

## 1 QLabel控件

### 1.1 设置标签的文本

### 1.2 设置标签的图片

### 1.3 设置标签的动态图

## 2 QLineEdit

### 2.1 设置QlineEdit内容和获取内容

### 2.1 设置行编辑密码模式

## 3 自定义控件

## 4 定时器

# 1 QLabel控件

## 1.1 设置标签的文本

```
1 #include "widget.h"
2
3 Widget::Widget(QWidget *parent)
4     : QWidget(parent)
5 {
6     this->resize(600,400);
7     label = new QLabel(this);
8     label->resize(200,200);
9     label->move(100,100);
10    // label->setText("我是一个标签");
11    label ->setText("<a href=\"https://www.baidu.com\">百度一下</a>");
12    label->setOpenExternalLinks(true); //设置点击链接自动打开
13 }
14
15 Widget::~Widget()
16 {
17
18 }
19
```

## 1.2 设置标签的图片

```
1 Widget::Widget(QWidget *parent)
2 : QWidget(parent)
3 {
4     this->resize(600,400);
5     label = new QLabel(this);
6     label->resize(100,100);
7     label->move(100,100);
8     // label->setText("我是一个标签");
9     label ->setText("<a href=\"https://www.baidu.com\">百度一下</a>");
10    label->setOpenExternalLinks(true);//设置点击链接自动打开
11
12
13    label_pic = new QLabel(this);
14    label_pic->resize(100,100);
15    label_pic->move(200,200);
16    label_pic->setPixmap(QPixmap("../Image/face.png"));
17    label_pic->setScaledContents(true);//设置自适应大小
18
19
20 }
21
```

## 1.3 设置标签的动态图

```
1 #include "widget.h"
2 #include <QPixmap>
3 #include <QMovie>
4 Widget::Widget(QWidget *parent)
5 : QWidget(parent)
6 {
7     this->resize(600,400);
8     label = new QLabel(this);
9     label->resize(100,100);
10    label->move(100,100);
11    // label->setText("我是一个标签");
12    label ->setText("<a href=\"https://www.baidu.com\">百度一下</a>");
13    label->setOpenExternalLinks(true);//设置点击链接自动打开
14
15
16    label_pic = new QLabel(this);
```

```

17  label_pic->resize(100,100);
18  label_pic->move(200,200);
19  label_pic->setPixmap(QPixmap("../Image/face.png"));
20  label_pic->setScaledContents(true); //设置自适应大小
21
22  label_move = new QLabel(this);
23  label_move->resize(100,100);
24  label_move->move(300,200);
25
26  QMovie * move = new QMovie("../Image/boy.gif");
27  label_move->setMovie(move);
28  label_move->setScaledContents(true);
29  move->start();
30
31
32
33  }
34
35  Widget::~Widget()
36  {
37
38  }
39

```

## 2 QLineEdit

### 2.1 设置QlineEdit内容和获取内容

```

1  #include "widget.h"
2  #include "ui_widget.h"
3  #include <QDebug>
4  Widget::Widget(QWidget *parent) :
5      QWidget(parent),
6      ui(new Ui::Widget)
7  {
8      ui->setupUi(this);
9      ui->lineEdit->setText("hello"); //设置行编辑内容
10 }
11
12 Widget::~Widget()
13 {
14     delete ui;

```

```

15 }
16
17 void Widget::on_lineEdit_returnPressed()
18 {
19     qDebug()<<ui->lineEdit->text();//获取行编辑内容
20 }
21

```

## 2.1 设置行编辑密码模式

```

1 #include "widget.h"
2 #include "ui_widget.h"
3 #include <QDebug>
4 Widget::Widget(QWidget *parent) :
5     QWidget(parent),
6     ui(new Ui::Widget)
7 {
8     ui->setUpUi(this);
9     ui->lineEdit->setEchoMode(QLineEdit::Password);//设置密码模式
10    ui->lineEdit->setTextMargins(100,0,0,0);//设置间距
11    ui->lineEdit->setText("hello");//设置行编辑内容
12 }
13
14 Widget::~Widget()
15 {
16     delete ui;
17 }
18
19 void Widget::on_lineEdit_returnPressed()
20 {
21     qDebug()<<ui->lineEdit->text();//获取行编辑内容
22 }
23

```

## 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，实现了一个自定义窗口，

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

signals:

public slots:

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

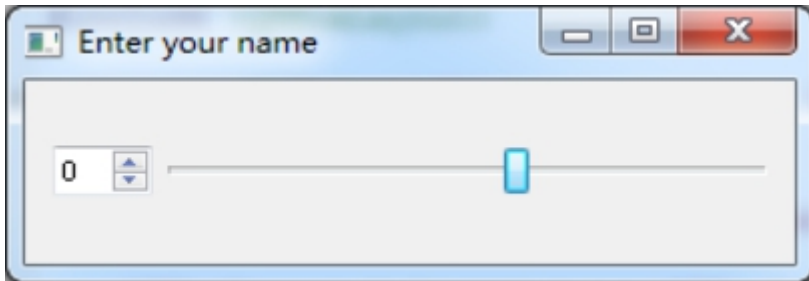
// smallwidget.cpp
SmallWidget::SmallWidget(QWidget *parent) : QWidget(parent)
{
    spin = new QSpinBox(this);
    slider = new QSlider(Qt::Horizontal, this);

    // 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);

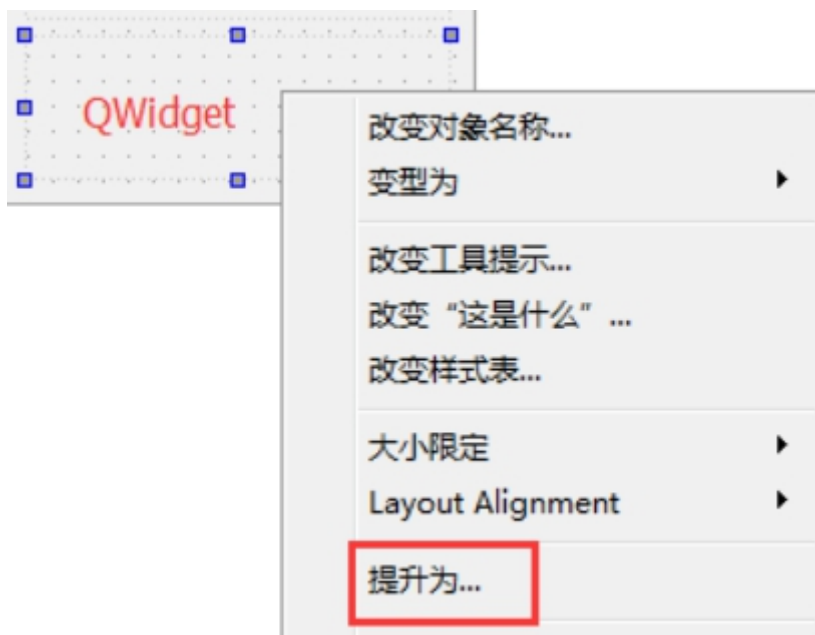
    // 添加消息响应
```

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

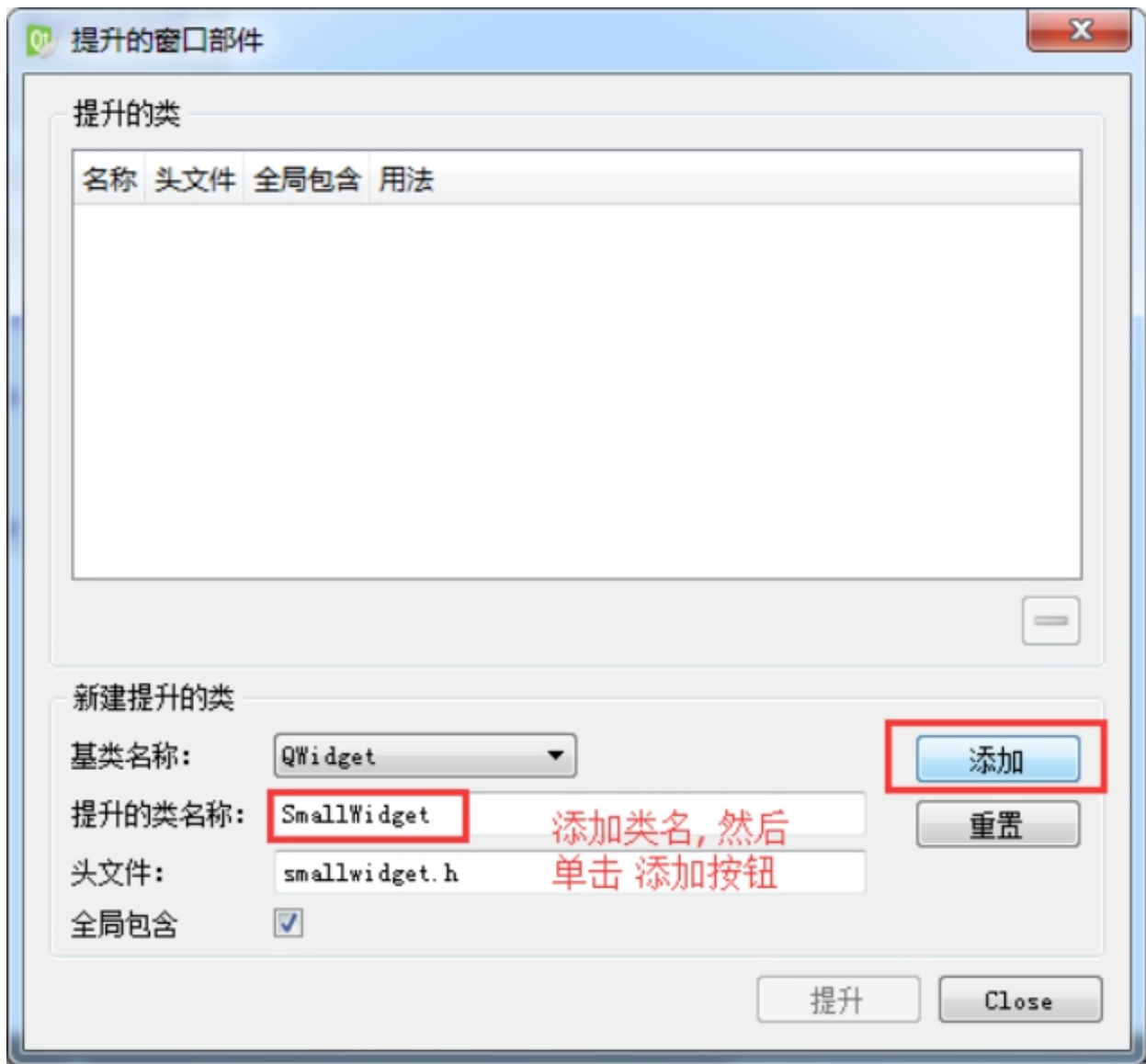
connect(slider, &QSlider::valueChanged,
spin, &QSpinBox::setValue);
}
```



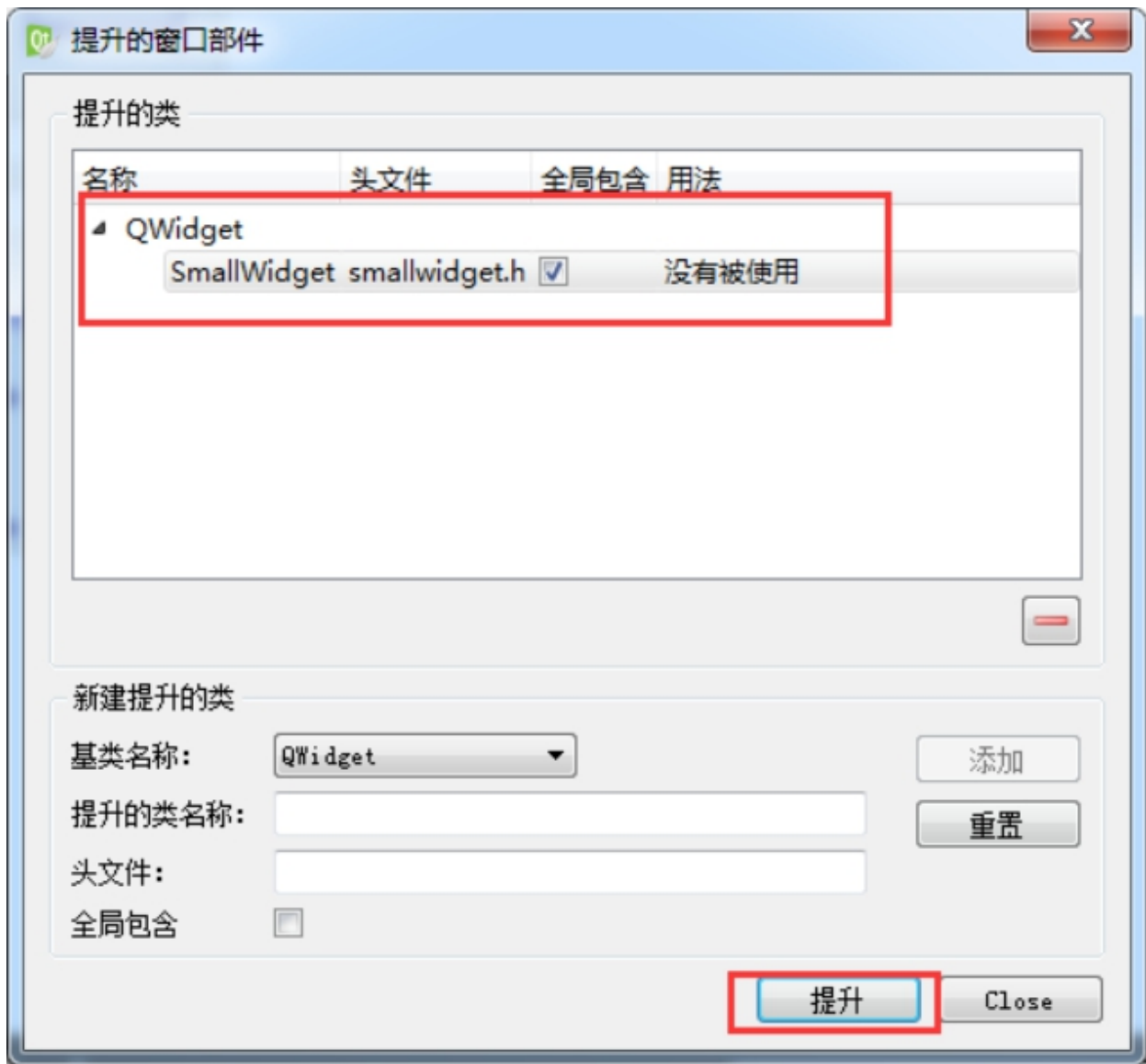
那么这个SmallWidget可以作为独立的窗口显示,也可以作为一个控件来使用:  
打开Qt的.ui文件,因为SmallWidget是派生自QWidget类,所以需要在ui文件中先  
放入一个QWidget控件,然后再上边鼠标右键



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

```
1 #include "widget.h"
2 #include "ui_widget.h"
3 #include <QDebug>
```



```
4 Widget::Widget(QWidget *parent) :
5     QWidget(parent),
6     ui(new Ui::Widget)
7 {
8     ui->setupUi(this);
9     t = new QTimer;
10    //设置定时器的超时事件 如果超时 会发出一个超时信号
11    connect(t,&QTimer::timeout,this,[](){
12
13        qDebug()<<"timeout";
14    });
15    //定时器必须启动
16
17 }
18
19 Widget::~Widget()
20 {
21     delete ui;
22 }
23
24 void Widget::on_pushButton_clicked()
25 {
26     t->start(1000); //启动定时器 1ms超时一次
27 }
28
29 void Widget::on_pushButton_2_clicked()
30 {
31     t->stop(); //停止定时器
32 }
33
```

