QT第二天

一、QPushbutton类

该类提供了一个按钮，通过按钮可以有对应的信号处理，对按钮常用的函数有

```
1>  设置文本：btn1->setText("按钮1");
2>  获取文本：btn1->text();
3>  设置父组件：btn1->setParent(this);
4>  设置大小：btn1->resize(100, 40);
5>  设置是否可用：btn1->setEnabled(false);
6>  移动位置：btn2->move(0, 40);
7>  设置图标：  btn3->setIcon(QIcon("D:/22061QT/day2/03Login/icon/cancel.png"));

```

```
#include "widget.h"
#include "ui_widget.h"

Widget::Widget(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::Widget)
{
    ui->setupUi(this);

    //QPushButton btn1(this);         //在栈区申请一个按钮
                                      //栈区申请的空间，会很快结束

    QPushButton *btn1 = new QPushButton;       //调用无参构造
    btn1->setText("按钮1");              //设置按钮文本
    btn1->setParent(this);             //设置父组件
    btn1->resize(100, 40);              //设置按钮大小
    btn1->setEnabled(false);             //设置按钮不可用状态

    //定义按钮，使用有参构造函数
    QPushButton *btn2 = new QPushButton(this);
    btn2->setText("按钮2");             //设置文本信息
    btn2->resize(btn1->size());         //设置大小为其他按钮的大小
    //btn2->move(0, 40);
    btn2->move(0, btn1->height());      //移动按钮位置，以其他按钮作为标准
```

```
25
26      //定义按钮，并初始化文本信息
27      QPushButton *btn3 = new QPushButton("按钮3", this);
28      btn3->resize(btn1->size());         //设置大小
29      btn3->move(btn1->width(), 0);         //移动按钮
30      btn3->setIcon(QIcon("D:/22061QT/day2/03Login/icon/cancel.png"));
        //设置按钮的图标
31
32      //定义按钮，初始化图标
33      QPushButton *btn4 = new QPushButton(QIcon("D:/22061QT/day2/03Login/icon/cancel.png"),
34                                          "按钮4",
35                                          this);
36      btn4->resize(btn1->size());
37      btn4->move(btn1->width(), btn1->height());
38
39      //定义一个标签
40      QLabel *lab = new QLabel("爱好：", this);
41      lab->move(200,200);
42
43  }
44
45  Widget::~Widget()
46  {
47      delete ui;
48  }
49
```

二、信号与槽机制

```
1   1. 信号函数和槽函数是QT类中特有的类的成员函数，而标准C++类中是没有信号函数和槽函数的。
2       因此需要使用moc.exe工具将QT类中的信号函数和槽函数转换为标准的C++代码。
3
4       带有信号函数和槽函数的类书写格式:
5       class 类名{
6           Q_OBJECT
7           signals:
8               信号函数声明
9           public slots:
10              参函数的声明或定义
11      };
```

```
2. 信号函数和槽函数的主要作用
    主要是用来完成QT工程中两个组件之间实现通信的一种特殊的机制。
    即一个组件发出一个信号函数，这个信号函数就会被连接到一个槽函数上，
    槽函数就会被执行，槽函数中就是编写的代码的逻辑。

3. QT类中有定义号的信号函数和槽函数
    也可以在类中自定义信号函数和槽函数
4. 信号函数是一个不完整的函数，只有声明，没有定义
   槽函数是一个完整的函数，既有声明也有定义

5. 信号函数和槽函数，需要使用共有的成员方法进行连接后，才能起作用
```

2.1 qt4版本的连接（不安全）

```
connect(const QObject *sender, const char *signal, const char *method)  //qt版本连接
    //参数1：发射信号的组件
    //参数2：发射的信号函数
    //参数3：信号的接受者组件
    //参数4：接受后的处理函数，槽函数
    使用该连接时必须用 SIGNAL() 将信号函数转变成const char*使用
    使用SLOT()将槽函数转变成const char*使用
    举个例子：
        QLabel *label = new QLabel;
        QScrollBar *scrollBar = new QScrollBar;
        QObject::connect(scrollBar, SIGNAL(valueChanged(int)),
                         label,  SLOT(setNum(int)));

     即使写的信号函数和槽函数不正确，也不会报错，这就使得后期调试代码容易出问题
```

2.2 qt5版本的连接

```
connect(const QObject *sender, PointerToMemberFunction signal,
const QObject *receiver, PointerToMemberFunction method)
//参数1：信号发射者的组件
//参数2：发射的信号
//参数3：信号接收者组件
//参数4：槽函数
//信号函数和槽函数，用的都是函数的指针
    举个例子：
            QLabel *label = new QLabel;
            QLineEdit *lineEdit = new QLineEdit;
            QObject::connect(lineEdit,
&QLineEdit::textChanged,
                    label,  &QLabel::setText);
该连接是安全的连接，但凡有一点写错，直接报错
```

2.3 使用Larmda表达式作为槽函数使用

```
connect(const QObject *sender, PointerToMemberFunction signal,
Functor functor)
//参数1:信号发射的组件
//参数2：发射的信号的指针
//参数3：仿函数，可以是一个Larmda表达式
举个例子：
    QByteArray page = ...;
  QTcpSocket *socket = new QTcpSocket;
  socket->connectToHost("qt-project.org", 80);
  QObject::connect(socket, &QTcpSocket::connected, [=] () {
          socket->write("GET " + page + "\r\n");
      });
```

2.4 案例

```
.pro文件
QT       += core gui texttospeech

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++11

# The following define makes your compiler emit warnings if
you use
# any Qt feature that has been marked deprecated (the exact
warnings
```

```
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS

# You can also make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to a certain version of Qt.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \
    main.cpp \
    widget.cpp

HEADERS += \
    widget.h

FORMS += \
    widget.ui

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target
```

头文件：

```
#ifndef WIDGET_H
#define WIDGET_H

#include <QWidget>
#include<QPushButton>
#include <QTextToSpeech>      //引入文本转语音的头文件

QT_BEGIN_NAMESPACE
namespace Ui { class Widget; }
QT_END_NAMESPACE

class Widget : public QWidget
{
    Q_OBJECT
```

```
49
50  public slots:
51      void close();             //声明自定义的槽函数
52
53      void read();              //声明阅读函数
54
55  public:
56      Widget(QWidget *parent = nullptr);
57      ~Widget();
58
59  private:
60      Ui::Widget *ui;
61
62      //定义一个按钮
63      QPushButton *btn1;
64
65      //定义一个按钮
66      QPushButton *btn2;
67
68      //定义按钮3
69      QPushButton *btn3;
70
71      //实例化一个播报者
72      QTextToSpeech speecher;
73  };
74  #endif // WIDGET_H
75  ----------------------------------------------------------------
    ---------------------------------
```

源文件

```
76  #include "widget.h"
77  #include "ui_widget.h"
78
79  Widget::Widget(QWidget *parent)
80      : QWidget(parent)
81      , ui(new Ui::Widget)
82  {
83      ui->setupUi(this);
84
85      //给按钮申请空间
86      btn1 = new QPushButton("按钮1", this);
87
88      //当按钮一旦被按下，就会触发一个clicked的信号，我们可以将该信号，连
    接到对应的槽函数中，处理相关逻辑
89      //通过connect函数进行连接
90      //测试qt4版本的连接函数
91
```

```
92      connect(btn1, SIGNAL(clicked()), this, SLOT(close()));
    //不安全的连接
93
94      //给按钮2申请空间
95      btn2 = new QPushButton("按钮2", this);
96      btn2->move(btn1->width(), 0);            //移动按钮
97      //当按钮一旦被按下，就会触发一个clicked的信号，我们可以将该信号，连
    接到对应的槽函数中，处理相关逻辑
98      //测试qt5版本的连接函数
99      connect(btn2, &QPushButton::clicked, this,
    &Widget::read);  //安全的连接
100
101     //给btn3申请空间
102     btn3 = new QPushButton("按钮3", this);
103     btn3->move(0, btn1->height());
104     //测试使用Larmda表达式作为槽函数
105     connect(btn3, &QPushButton::clicked, [&](){
106         this->resize(500,400);
107     });
108
109
110 }
111
112 Widget::~Widget()
113 {
114     delete ui;
115 }
116
117 //自定义的槽函数
118 void Widget::close()
119 {
120     this->close();
121 }
122
123 //自定义阅读的槽函数
124 void Widget::read()
125 {
126     speecher.say(QString("今天是周二，天气晴，心情不错，又学到新知识
    了！！！"));
127 }
128
129
```

2.5 ui界面上的信号与槽连接方式一

1> 选中ui界面上的组件

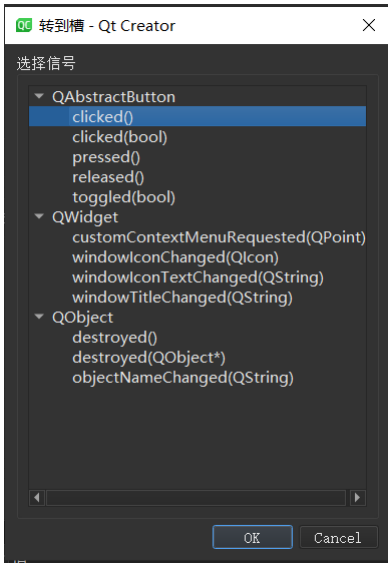
2> 在状态栏中选中signals _slots Edit

3> 按+号，新增一组连接

4> 选中信号发送者、发射的信号、信号接受者、对应的槽函数

5> 注意：该方法只能使用系统提供的信号函数和槽函数

2.6 ui界面上的信号与槽连接方式二



1> 在ui界面上，选中该组件
2> 右键转到槽
3> 选中对应的信号函数
4> 在槽函数中编写对应的逻辑代码即可

2.6 断开连接

```
qt4版本的断开
    disconnect(const QObject *sender,       //信号发射者
               const char *signal,         //发射的信号需要用SIGNAL()转换
               const QObject *receiver,    //信号接收者
               const char *method)         //槽函数   需要用SLOTS()转换

qt5版本的断开
    disconnect(const QObject *sender,                //信号发射者
               PointerToMemberFunction signal,  //发射的信号
               const QObject *receiver,          //信号接收者
               PointerToMemberFunction method)   //槽函数
```

```cpp

//断开1按钮对应的槽函数
void Widget::on_disconnectBtn1_clicked()
{
    //qt4版本
    disconnect(btn1, SIGNAL(clicked()), this, SLOT(close()));
  //将btn1与槽函数断开连接
}
//断开2按钮对应的槽函数
void Widget::on_disconnectBtn2_clicked()
{
    //qt5版本断开连接
    disconnect(btn2, &QPushButton::clicked, this,
&Widget::read);  //将btn2与槽函数断开连接
}
```

2.7 发射自定义的信号

1> 在类中的signals权限下，声明信号函数
2> 在程序适当位置，将自定义的信号发射：emit 函数名(实参表);
3> 将自定义的信号函数与槽函数进行连接
4> 当程序执行到,发射信号语句后，就会执行连接的槽函数

例如工程2中，btn6按钮

```cpp
QT_END_NAMESPACE

class Widget : public QWidget
{
    Q_OBJECT

signals:
    void sig1(QString mes);

public slots:
    void close();              //声明自定义的槽函数

    void read();               //声明阅读函数
```

声明自定义信号函数

//自定义信号函数，只有声明没有定义

```cpp

//按钮6对应的槽函数
void Widget::on_btn6_clicked()
{
    qDebug()<<"11111111111111111111111111";

    //发射一个自定义的信号
    emit sig1("hello world");

}
```

在程序的某个部分，发射自定义的信号函数

```
//将自定义的信号函数与自定义的槽函数进行连接
connect(this, &Widget::sig1, this, &Widget::on_widget_sig1);
```

2.8 信号函数与槽函数的总结

```
1> 信号函数和槽函数连接时，一般要求参数的类型和个数一致
    connect(发送者,SIGNAL(signalFunc()),
                接收者, SLOT(slotFunc())); // OK
    connect(发送者,SIGNAL(signalFunc(int)),
                接收者, SLOT(slotFunc(int))); // OK
    connect(发送者,SIGNAL(signalFunc(int,char)),
                接收者, SLOT(slotFunc(int,char))); // OK
    connect(发送者,SIGNAL(signalFunc(int,char)),
                接收者, SLOT(slotFunc(char,int))); // error
    connect(发送者,SIGNAL(signalFunc(int)),
                接收者, SLOT(slotFunc(char))); // error
    2. 信号函数和槽函数的参数个数不一致
        信号函数的参数个数 大于 槽函数的参数个数:
        connect(发送者,SIGNAL(signalFunc(int,char)),
                接收者, SLOT(slotFunc(int))); // OK
        connect(发送者,SIGNAL(signalFunc(int,char)),
                接收者, SLOT(slotFunc(char))); // error
        槽函数的参数个数 大于 信号函数的参数个数:
        connect(发送者,SIGNAL(signalFunc(int)),
                接收者, SLOT(slotFunc(int,char))); // error
        connect(发送者,SIGNAL(signalFunc(int)),
                接收者, SLOT(slotFunc(int,char = 缺省值))); //
OK
    3. 同一个信号函数可以连接到多个槽函数
        connect(发送者,SIGNAL(signalFunc(int)),
                接收者, SLOT(slotFunc1(int))); // OK
        connect(发送者,SIGNAL(signalFunc(int)),
                接收者, SLOT(slotFunc2(int))); // OK
    4. 多个信号函数可以连接到同一个槽函数
        connect(发送者,SIGNAL(signalFunc1(int)),
                接收者, SLOT(slotFunc(int))); // OK
        connect(发送者,SIGNAL(signalFunc2(int)),
                接收者, SLOT(slotFunc(int))); // OK
```

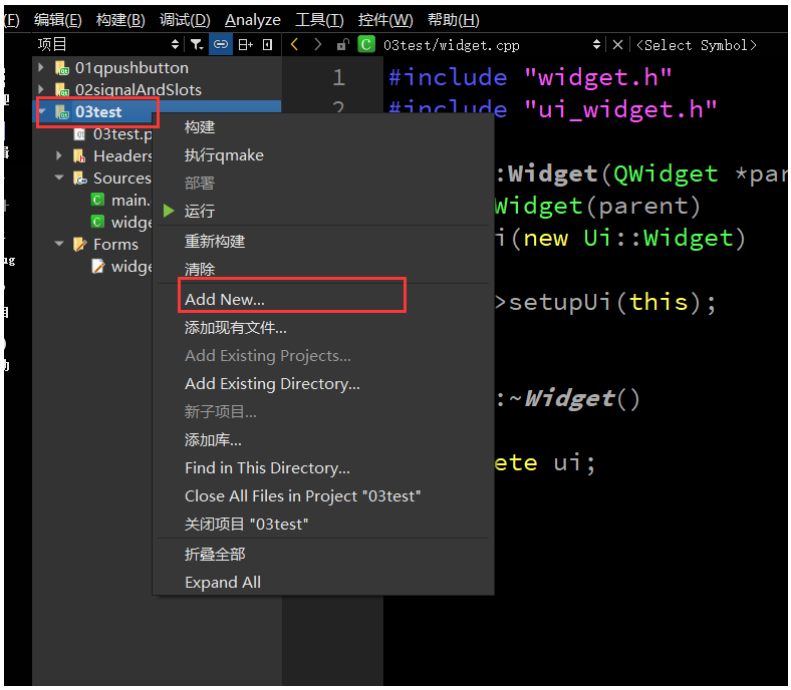
```
#include "widget.h"
#include "ui_widget.h"


```

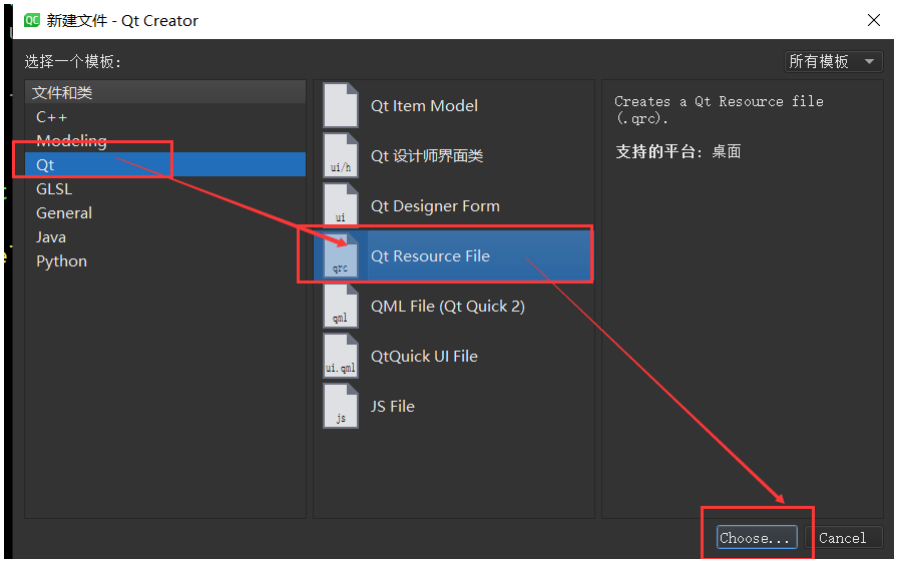
```
5
6  Widget::Widget(QWidget *parent)
7      : QWidget(parent)
8      , ui(new Ui::Widget)
9  {
10     ui->setupUi(this);
11
12     //同一信号连接两个槽函数
13     connect(ui->pushButton, &QPushButton::clicked, this,
   &Widget::on_pushButton_slot1);
14     connect(ui->pushButton, &QPushButton::clicked, this,
   &Widget::on_pushButton_slot2);
15
16     //多个信号连接同一个槽函数
17     connect(ui->btn2, &QPushButton::clicked, this,
   &Widget::on_pushButton_slot1);
18
19 }
20
21 Widget::~Widget()
22 {
23     delete ui;
24 }
25
26
27 void Widget::on_pushButton_slot1()
28 {
29     ui->Lab1->setText("hello world");
30 }
31
32 void Widget::on_pushButton_slot2()
33 {
34     ui->Lab2->setNum(520);
35 }
36
```

三、加载资源文件

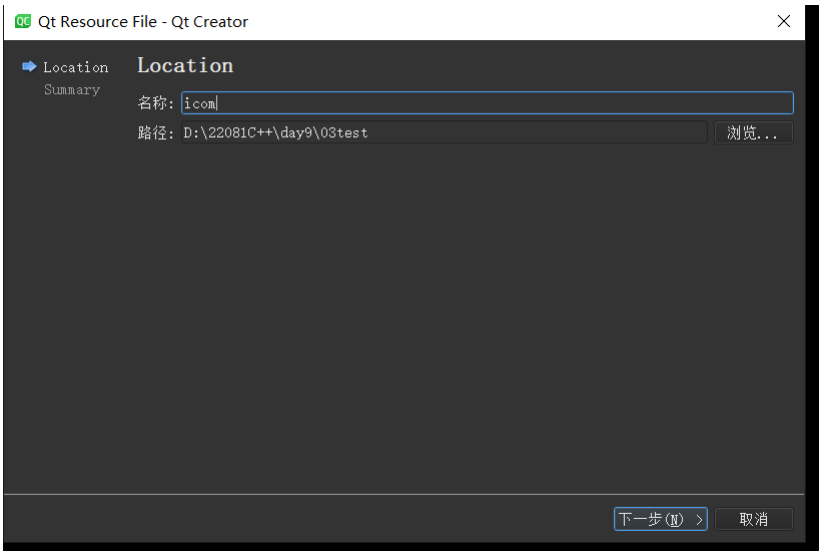
1> 点击项目名称，右键，add New



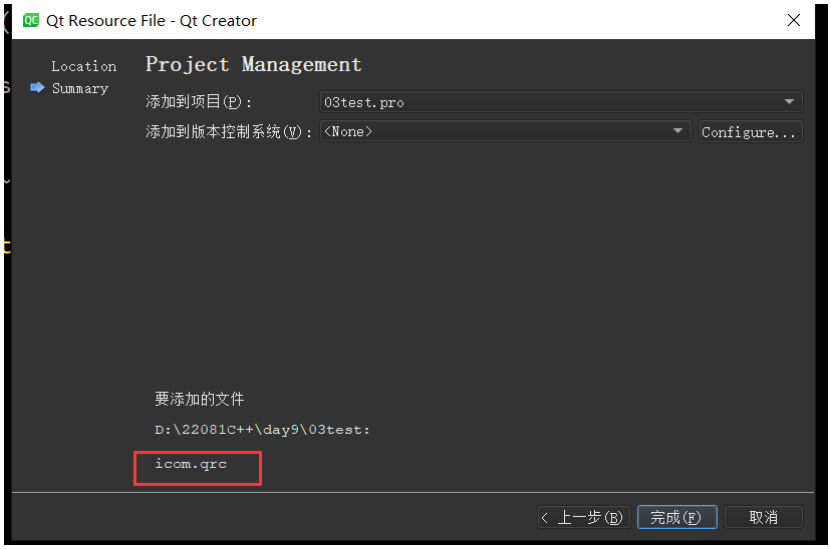
2> 选中qt ---> 资源文件 --->选中



3> 起个名字

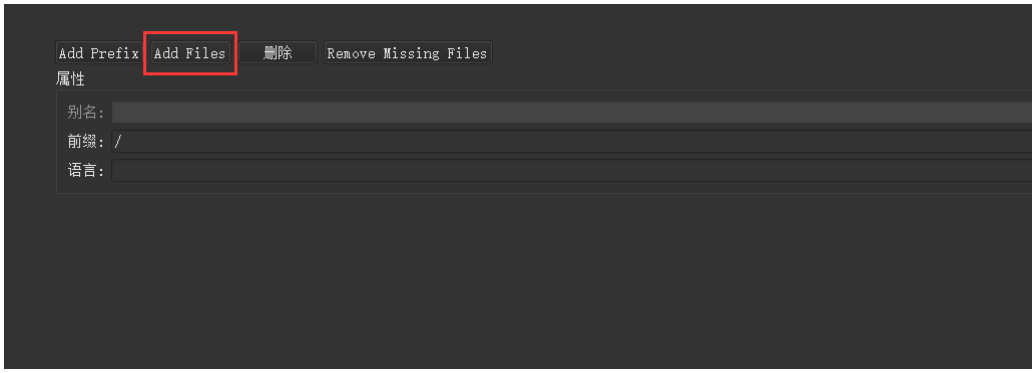


4> 完成

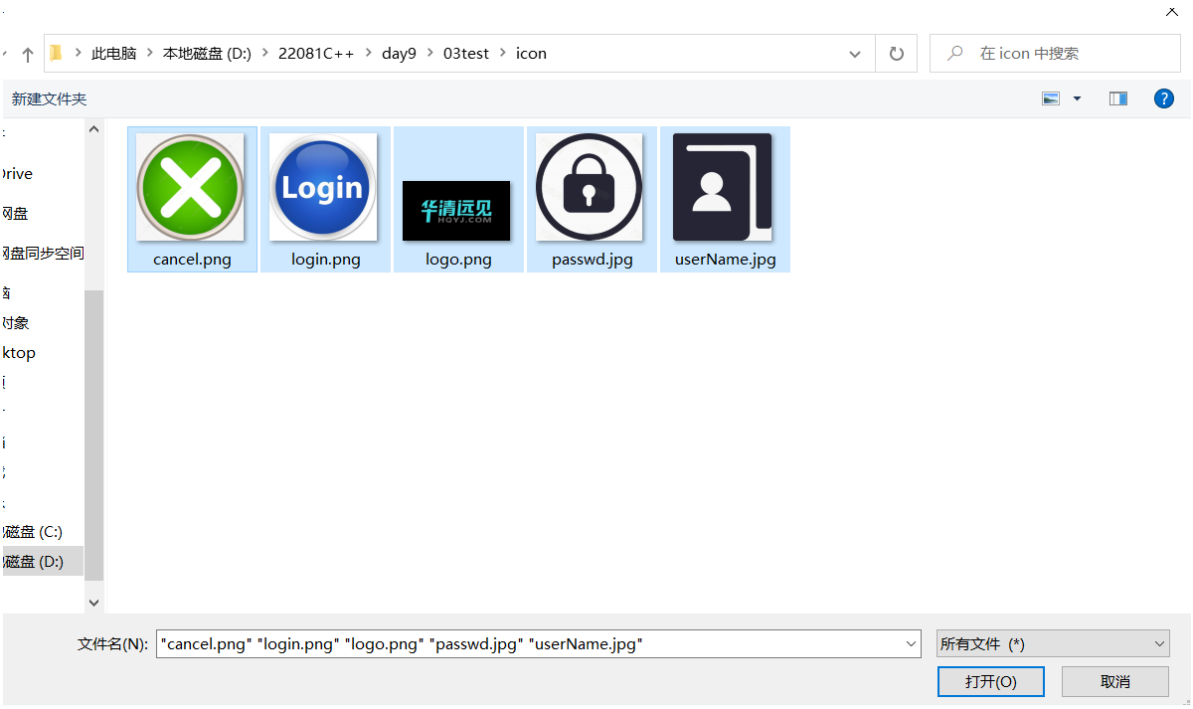


5> 添加前缀

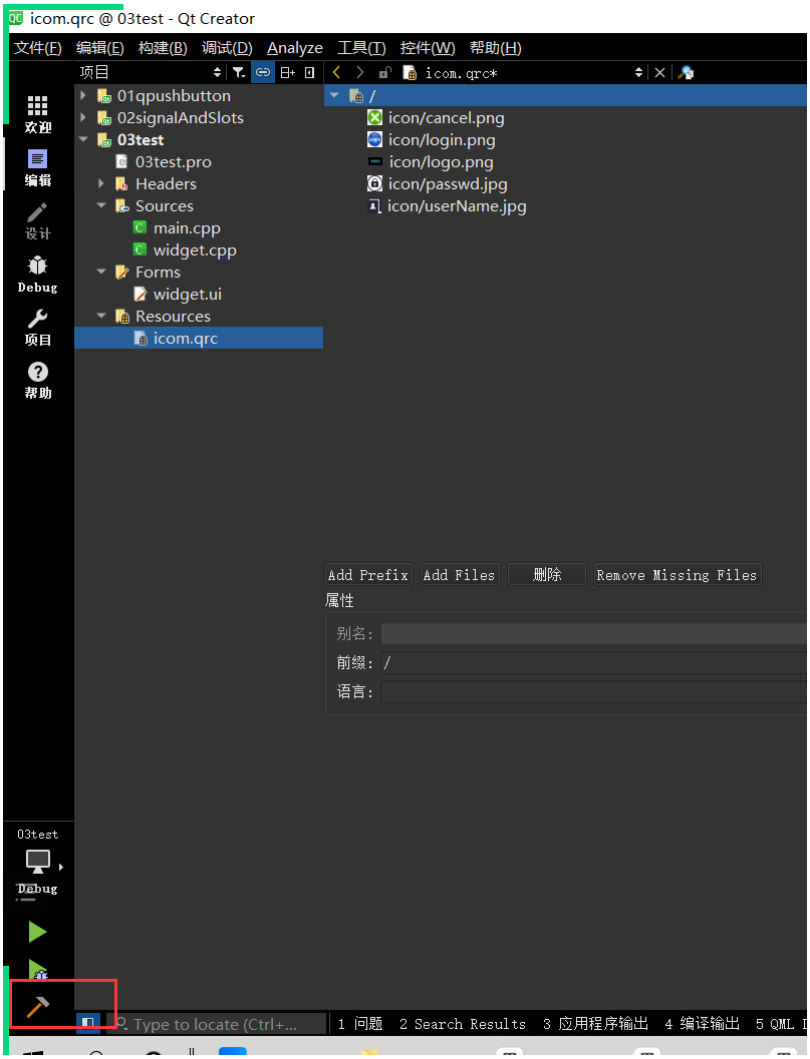
6> 添加文件



7> 选中要添加的资源



8> 点击锤子进行保存



作业：

将登录框界面重新完成一下，使用手动连接信号与槽，ui界面可以在用拖拽的形式，但是，信号与槽函数连接不要使用右键转到槽

要求：登录按钮连接槽函数使用qt4版本的连接，取消按钮连接草韩式使用qt5版本的连接