放入一个 Widget 做布局操作，可以先利用一些测试控件放入到其中，然后做垂直布局，然后把测试的控件删除掉，这时该 Widget 中就有了一个 layout 布局

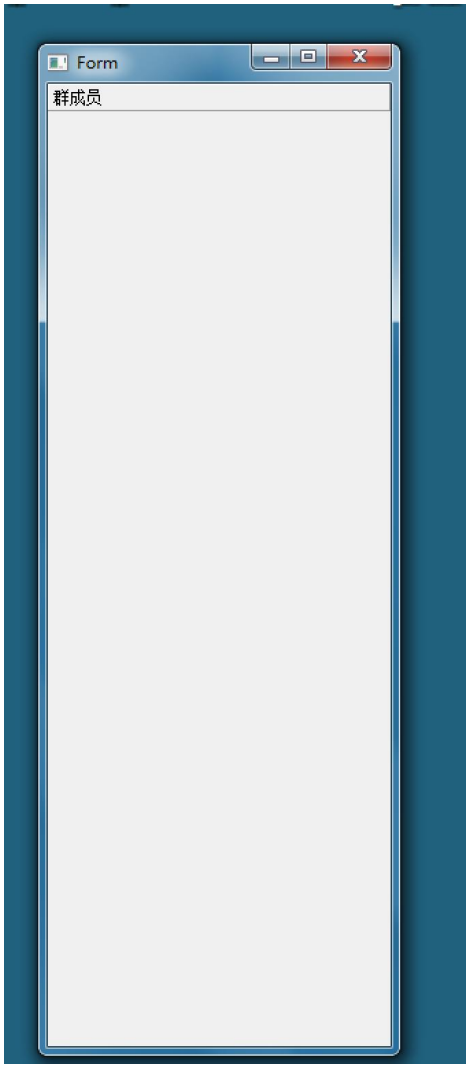


测试，main 函数中修改代码

```
#include "widget.h"
#include <QApplication>
#include "dialoglist.h"
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
//    Widget w;
//    w.show();

    DialogList d;
    d.show();
    return a.exec();
}
```

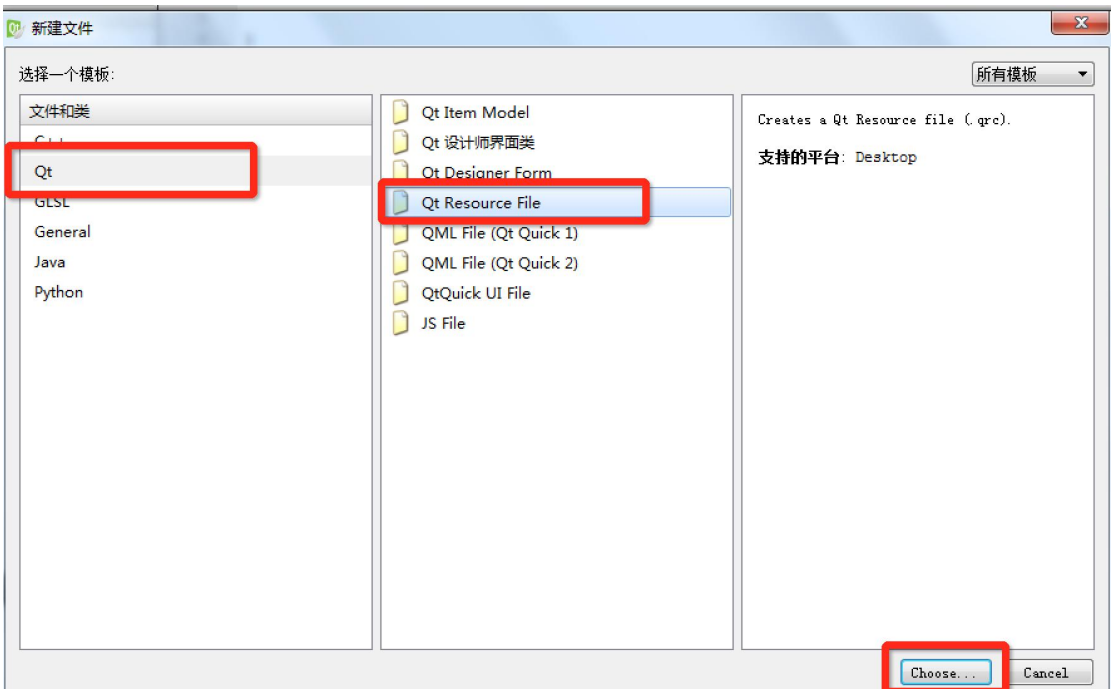
运行项目，效果如图

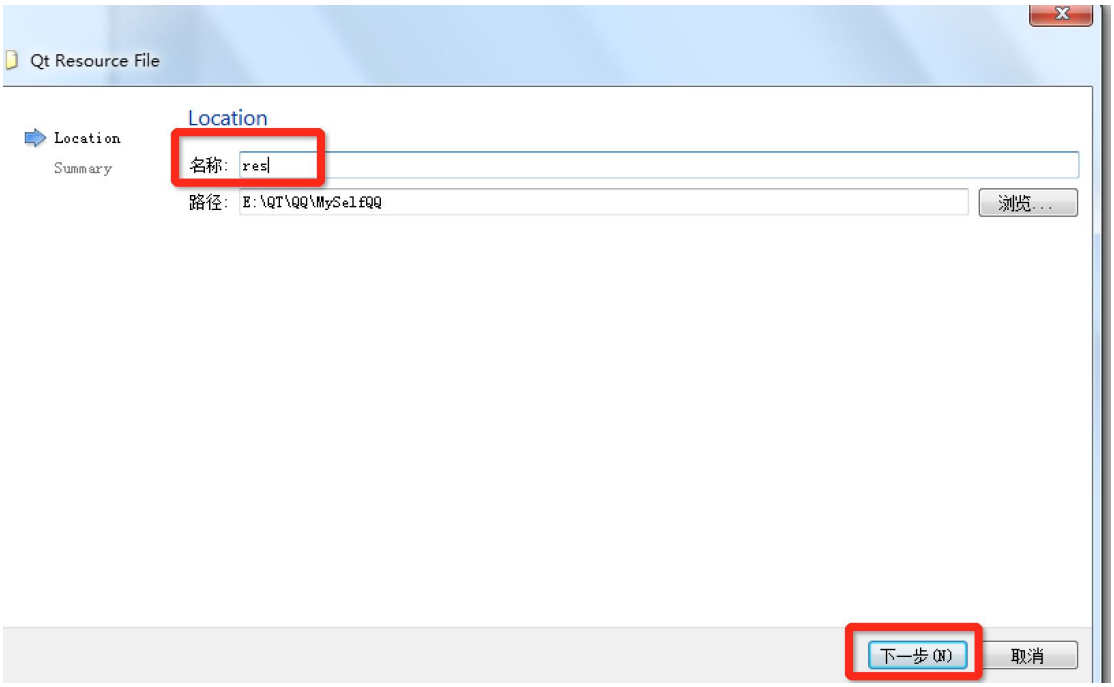


2.3 资源导入

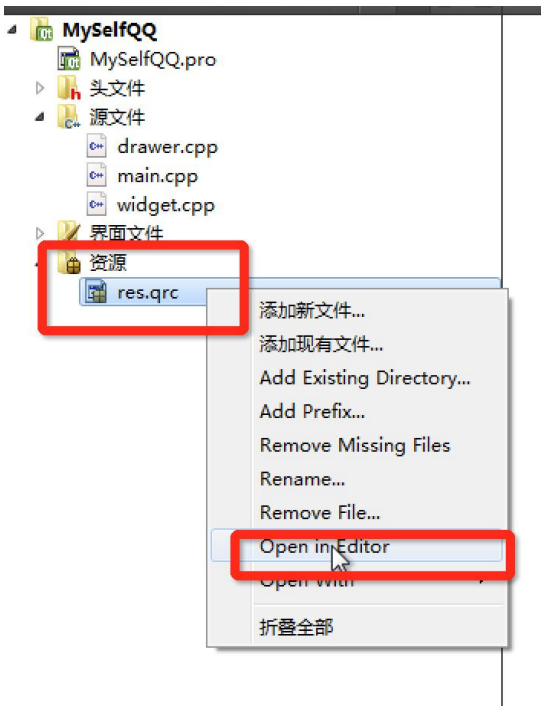
向项目中导入资源，对应九个按钮需要九张图片作为头像图标使用，搜集九张图片（可用共享的资源或者自己收藏的图片，大小在 80*80 左右）

添加新文件 – Qt – Qt Resource 点击 choose 名称 res 下一步，点击完成，生成 res.qrc 文件

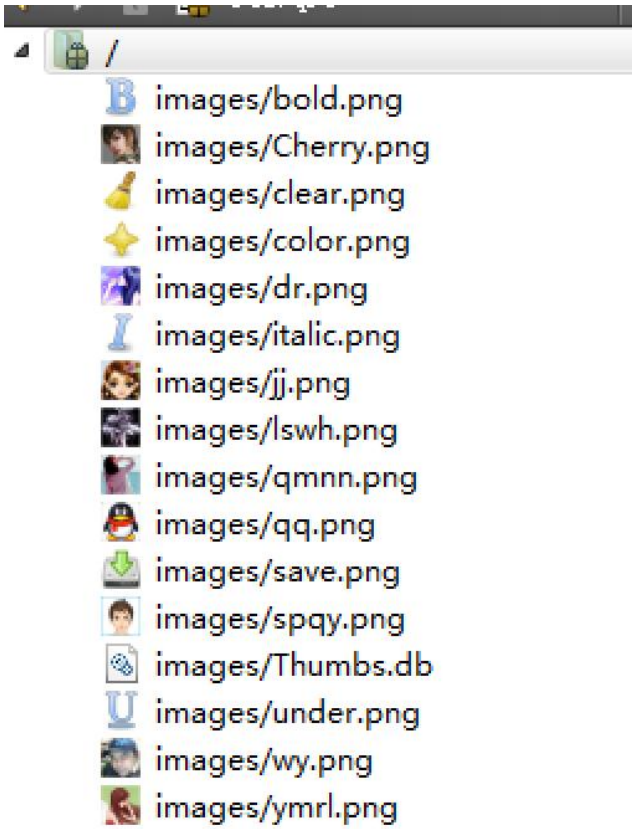




右击 res.qrc，点击 open in Editor ，添加前缀 /



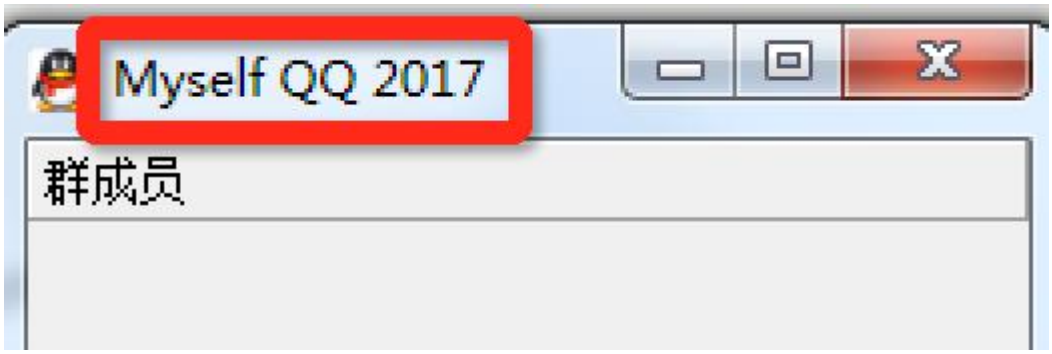
添加文件 – 将准备好的文件选中，点击打开，添加成功



2.4 设置窗体标题和图标

2.4.1 设置标题

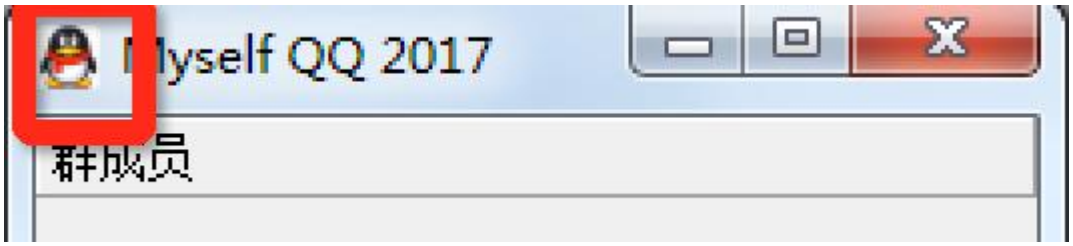
在 Drawer 构造函数中加入如下代码
setWindowTitle(“Myself QQ 2017”);



2.4.2 设置图标

设置主窗体标题图标

setWindowIcon(QPixmap(“:/images/qq.png”));



2.5 设置列表中的按钮

2.5.1 创建 9 个按钮存放到 QVector 容器中

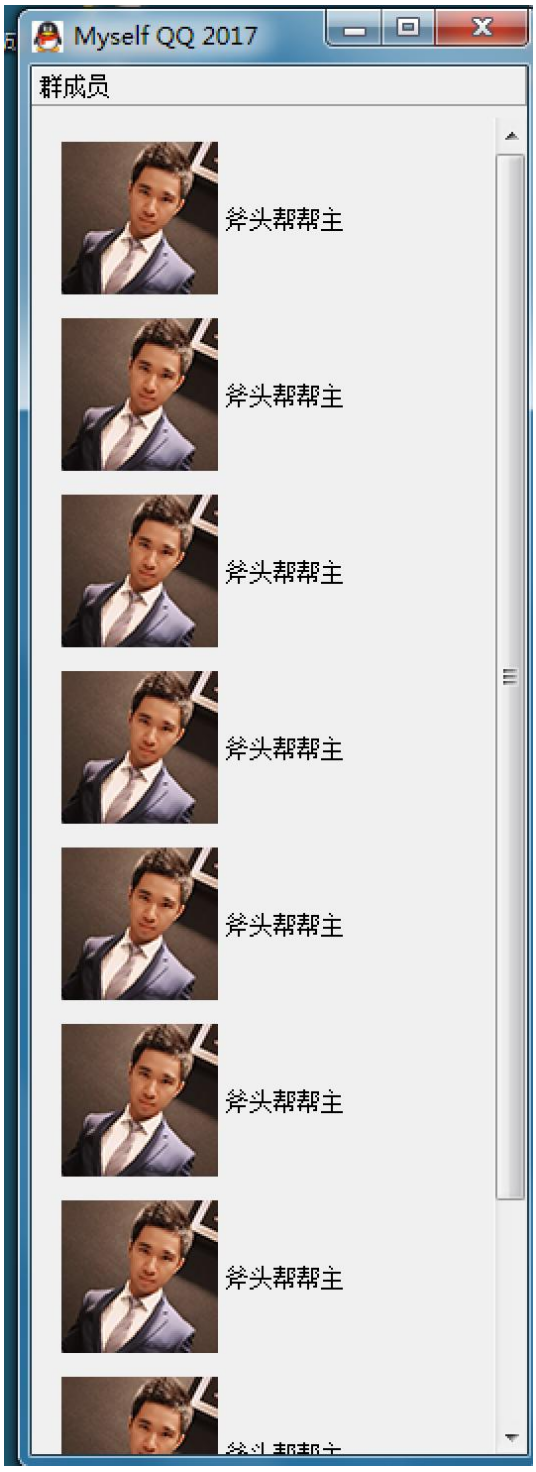
```
    QVector<QToolButton *>vToolBtn; //声明存放 QtoolButton 的容器 vToolBtn

    for(int i = 0 ;i < 9 ; i ++)
    {
        QToolButton * btn = new QToolButton;  //创建新按钮
        btn->setText("斧头帮帮主");     //设置按钮名称
        btn->setIcon(QPixmap(":/images/ftbz.png"));   //设置图片
        btn->setIconSize(QPixmap(":images/ftbz.png").size()); //设置图片大小
        btn->setAutoRaise(true);  //设置图片透明效果
        btn->setToolButtonStyle(Qt::ToolButtonTextBesideIcon);//设置按钮风格，同时显示文字和图标

        ui->vLayout->addWidget(btn); //将按钮添加到布局中

        vToolBtn.push_back(btn); //将 9 个按钮放入到布局中
  }
```

运行效果如图：（有点辣眼睛。。。）



2.5.2 替换图片，改为不同的资源

```
// QList<QString> 等同于 QStringList
QList<QString>nameList;
nameList << "斧头帮帮主" << "忆梦如澜" <<"北京出版人"<<"Cherry"<<"淡然"
         <<"娇娇girl"<<"落水无痕"<<"青墨暖暖"<<"无语";
QStringList iconNameList; //图标资源列表
iconNameList << "ftbz"<< "ymrl" <<"qq" <<"Cherry"<< "dr"
             <<"jj"<<"lswh"<<"qmnn"<<"wy";
```

声明姓名和图标资源的字符串

```
for(int i = 0 ;i < 9 ; i ++)
{
    QToolButton * btn = new QToolButton;  //创建新按钮
    btn->setText(nameList[i]);      //设置按钮名称
    QString iconName = QString(":/images/%1.png").arg(iconNameList.at(i));
    btn->setIcon(QPixmap(iconName));    //设置图片
    btn->setIconSize(QPixmap(iconName).size()); //设置图片大小
    btn->setAutoRaise(true);  //设置图片透明效果
```

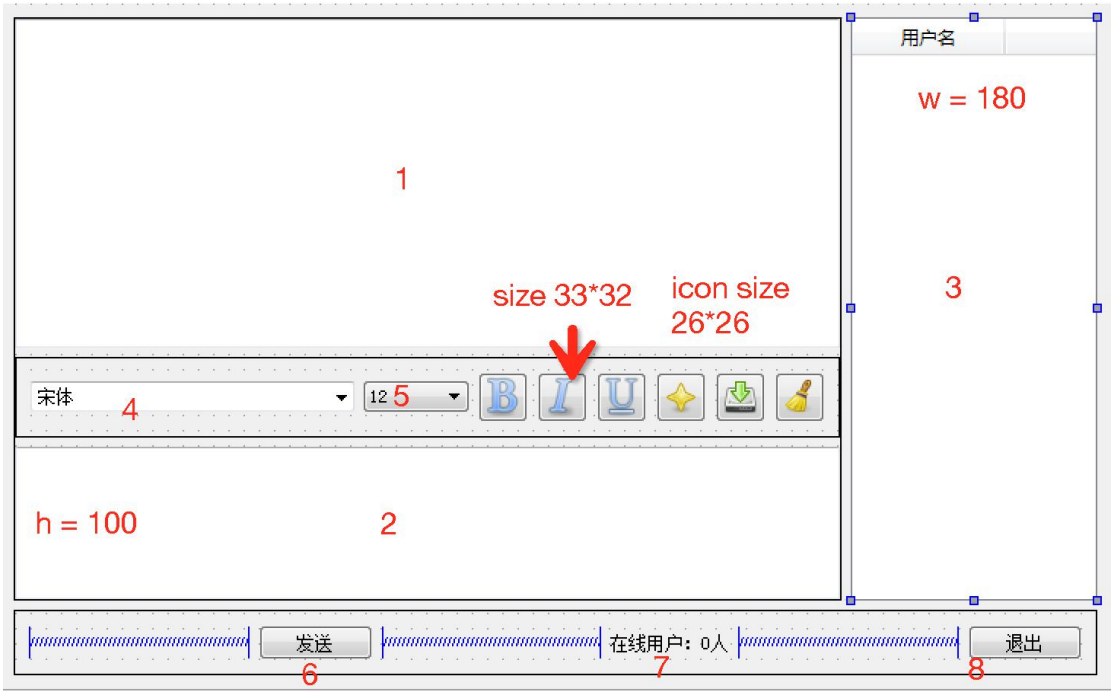
修改9个图片的资源

再次运行，效果如图

3 设计聊天窗口

3.1 界面

双击 widget.ui 文件进入设计模式，界面宽度属性分别设置为 730 和 450，向界面中拖入部件并且进行设置，最终效果如图



3.2 控件类型和属性设置

3.2.1 各个控件设置

如上图中标注的 1~8，下表中为上图的属性设置以及控件类型

序号或图标	类　　型	objectName 属性
①	Text Browser	msgBrowser
②	Text Edit	msgTxtEdit
③	Table Widget	usrTblWidget
④	Font Combo Box	fontCbx
⑤	Combo Box	sizeCbx
⑥	Push Button	sendBtn
⑦	Label	usrNumLbl
⑧	Push Button	exitBtn
	Tool Button	boldTBtn
	Tool Button	italicTBtn
	Tool Button	underlineTBtn
	Tool Button	colorTBtn
	Tool Button	saveTBtn
	Tool Button	clearTBtn

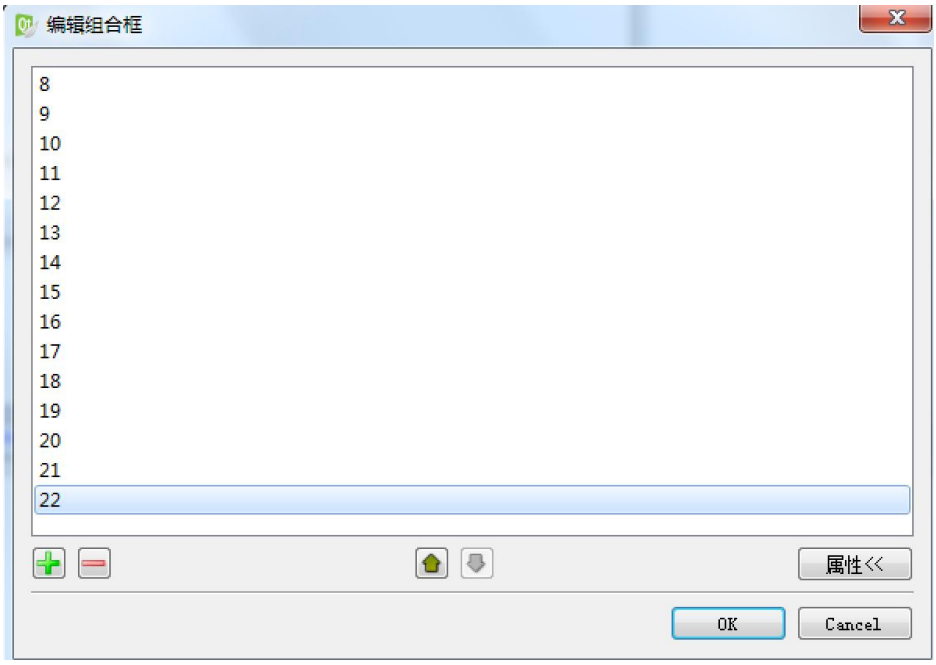
3.2.2 ToolBtn 详细设置

其中前三个按钮  ，选中　checkable 属性；

其中所有的 ToolBtn  属性中的 toolTIp 依次更改为 加粗、倾斜、下划线、更改字体颜色、保存聊天记录和 清空聊天记录

3.2.3 字体大小下拉框设置

界面上 5 号控件设置字体大小，设置区间为 8~22（与腾讯 QQ 软件完全相同），双击该部件，点击 + 号按钮添加新项目如图

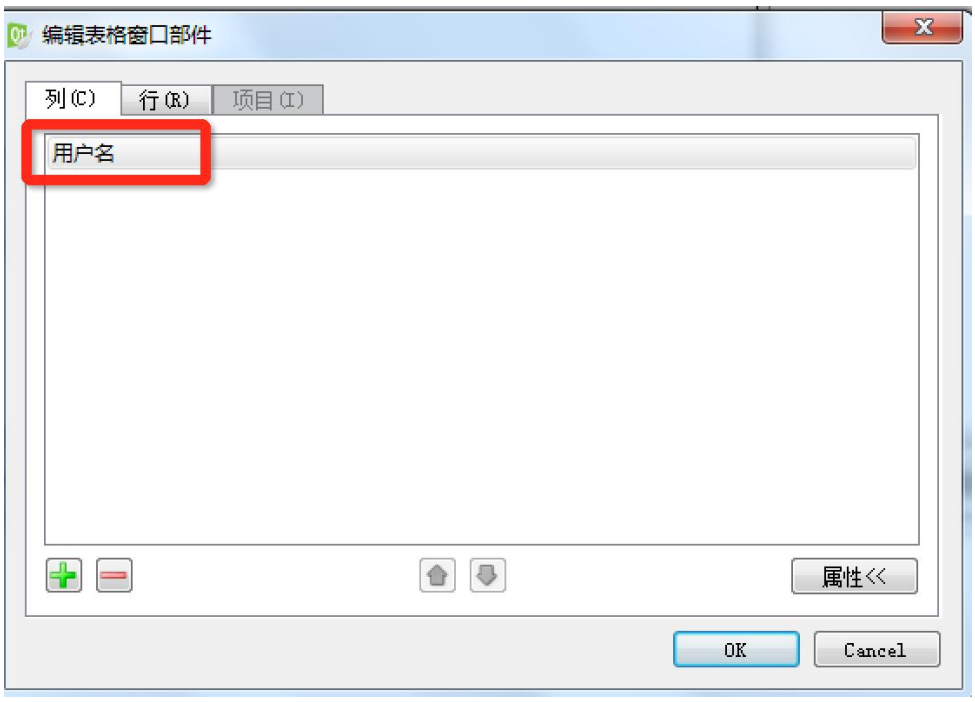


currentIndex 属性设置为 4，即默认为 12 号字

3.2.4 TableWidget 设置

显示用户列表的 TableWidget 控件，将 selectionModel 属性选择为 SingleSelection（带有选中效果），将 selectBehavior 选择为 SelectRows（选中整行），取消选中的 showGrid（表格显示）

双击 TableWidget 部件，添加“用户名”列，如图



至此，所有弹出的聊天信息窗口全部设计完毕

4 关联图片按钮与聊天窗口

4.1 添加按钮点击事件

下一步就是要关联起聊天的列表窗口和具体的聊天信息窗口了，也就是点击按钮弹出窗口。

```
    //添加点击事件
    for(int i = 0 ; i < vToolBtn.size();i++)
    {
        connect(vToolBtn[i],&QToolButton::clicked,this,[=]()
        {
            //qDebug() <<i;
            //此时，widget 的构造函数已经修改，（见 4.2 步骤）创建 widget 窗口时，参数 1 ：0 代表以顶层方式弹出
            //参数 2：代表 vToolBtn[i]->text()代表告诉聊天窗口 人物的姓名
           Widget *chatWidget = new Widget(0,vToolBtn[i]->text());
           chatWidget->setWindowTitle(vToolBtn[i]->text());
           chatWidget->setWindowIcon(vToolBtn[i]->icon());
           chatWidget->show();
        });
    }
```

4.2 修改弹出框的构造函数

!!! Widget 文件中的 构造改为 explicit Widget(QWidget *parent ,QString usrname); 参数 2 用于传入用户名，让具体的弹出框知道自己的用户名是什么

4.3 测试

运行代码，点击窗口，弹出响应的对话框

4.6 解决一个窗口多次弹出的 bug

在 dialogList.h 中加入标示

QVector<bool>isShow; 代表是否打开窗口的标识，false 未打开，true 打开

dialogList.cpp 构造中 将 9 个标志位都置为 false

```
    //九个标识位设置
    for(int i = 0 ; i < 9 ;i++)
    {
        isShow.push_back(false);
    }
```

修改点击事件，根据打开的标志位来判断

```
//添加点击事件
for(int i = 0 ; i < vToolBtn.size();i++)
{
    connect(vToolBtn[i],&QToolButton::clicked,this,[=]()
    {
        if(isShow[i]) //如果打开的标志位为true，不要重复打开
        {
            QMessageBox::warning(this,"警告",QString("用户%1窗口已弹出").arg(vToolBtn[i]->text())
            return;
        }
        isShow[i] = true; //如果打开标识位不为ture，可以打开，并且修改打开标志位为true

        //qDebug() <<i;
        //此时，widget的构造函数已经修改，创建widget窗口时，参数1 ：0代表以顶层方式弹出
        //参数2：代表vToolBtn[i]->text()代表告诉聊天窗口 人物的姓名
       Widget *chatWidget = new Widget(0,vToolBtn[i]->text());
       chatWidget->setWindowTitle(vToolBtn[i]->text());
       chatWidget->setWindowIcon(vToolBtn[i]->icon());
       chatWidget->show();
    });
}
```

然后在 widget 函数中声明关闭的信号，和重写关闭方法

```
signals:
    void widgetClose();

protected:
   void closeEvent(QCloseEvent *);
```

然后实现重写的关闭方法，发送关闭信号

```
void Widget::closeEvent(QCloseEvent * e)
{
    emit this->widgetClose(); //发送关闭当前窗口的自定义信号
    QWidget::closeEvent(e);
}
```

点击事件中再添加一行代码

```
//添加点击事件
for(int i = 0 ; i < vToolBtn.size();i++)
{
    connect(vToolBtn[i],&QToolButton::clicked,this,[=]()
    {
        if(isShow[i]) //如果打开的标志位为true，不要重复打开
        {
            QMessageBox::warning(this,"警告",QString("用户%1窗口已弹出").arg(vToolBtn[i]->text
            return;
        }
        isShow[i] = true; //如果打开标识位不为ture，可以打开，并且修改打开标志位为true

        //qDebug() <<i;
        //此时，widget的构造函数已经修改，创建widget窗口时，参数1 ：0代表以顶层方式弹出
        //参数2：代表vToolBtn[i]->text()代表告诉聊天窗口 人物的姓名
       Widget *chatWidget = new Widget(0,vToolBtn[i]->text());
       chatWidget->setWindowTitle(vToolBtn[i]->text());
       chatWidget->setWindowIcon(vToolBtn[i]->icon());
       chatWidget->show();

         //如果窗口关闭，将标志位设置回来
       connect(chatWidget,&Widget::widgetClose,this,[=](){isShow[i] = false;});

    });
```

测试，一个窗口不能多次弹出

5 实现基本聊天功能

widget 中 定义枚举 enum MsgType {Msg,UsrEnter,UsrLeft} 分别代表 聊天信息、新用户加入、用户退出

5.1 声明聊天的方法

widge.h 中

```
public:
    void sndMsg(MsgType type); //广播 UDP 消息
    void usrEnter(QString username);//处理新用户加入
    void usrLeft(QString usrname,QString time); //处理用户离开
    QString getUsr(); //获取用户名
    QString getMsg(); //获取聊天信息
private:
    QUdpSocket * udpSocket; //udp 套接字
    qint16 port; //端口
    QString uName; //用户名

    void ReceiveMessage();    //接受 UDP 消息
```

5.2 widget.cpp 中实现

构造函数中：

```
this->uName = usrname; //获取用户名
udpSocket = new QUdpSocket(this);
port = 23333;
udpSocket->bind(port,QUdpSocket::ShareAddress |QUdpSocket::ReuseAddressHint);    //采用 ShareAddress 模式(即允许其它的服务连接到相同的地址和端口，特别是用在多客户端监听同一个服务器端口等时特别有效)，和 ReuseAddressHint 模式(重新连接服务器)
connect(udpSocket,&QUdpSocket::readyRead,this,&Widget::ReceiveMessage); 监听信号
sndMsg(UsrEnter);//有新用户加入
```

发送消息函数 sndMsg

```
void Widget::sndMsg(MsgType type)
{
    QByteArray data;
    QDataStream out(&data,QIODevice::WriteOnly);
    out << type << getUsr();   //将消息类型 和 用户名 放入到流中
    switch (type) {
    case Msg:
        if(ui->msgTxtEdit->toPlainText() == "")
        {
            QMessageBox::warning(0,"警告","发送内容不能为空",QMessageBox::Ok);
            return;
        }
        out  <<getMsg();  //发送的是聊天信息     发送格式    消息类型 + 用户名 + 发送内容
        break;
    case UsrEnter:        //发送的是新用户进入  发送格式    消息类型 + 用户名
        break;
    case UsrLeft:         // 发送的是用户离开  发送格式     消息类型 + 用户名
        break;
    default:
        break;
    }
    udpSocket->writeDatagram(data,data.length(),QHostAddress::Broadcast,port);
}
```

接受消息 ReceiveMessage

```
void Widget::ReceiveMessage()
```

```
{

    QByteArray datagram;
    datagram.resize(udpSocket->pendingDatagramSize());
    udpSocket->readDatagram(datagram.data(),datagram.size());
    QDataStream in (&datagram,QIODevice::ReadOnly);
    int msgType;
    in >> msgType; //用户类型获取

    QString usrName,msg;  //用户名、信息
    QString time = QDateTime::currentDateTime().toString("yyyy-MM-dd
hh:mm:ss");

    switch (msgType){
    case Msg:
        in >> usrName  >>msg;
        ui->msgBrowser->setTextColor(Qt::blue);
        ui->msgBrowser->setCurrentFont(QFont("Times New Roman",12));
        ui->msgBrowser->append("[ " + usrName + " ]" + time);
        ui->msgBrowser->append(msg);
        break;
    case UsrEnter:
        in >> usrName ;
        usrEnter(usrName);
        break;
    case UsrLeft:
        in >> usrName;
        usrLeft(usrName,time);
        break;
    default:
        break;
    }
}
```

获取用户名

```
//获取用户名
QString Widget::getUsr()
{
    return uName;
}
```

获取聊天信息

```
//获取聊天信息
QString Widget::getMsg()
{

    QString msg = ui->msgTxtEdit->toHtml();
    ui->msgTxtEdit->clear();
    ui->msgTxtEdit->setFocus();
    return msg;
}
```

新用户进入

```
//处理新用户加入
void Widget::usrEnter(QString usrname)
{
    bool isEmpty =
ui->usrTblWidget->findItems(usrname,Qt::MatchExactly).isEmpty();
    if(isEmpty)
    {
        QTableWidgetItem *usr = new QTableWidgetItem(usrname);

        ui->usrTblWidget->insertRow(0);
        ui->usrTblWidget->setItem(0,0,usr);
        ui->msgBrowser->setTextColor(Qt::gray);
        ui->msgBrowser->setCurrentFont(QFont("Times New Roman",10));
        ui->msgBrowser->append(tr("%1 在线！").arg(usrname));
        ui->usrNumLbl->setText(tr("在线人
数：%1").arg(ui->usrTblWidget->rowCount()));

        //已经在线的各个端点也要告知新加入的端点他们自己的信息，若不这样做，在新
端点用户列表中就无法显示其他已经在线的用户
        sndMsg(UsrEnter);
    }
}
```

用户离开

```
//用户离开
void Widget::usrLeft(QString usrname,QString time)
{
    int rowNum = ui->usrTblWidget->findItems(usrname,
Qt::MatchExactly).first()->row();
    ui->usrTblWidget->removeRow(rowNum);
    ui->msgBrowser->setTextColor(Qt::gray);
```

```
    ui->msgBrowser->setCurrentFont(QFont("Times New Roman", 10));
    ui->msgBrowser->append(QString("%1 于 %2 离开！").arg(usrname).arg(time));
    ui->usrNumLbl->setText(QString("在线人
数：%1").arg(ui->usrTblWidget->rowCount()));
}
```

重写离开 Event　void *closeEvent*(QCloseEvent *)

```
void Widget::closeEvent(QCloseEvent *e)
{
    emit this->widgetClose();
    sndMsg(UsrLeft);
    udpSocket->close();
    udpSocket->destroyed();
    QWidget::closeEvent(e);
}
```

发送按钮和离开按钮信号槽连接

```
    connect(ui->sendBtn,&QPushButton::clicked,this,[=](){
        sndMsg(Msg);
    });

    connect(ui->exitBtn,&QPushButton::clicked,this,[=]()
    {
        this->close();
    });
```

至此可以测试聊天功能

6 辅助功能

6.1 字体设置

```
    connect(ui->fontCbx,&QFontComboBox::currentFontChanged,this,[=](const QFont &f){
        ui->msgTxtEdit->setCurrentFont(f);
        ui->msgTxtEdit->setFocus();
    });
```

6.2 字号设置

```
    void (QComboBox:: * cbxSingal)(const QString &text) = &QComboBox::currentIndexChanged;
    connect(ui->sizeCbx,cbxSingal,this,[=](const QString &text){
        ui->msgTxtEdit->setFontPointSize(text.toDouble());
        ui->msgTxtEdit->setFocus();
    });
```

6.3 加粗

```
    connect(ui->boldTBtn,&QToolButton::clicked,this,[=](bool checked){
        if(checked)
        {
            ui->msgTxtEdit->setFontWeight(QFont::Bold);
        }
        else
        {
            ui->msgTxtEdit->setFontWeight(QFont::Normal);
        }
        ui->msgTxtEdit->setFocus();
    });
```

6.4 倾斜

```
    connect(ui->italicTBtn,&QToolButton::clicked,this,[=](bool checked){

        ui->msgTxtEdit->setFontItalic(checked);

        ui->msgTxtEdit->setFocus();
    });
```

6.5 下划线

```
    connect(ui->underlineTBtn,&QToolButton::clicked,this,[=](bool checked){
```

```
        ui->msgTxtEdit->setFontUnderline(checked);

        ui->msgTxtEdit->setFocus();
    });
```

6.6 设置文本颜色

```
    connect(ui->colorTBtn,&QToolButton::clicked,this,[=](){
        color = QColorDialog::getColor(color,this); //color 对象可以在 widget.h 中定义私有成员
        if(color.isValid())
        {
            ui->msgTxtEdit->setTextColor(color);
            ui->msgTxtEdit->setFocus();
        }
    });
```

6.7 保存聊天记录

```
connect(ui->saveBtn,&QToolButton::clicked,this,[=](){
            if(ui->msgBrowser->document()->isEmpty())
            {
                QMessageBox::warning(this,"警告","聊天记录为空，无法保存!
",QMessageBox::Ok);
            }
            else
            {
                QString fName = QFileDialog::getSaveFileName(this,"保存聊天记录
","聊天记录","(*.txt)");
                if(!fName.isEmpty())
                {
                    //保存名称不为空再做保存操作
                    QFile file(fName);
                    file.open(QIODevice::WriteOnly | QFile::Text);

                    QTextStream stream(&file);
                    stream << ui->msgBrowser->toPlainText();
                    file.close();
                }
            }
        });
```

6.8 清空聊天记录

```
connect(ui->clearTBtn,&QToolButton::clicked,[=](){
    ui->msgBrowser->clear();
});
```

至此本案例全部完成，可以测试