1 QLabel控件

1.1设置标签的文本

```
#include "widget.h"

Widget::Widget(QWidget *parent)
 : QWidget(parent)
{
 this->resize(600,400);
 label = new QLabel(this);
 label->resize(200,200);
 label->move(100,100);
  // label->setText("我是一个标签");
  label ->setText("<a href=\"https://www.baidu.com\">百度一下</a>");
  label->setOpenExternalLinks(true);//设置点击链接自动打开
  }

Widget::~Widget()
{

}

```

1.2 设置标签的图片

```
Widget::Widget(QWidget *parent)
 : QWidget(parent)
{
 this->resize(600,400);
 label = new QLabel(this);
 label->resize(100,100);
 label->move(100,100);
 // label->setText("我是一个标签");
 label ->setText("<a href=\"https://www.baidu.com\">百度一下</a>");
  label->setOpenExternalLinks(true);//设置点击链接自动打开


  label_pic = new QLabel(this);
  label_pic->resize(100,100);
  label_pic->move(200,200);
  label_pic->setPixmap(QPixmap("../Image/face.png"));
  label_pic->setScaledContents(true);//设置自适应大小


  }

```

1.3 设置标签的动态图

```
#include "widget.h"
#include <QPixmap>
#include <QMovie>
Widget::Widget(QWidget *parent)
 : QWidget(parent)
{
 this->resize(600,400);
 label = new QLabel(this);
 label->resize(100,100);
  label->move(100,100);
  // label->setText("我是一个标签");
  label ->setText("<a href=\"https://www.baidu.com\">百度一下</a>");
  label->setOpenExternalLinks(true);//设置点击链接自动打开


  label_pic = new QLabel(this);
```

```
 label_pic->resize(100,100);
 label_pic->move(200,200);
 label_pic->setPixmap(QPixmap("../Image/face.png"));
 label_pic->setScaledContents(true);//设置自适应大小

 label_move = new QLabel(this);
 label_move->resize(100,100);
 label_move->move(300,200);

 QMovie * move = new QMovie("../Image/boy.gif");
 label_move->setMovie(move);
 label_move->setScaledContents(true);
 move->start();



 }

Widget::~Widget()
{

}

```

2 QLineEdit

2.1 设置QlineEdit内容和获取内容

```
#include "widget.h"
#include "ui_widget.h"
#include <QDebug>
Widget::Widget(QWidget *parent) :
 QWidget(parent),
 ui(new Ui::Widget)
{
 ui->setupUi(this);
 ui->lineEdit->setText("hello");//设置行编辑内容
 }

Widget::~Widget()
{
 delete ui;
```

```
}

void Widget::on_lineEdit_returnPressed()
{
 qDebug()<<ui->lineEdit->text();//获取行编辑内容
}

```

2.1 设置行编辑密码模式

```
#include "widget.h"
#include "ui_widget.h"
#include <QDebug>
Widget::Widget(QWidget *parent) :
 QWidget(parent),
 ui(new Ui::Widget)
{
 ui->setupUi(this);
 ui->lineEdit->setEchoMode(QLineEdit::Password);//设置密码模式
  ui->lineEdit->setTextMargins(100,0,0,0);//设置间距
  ui->lineEdit->setText("hello");//设置行编辑内容
}

Widget::~Widget()
{
 delete ui;
}

void Widget::on_lineEdit_returnPressed()
{
 qDebug()<<ui->lineEdit->text();//获取行编辑内容
}

```

3 自定义控件

在搭建Qt窗口界面的时候，在一个项目中很多窗口，或者是窗口中的某个模块会被经常性的重复使用。一般遇到这种情况我们都会将这个窗口或者模块拿出来做

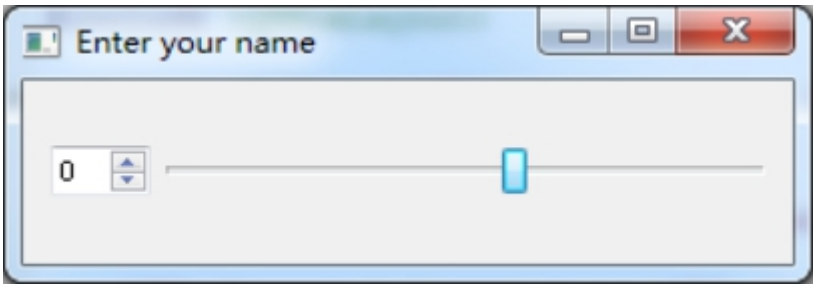
成一个独立的窗口类，以备以后重复使用。

在使用Qt的ui文件搭建界面的时候，工具栏栏中只为我们提供了标准的窗口控件，如果我们想使用自定义控件怎么办？

例如：我们从QWidget派生出一个类SmallWidget，实现了一个自定窗口，

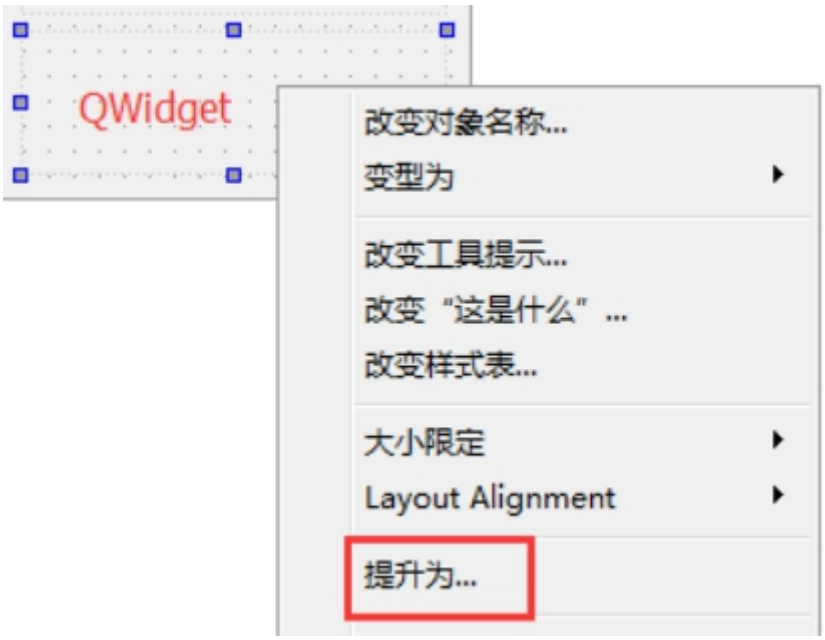
```
// smallwidget.h
class SmallWidget : public QWidget
{
    Q_OBJECT
public:
    explicit SmallWidget(QWidget *parent = 0);

signals:

public slots:
private:
    QSpinBox* spin;
    QSlider* slider;
};

// smallwidget.cpp
SmallWidget::SmallWidget(QWidget *parent) : QWidget(parent)
{
    spin = new QSpinBox(this);
    slider = new QSlider(Qt::Horizontal, this);

        // 创建布局对象
    QHBoxLayout* layout = new QHBoxLayout;
        // 将控件添加到布局中
    layout->addWidget(spin);
    layout->addWidget(slider);
        // 将布局设置到窗口中
    setLayout(layout);

        // 添加消息响应
```

```
    connect(spin,
static_cast<void (QSpinBox::*)(int)>(&QSpinBox::valueChanged),
 slider, &QSlider::setValue);
    connect(slider, &QSlider::valueChanged,
spin, &QSpinBox::setValue);
}
```

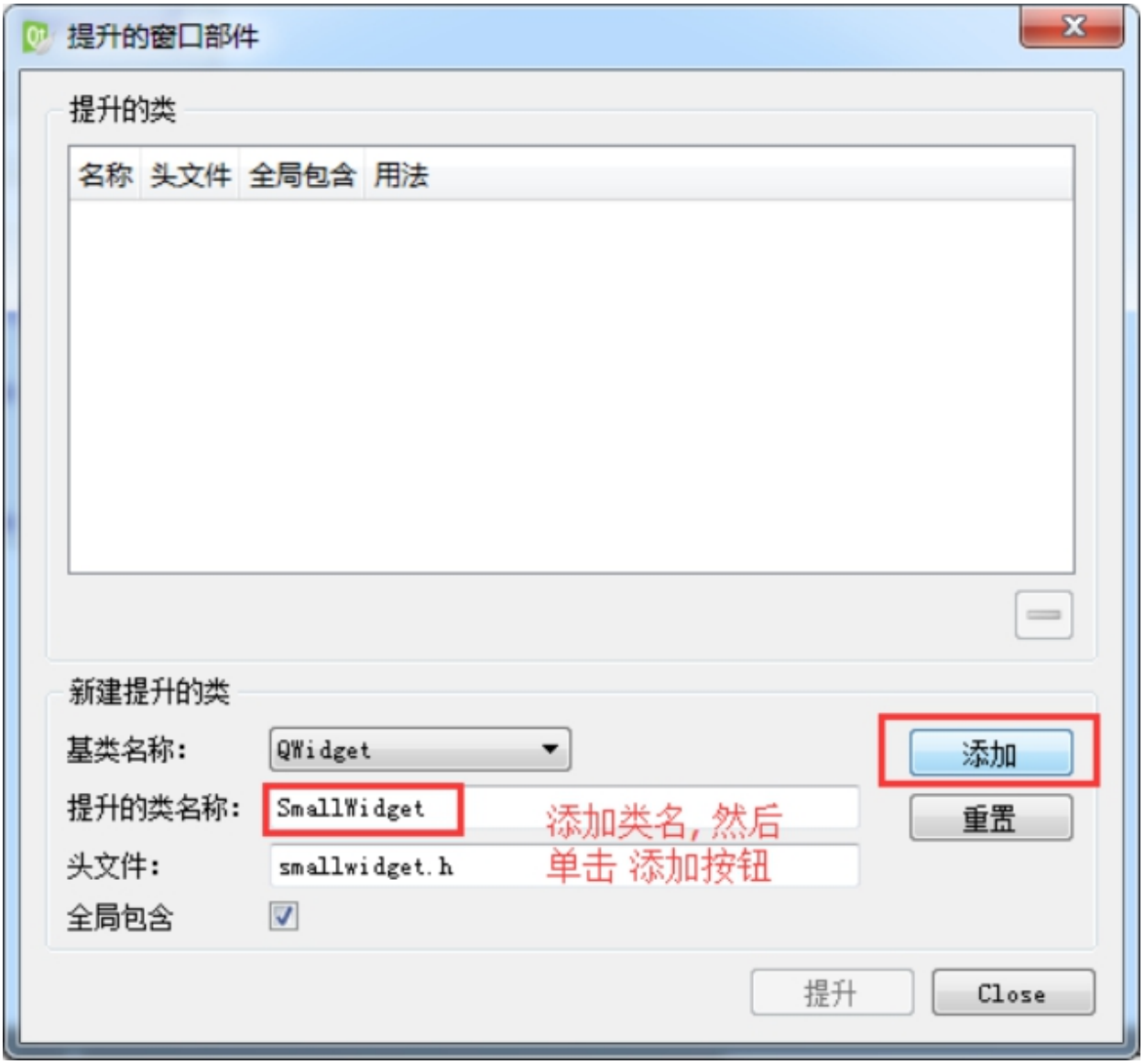


那么这个SmallWidget可以作为独立的窗口显示,也可以作为一个控件来使用:

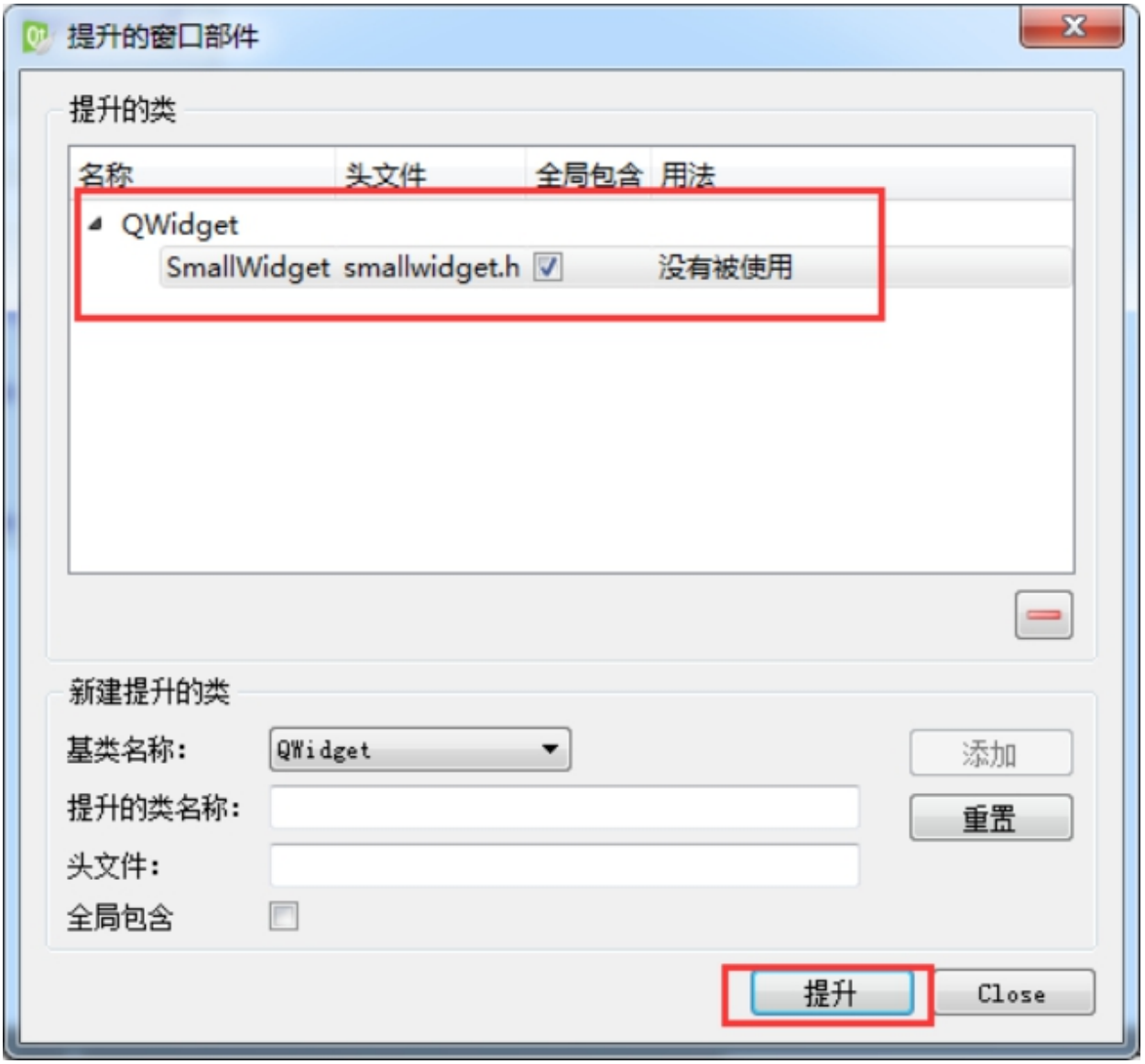
打开Qt的.ui文件,因为SmallWidget是派生自Qwidget类,所以需需要在ui文件中先放入一个QWidget控件, 然后再上边鼠标右键



弹出提升窗口部件对话框



添加要提升的类的名字,然后选择 添加



添加之后,类名会显示到上边的列表框中,然后单击提升按钮,完成操作.

我们可以看到, 这个窗口对应的类从原来的QWidget变成了SmallWidget

再次运行程序,这个widget_3中就能显示出我们自定义的窗口了.

4 定时器

QTimer

定时触发

```
#include "widget.h"
#include "ui_widget.h"
#include <QDebug>
```

```
Widget::Widget(QWidget *parent) :
 QWidget(parent),
 ui(new Ui::Widget)
{
 ui->setupUi(this);
 t = new QTimer;
  //设置定时器的超时事件 如果超时 会发出一个超时信号
  connect(t,&QTimer::timeout,this,[](){

  qDebug()<<"timeout";
  });
  //定时器必须启动

 }

Widget::~Widget()
{
  delete ui;
}

void Widget::on_pushButton_clicked()
{
  t->start(1000);//启动定时器 1ms超时一次
}

void Widget::on_pushButton_2_clicked()
{
  t->stop();//停止定时器
}

```