

知识点1【容器的布局】

知识点2【组容器】

知识点3【QListWidget】列表控件

知识点4【QTreeWidget】树控件1-2

知识点5【表格控件QTableWidget】

2-1

知识点6【下拉列表框】

知识点7【QLabel控件的使用】

1、Qlabel设置文本

2、设置图片

3、设置动画

知识点8【自定义控件】

1、定义一个自定义控件

2、给自己的ui文件 添加常用控件

3、在其他ui文件中使用 自定义控件MyWidget

4、改变spinbox的值 进度条移动

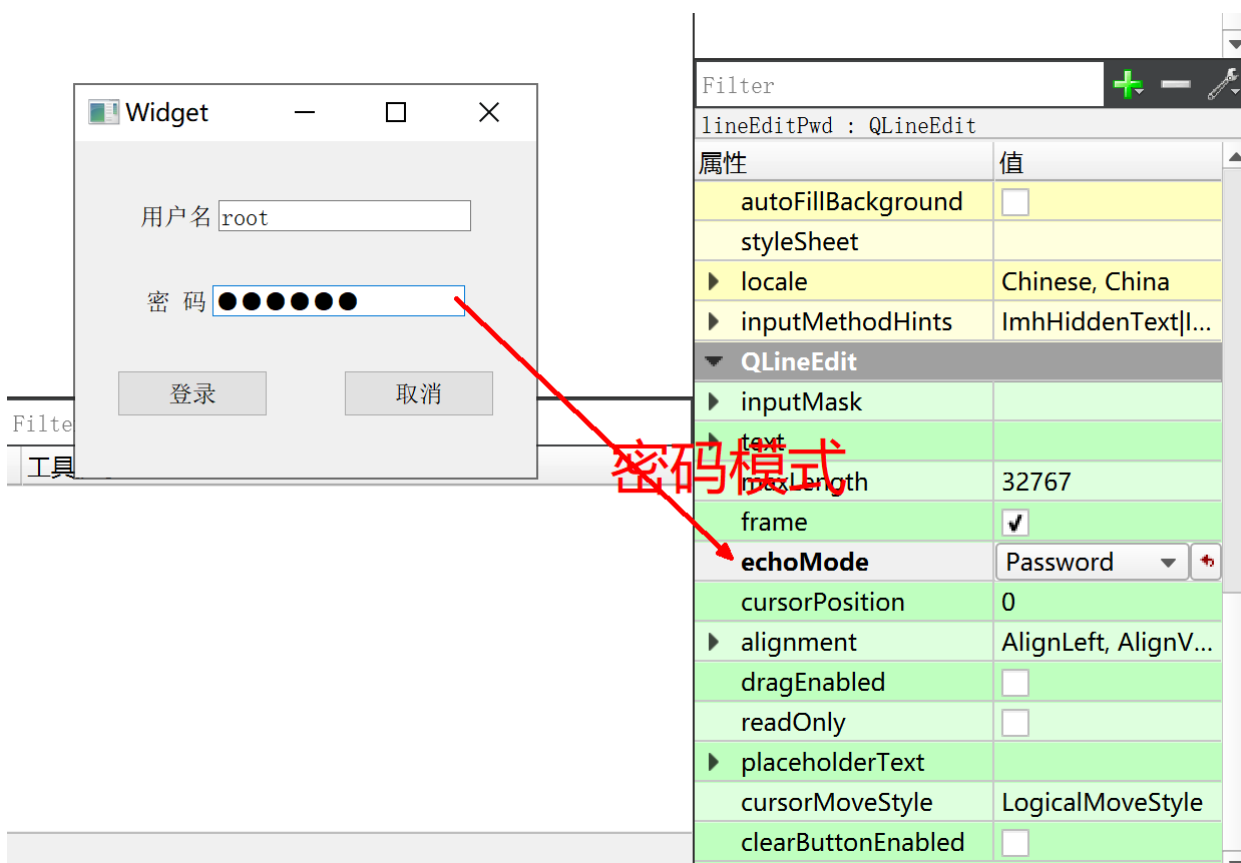
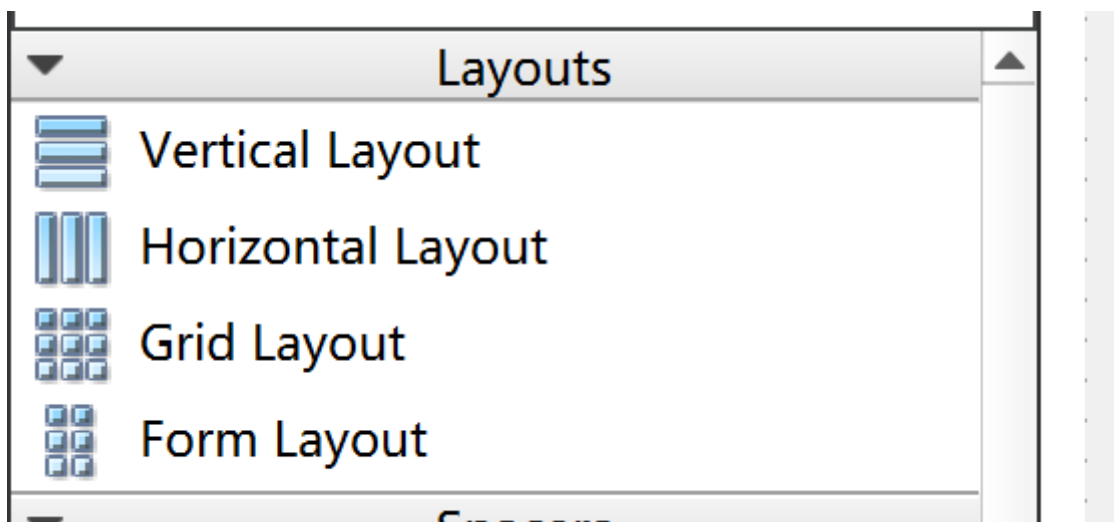
5、//拖动 slider 更改spinbox

spinbox槽函数

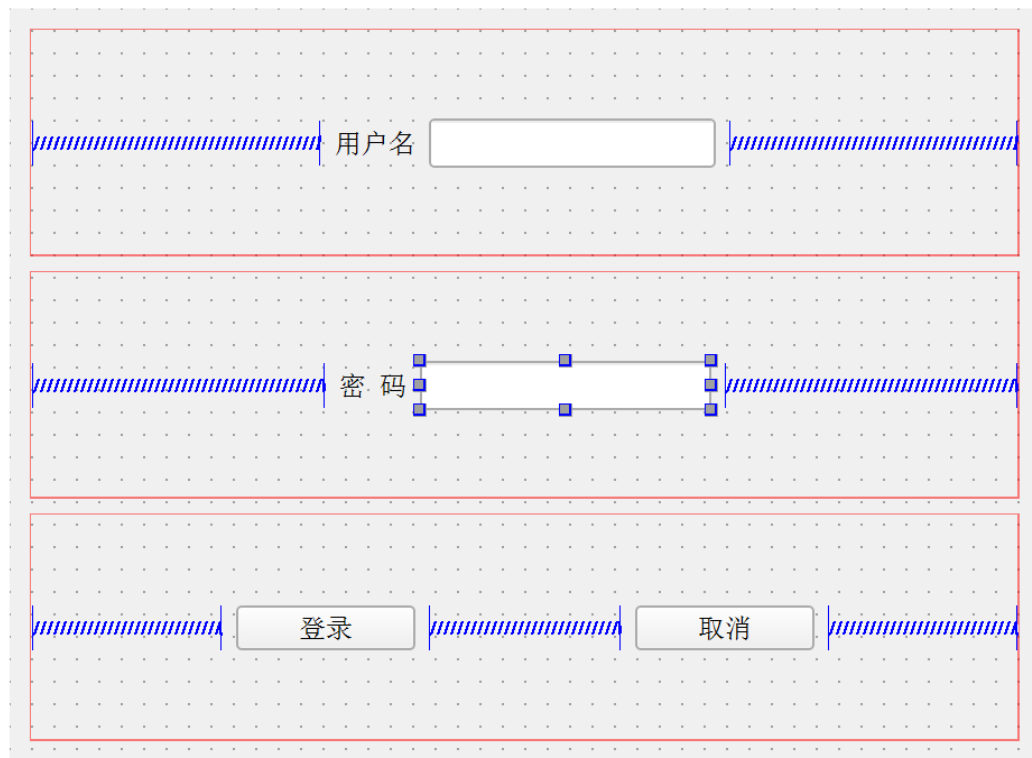
知识点1【容器的布局】

Qt 提供的布局中以下三种是我们最常用的：

- QHBoxLayout：按照水平方向从左到右布局；
- QVBoxLayout：按照竖直方向从上到下布局；
- QGridLayout：在一个网格中进行布局，类似于 HTML 的 table；



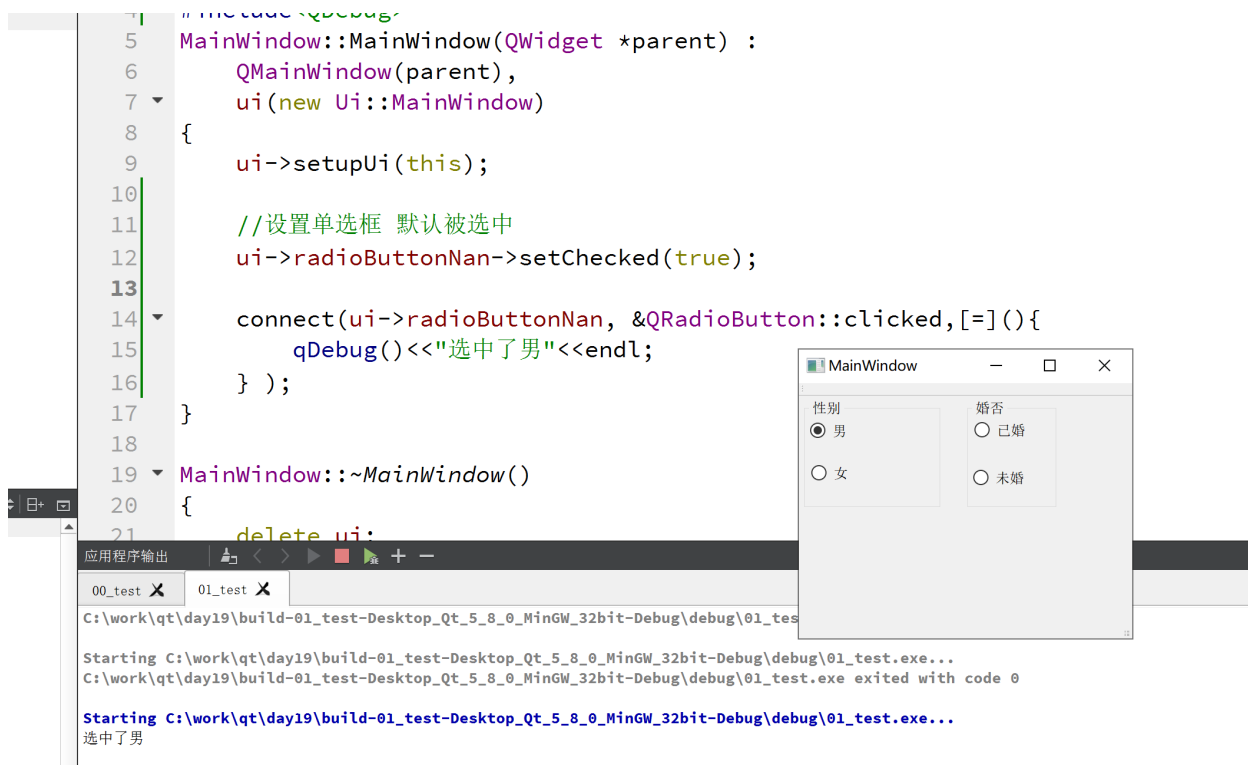
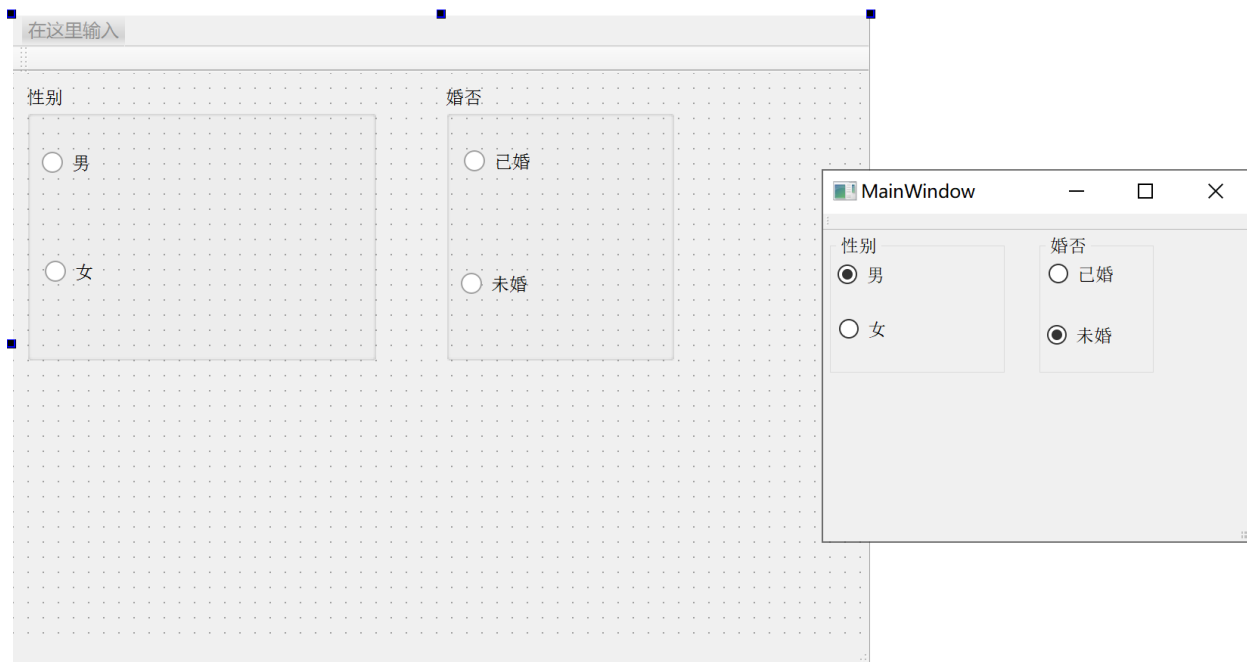
ui文件布局：



动起来：

```
1  Widget::Widget(QWidget *parent) :  
2      QWidget(parent),  
3      ui(new Ui::Widget)  
4  {  
5      ui->setupUi(this);  
6      //单击登录 获取用户输入的用户名和密码  
7      connect(ui->pushButton, &QPushButton::clicked, [=]() {  
8          //获取用户名  
9          QString user = ui->lineEditUser->text();  
10         QString pwd = ui->lineEditPwd->text();  
11  
12         qDebug() << "用户名:" << user << ", 密码: " << pwd << endl;  
13     } );  
14 }
```

知识点2 【组容器】



知识点3 【QListWidget】 列表控件

```
QListWidget(QWidget *parent = Q_NULLPTR)
```

```
~QListWidget()
```

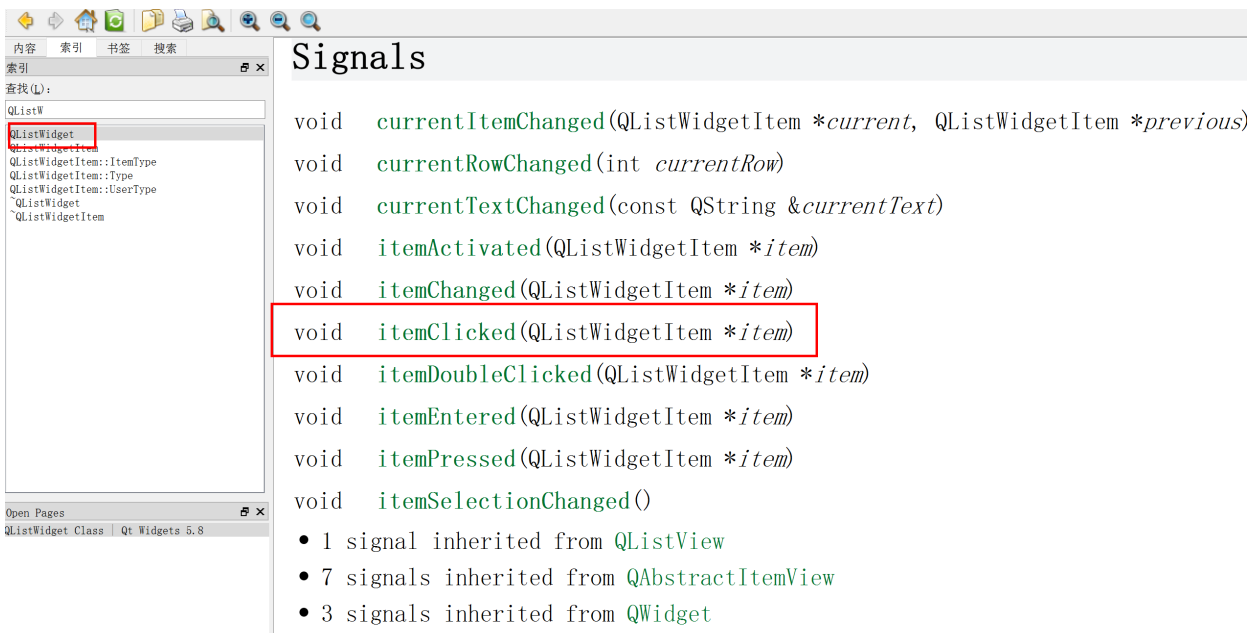
```
void addItem(const QString &label)
```

```
void addItem(QListWidgetItem *item)
```

```
void addItems(const QStringList &labels)
```

```
void closePersistentEditor(QListWidgetItem *item)
```

```
int count() const
```



Signals

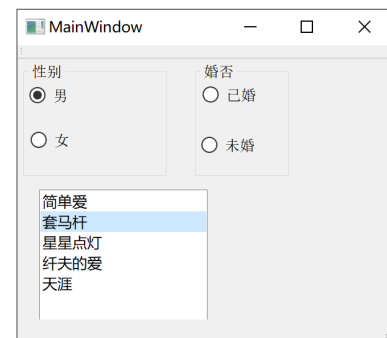
- void currentItemChanged(QListWidgetItem *current, QListWidgetItem *previous)
- void currentRowChanged(int currentRow)
- void currentTextChanged(const QString ¤tText)
- void itemActivated(QListWidgetItem *item)
- void itemChanged(QListWidgetItem *item)
- void itemClicked(QListWidgetItem *item)
- void itemDoubleClicked(QListWidgetItem *item)
- void itemEntered(QListWidgetItem *item)
- void itemPressed(QListWidgetItem *item)
- void itemSelectionChanged()

- 1 signal inherited from `QListView`
- 7 signals inherited from `QAbstractItemView`
- 3 signals inherited from `QWidget`

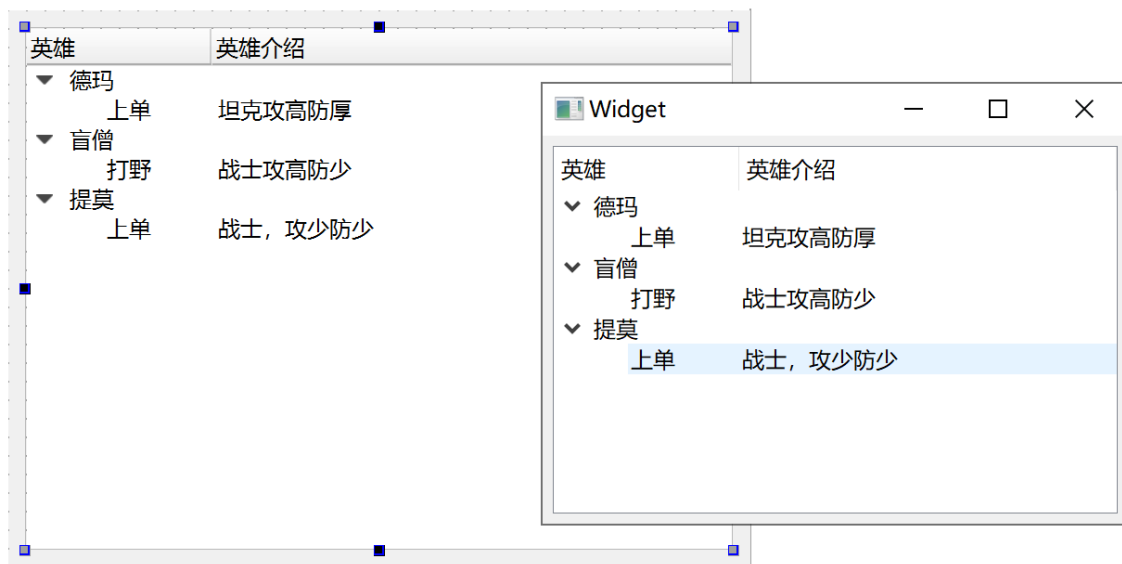
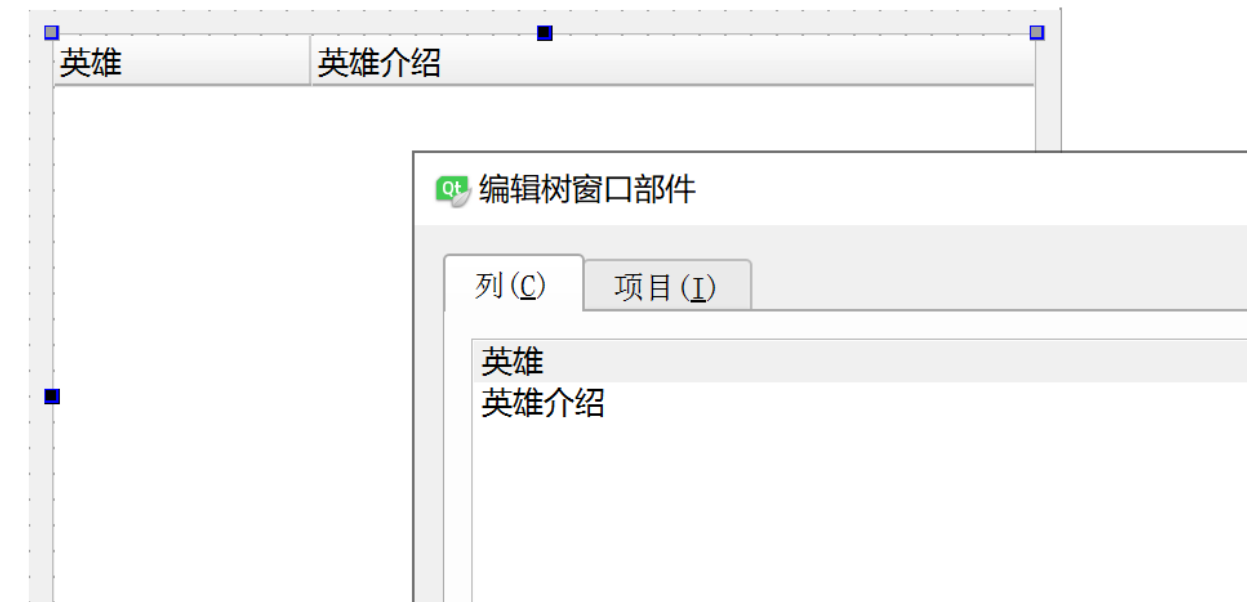
```
//列表控件
//往列表控件里面 添加 Item
QListWidgetItem *item = new QListWidgetItem("简单爱");
ui->listWidget->addItem(item);
```

```
//添加：静夜思的后四句
QStringList list;
list<<"套马杆"<<"星星点灯"<<"纤夫的爱"<<"天涯";
ui->listWidget->addItems(list);
```

```
connect(ui->listWidget, &QListWidget::itemClicked, [](QListWidgetItem *item){
    qDebug()<<item->text()<<endl;
} );
```



知识点4 【QTreeWidget】 树控件1-2



span)

设置树控件的头信息

void **setHeaderItem**(QTreeWidgetItem **item*)

void **setHeaderLabel**(const QString &*label*)

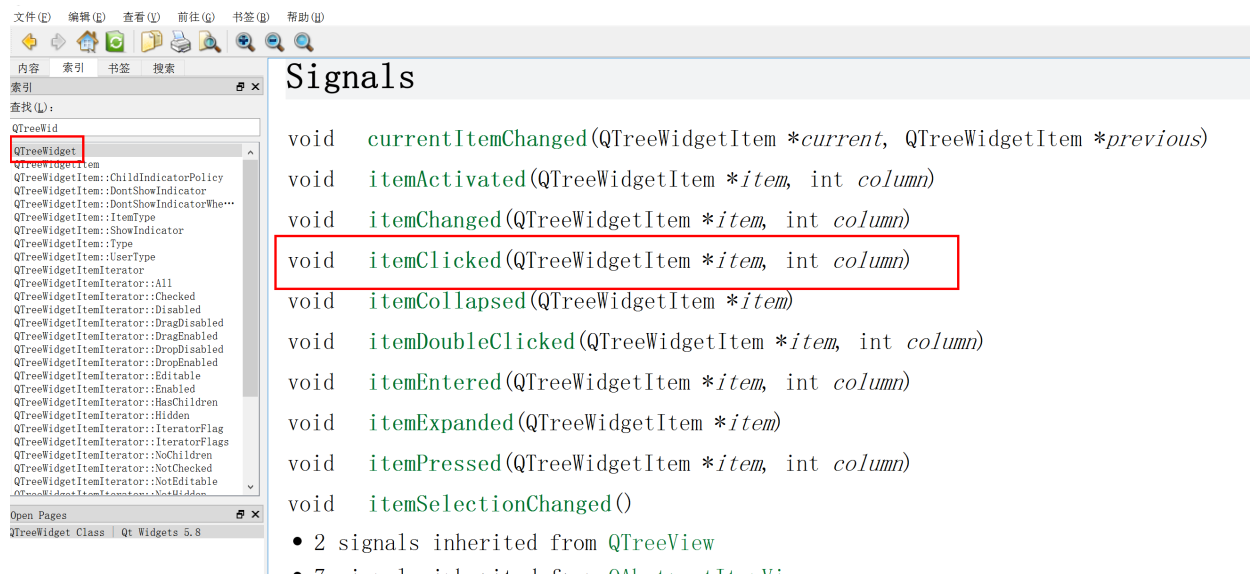
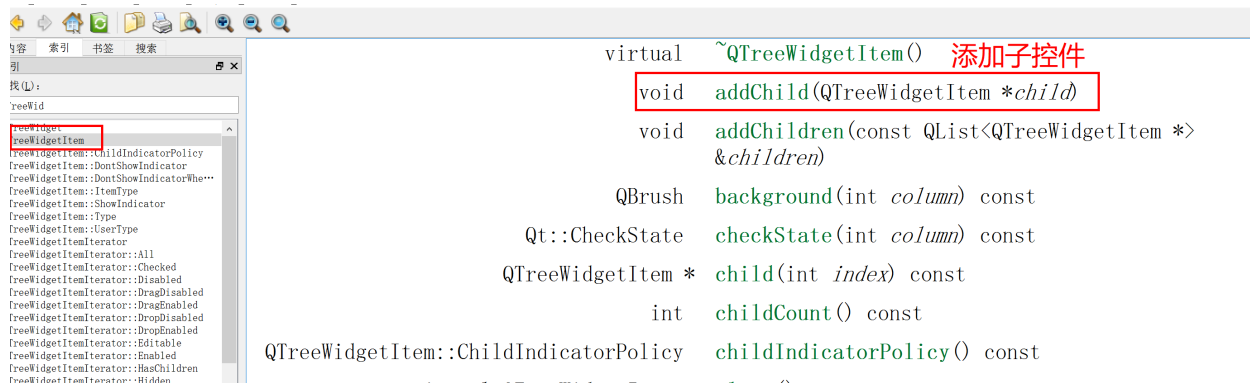
void **setHeaderLabels**(const QStringList &*labels*)

QTreeWidgetItem()

添加顶层控件

void **addTopLevelItem**(QTreeWidgetItem **item*)

void **addTopLevelItems**(const QList<QTreeWidgetItem *> &*items*)



QString text(int column) const

```

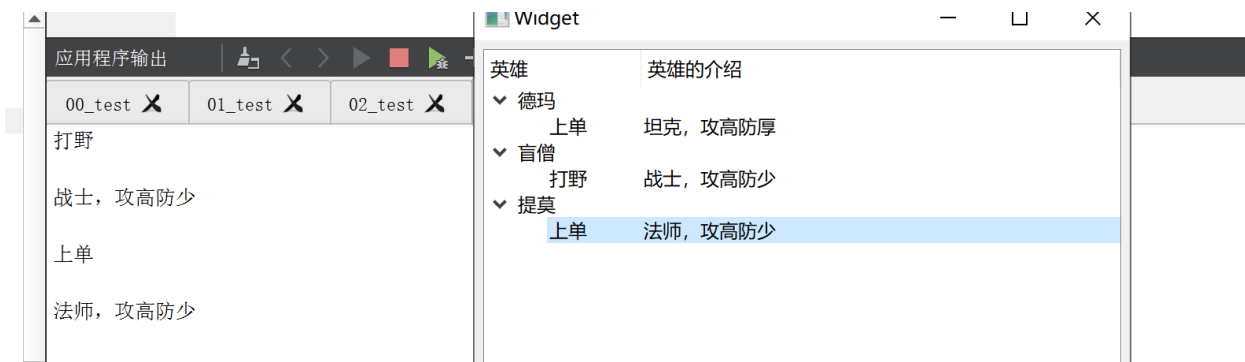
1 Widget::Widget(QWidget *parent) :
2     QWidget(parent),
3     ui(new Ui::Widget)
4 {
5     this->resize(800,600);
6     ui->setupUi(this);
7     // 设置树控件的头信息
8     QStringList list;
9     list<<"英雄"<<"英雄的介绍";
10    ui->treeWidget->setHeaderLabels(list);
11
12    // 添加顶层控件
13    QTreeWidgetItem *item1 = new QTreeWidgetItem(QStringList()<<"德玛");
14    ui->treeWidget->addTopLevelItem(item1);

```

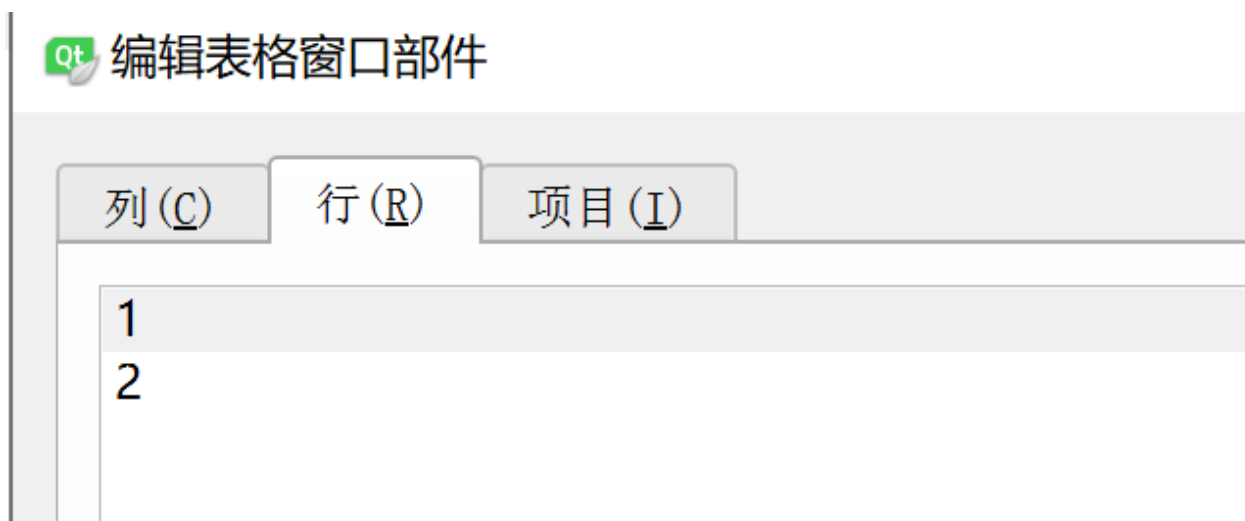
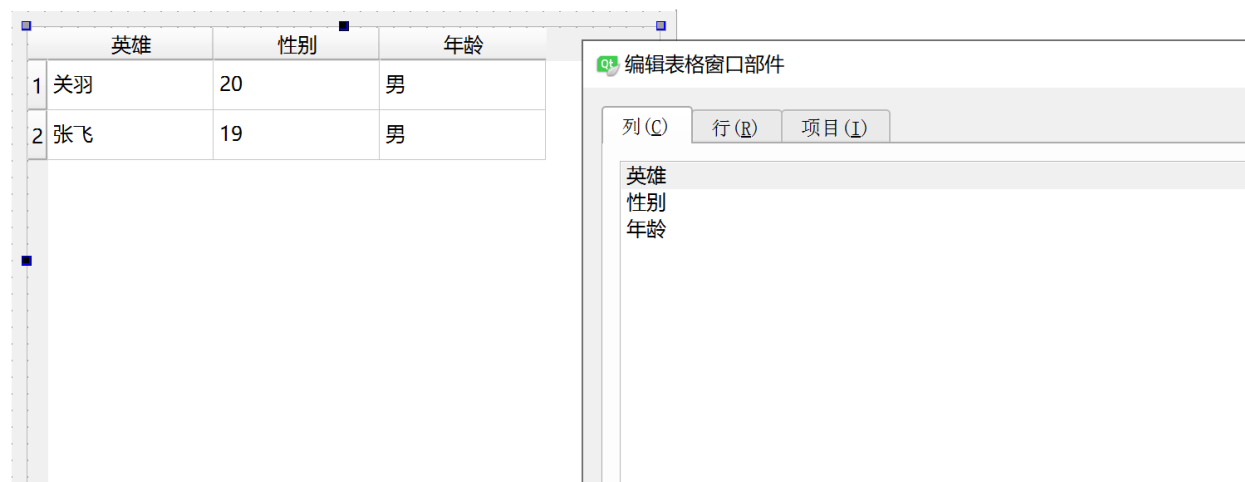
```

15 //QTreeWidgetItem *child = new QTreeWidgetItem(QStringList()<<"上单"<
<"坦克，攻高防厚");
16 //item1->addChild(child);
17 item1->addChild(new QTreeWidgetItem(QStringList()<<"上单"<<"坦克，攻高防
厚"));
18
19 QTreeWidgetItem *item2 = new QTreeWidgetItem(QStringList()<<"盲僧");
20 ui->treeWidget->addTopLevelItem(item2);
21 item2->addChild(new QTreeWidgetItem(QStringList()<<"打野"<<"战士，攻高防
少"));
22
23 QTreeWidgetItem *item3 = new QTreeWidgetItem(QStringList()<<"提莫");
24 ui->treeWidget->addTopLevelItem(item3);
25 item3->addChild(new QTreeWidgetItem(QStringList()<<"上单"<<"法师，攻高防
少"));
26
27 //动起来
28 connect(ui->treeWidget, &QTreeWidgetItem::itemClicked, [](QTreeWidgetItem *i
tem,int column){
29     qDebug()<< item->text(column).toUtf8().data() <<endl;
30 } );
31 }
32

```



知识点5 【表格控件QTableWidget】



```
void setCellWidget(int row, int column, QWidget  
void setColumnCount(int column) 设置表格的列数  
void setCurrentCell(int row, int column)  
void setCurrentCell(int row, int column,
```

```
void setRangeSelected(const QTableWidgetItemSe  
&range, bool select)  
void setRowCount(int rows)  
void setVerticalHeaderItem(int row, QTableWidgetItem  
*item) 设置表格的行数
```

设置表头信息

```
QItemSelectionModel::SelectionFlags command)  
void setHorizontalHeaderItem(int column,  
QTableWidgetItem *item)  
void setHorizontalHeaderLabels(const QStringList  
&labels) 设置水平表头信息  
void setItem(int row, int column, QTableWidgetItem *item)  
void setItemPrototype(const QTableWidgetItem *item)  
void setRangeSelected(const QTableWidgetItemSelectionRange  
&range, bool select)  
void setRowCount(int rows)  
void setVerticalHeaderItem(int row, QTableWidgetItem  
*item) 设置垂直的表头信息  
void setVerticalHeaderLabels(const QStringList &labels)  
void sortItems(int column, Qt::SortOrder order =  
Qt::AscendingOrder)
```

```

        QTableWidgetItem *item)

void    setHorizontalHeaderLabels(const QStringList
        &labels)    往表格中添加项目

void    setItem(int row, int column, QTableWidgetItem *item)

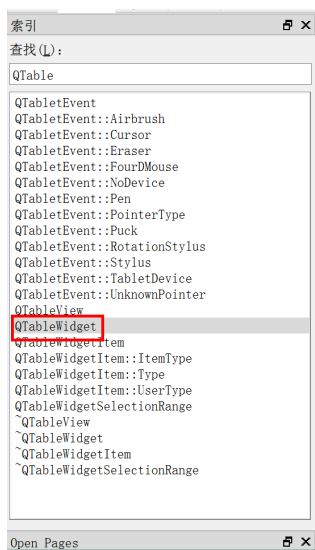
void    setItemPrototype(const QTableWidgetItem *item)

void    setRangeSelected(const QTableWidgetItemSelectionRange
        &range, bool select)

void    setRowCount(int rows)

void    setVerticalHeaderItem(int row, QTableWidgetItem
        *item)

```



```

        QTableWidgetItem *current, QTableWidgetItem *item)

void    itemActivated(QTableWidgetItem *item)

void    itemChanged(QTableWidgetItem *item)

void    itemClicked(QTableWidgetItem *item)

void    itemDoubleClicked(QTableWidgetItem *item)

void    itemEntered(QTableWidgetItem *item)

void    itemPressed(QTableWidgetItem *item)

void    itemSelectionChanged()

    • 7 signals inherited from QAbstractItemView
    • 3 signals inherited from QWidget
    • 2 signals inherited from QObject

```

知识点6 【下拉列表框】



```
QComboBox(QWidget *parent = Q_NULLPTR)
```

```
~QComboBox()
```

```
void addItem(const QString &text, const QVariant &userData =  
QVariant())
```

```
void addItem(const QIcon &icon, const QString &text, const QVariant  
&userData = QVariant())
```

```
void addItem(const QStringList &texts)
```

```
QCompleter * completer() const
```

```
int count() const
```

设置默认选项

Public Slots

```
void    clear()
void    clearEditText()
void    setCurrentIndex(int index)
void    setCurrentText(const QString &text)
void    setEditText(const QString &text)
```

- 19 public slots inherited from `QWidget`
- 1 public slot inherited from `QObject`

Signals

信号

Signals

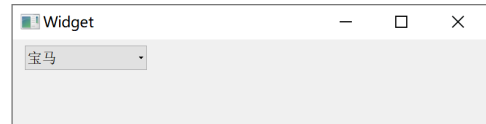
```
void    activated(int index)
void    activated(const QString &text)  信号被重载
void    currentIndexChanged(int index)  不能直接使用
void    currentIndexChanged(const QString &text)
void    currentTextChanged(const QString &text)
void    editTextChanged(const QString &text)
void    highlighted(int index)
void    highlighted(const QString &text)
```

- 3 signals inherited from `QWidget`
- 2 signals inherited from `QObject`

```

ui->setupUi(this);
//给下拉列表框 添加选项
QStringList list;
list<<"宝马"<<"奔驰"<<"奥迪"<<"五菱神车";
ui->comboBox->addItem(list);
ui->comboBox->resize(200,40);
//设置默认选项
ui->comboBox->setCurrentIndex(2);
//动起来 一旦信号发生重载 记得用函数指针匹配
void (QComboBox:: *p)(int) = &QComboBox::currentIndexChanged;
connect(ui->comboBox, p , [=](int index){
    qDebug()<<"index = "<<index<<endl;
    qDebug()<<ui->comboBox->currentText()<<endl;
} );

```



知识点7 【QLabel控件的使用】

1、QLabel设置文本

void setWordWrap(bool on)

QString text() const 得到文本

:TextFormat textFormat() const

Public Slots

void clear()

void setMovie(QMovie *movie) 设置动画

void setNum(int num)

void setNum(double num)

void setPicture(const QPicture &picture)

设置图片

void setPixmap(const QPixmap &)

void setText(const QString &) 设置文本

- 19 public slots inherited from QWidget
- 1 public slot inherited from QObject

2、设置图片

1、添加图片资源

2、QPixmap 对象加载图片

```

bool load(const QString &fileName, const char *format = Q_NULLPTR,
          Qt::ImageConversionFlags flags = Qt::AutoColor)

```

```

3 {
9     ui->setupUi(this);
10    //设置图片
11    QPixmap pix;
12    //给pix控件 加载一张图片
13    pix.load(":/image/sunny.png");
14    ui->label_2->setPixmap(pix);
15
16 }

```



3、设置动画

```
void setFormat(const QByteArray &format)
```

```
void setScaledSize(const QSize &size) 更改动画大小
```

```
int speed() const
```

```
int state() const
```

Public Slots

```
bool jumpToNextFrame()
```

```
void setPaused(bool paused)
```

```
void setSpeed(int percentSpeed) 播放动画
```

```
void start()
```

```
void stop() 结束动画
```

- 1 public slot inherited from QObject

```
ui->setupUi(this);  
//label设置一个动画  
QMovie *move = new QMovie(":/image/mario.gif");  
//设置动画的大小  
move->setScaledSize(QSize(400,300));  
ui->label->setMovie(move);  
  
//播放动画  
connect(ui->pushButton, &QPushButton::clicked, [=]() {  
    move->start();  
} );  
//结束动画  
connect(ui->pushButton_2, &QPushButton::clicked, [=]() {  
    move->stop();  
} );
```



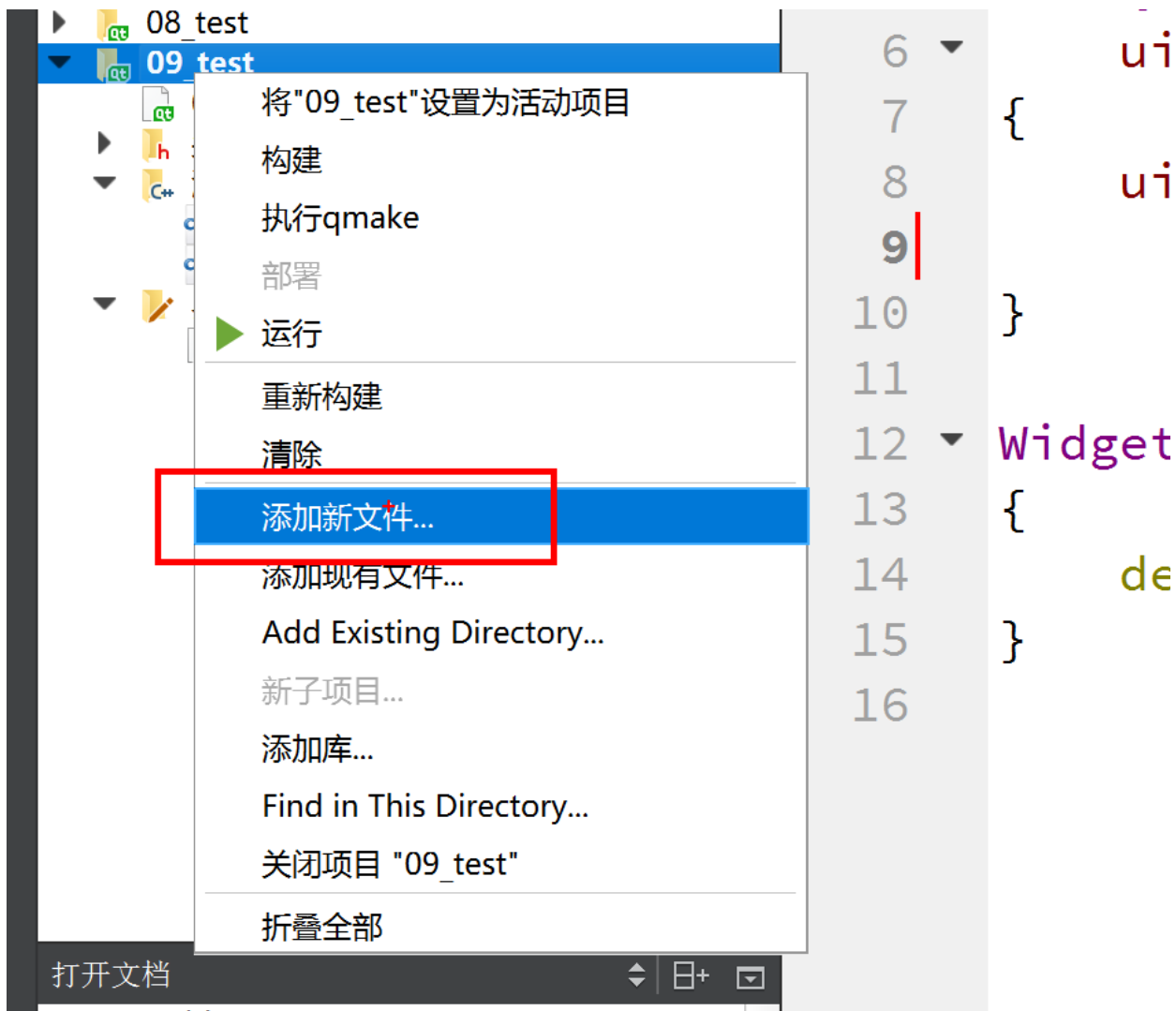
播放

暂停

知识点8 【自定义控件】

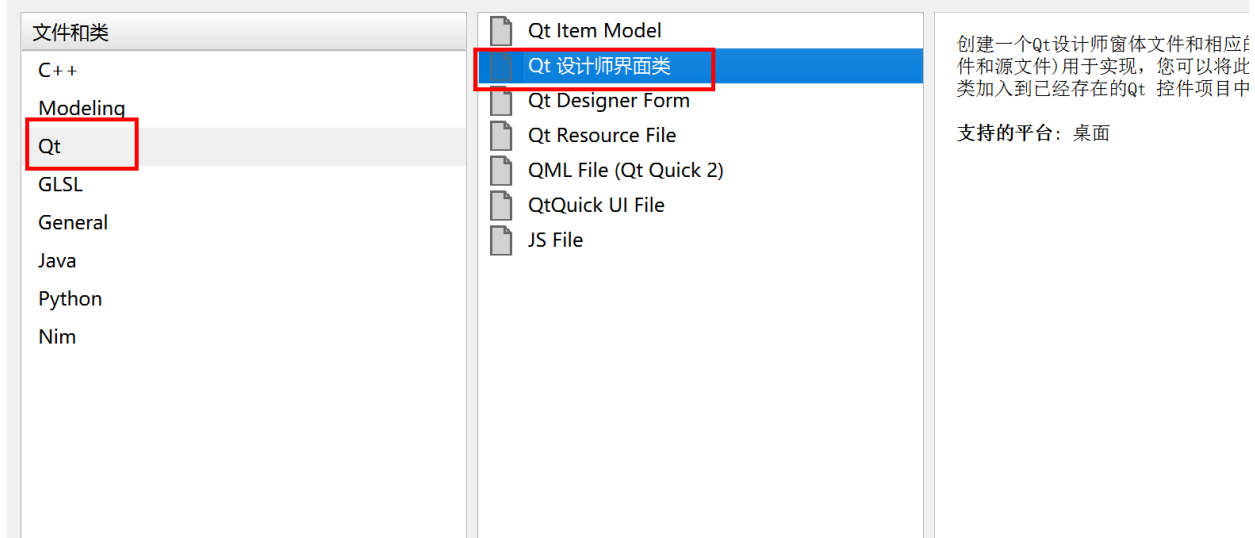
不是让你创建一个全新的控件 使用已有控件 拼出一个新的控件

1、定义一个自定义控件



新建文件

选择一个模板:



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
- Main Window
- Widget**
- 窗口部件

嵌入式设计

设备: 无

屏幕大小: 默认大小

下一步(N) 取消

选择类名

类

任意写

类名(C): MyWidget

头文件(H): mywidget.h

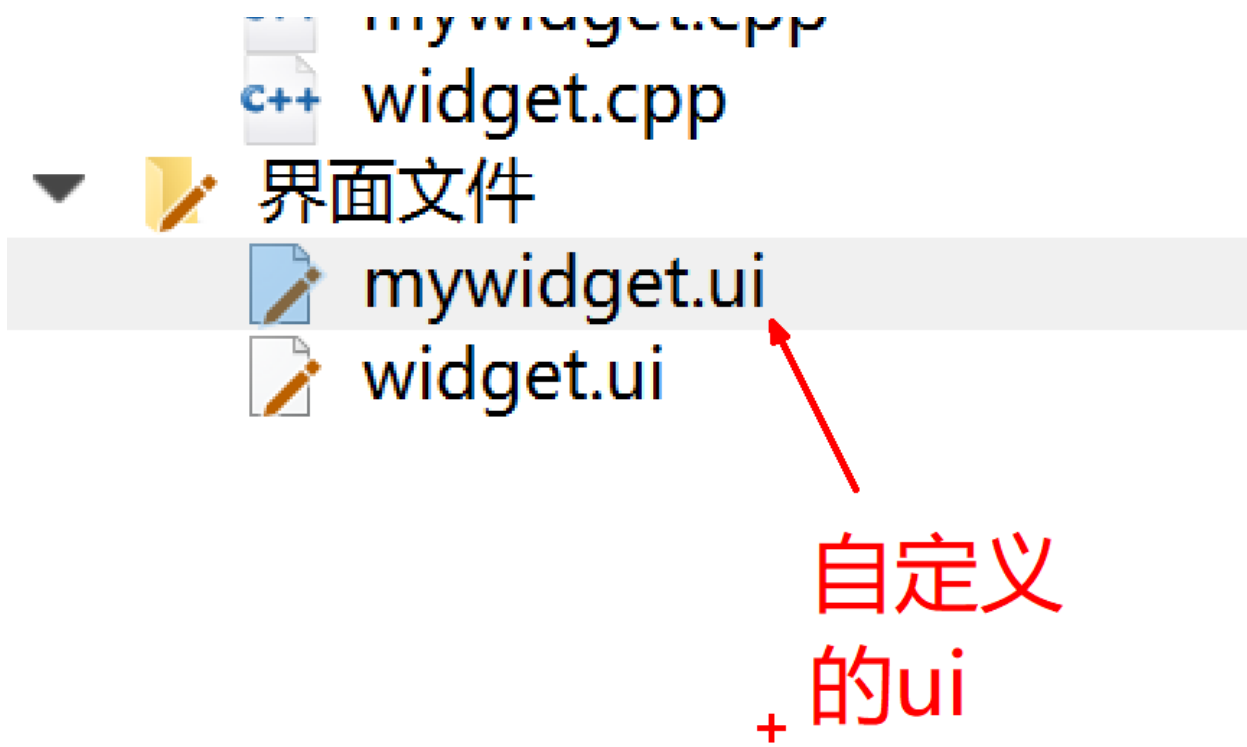
源文件(S): mywidget.cpp

界面文件(E): mywidget.ui

路径(P): C:\work\qt\day19\09_test 浏览...

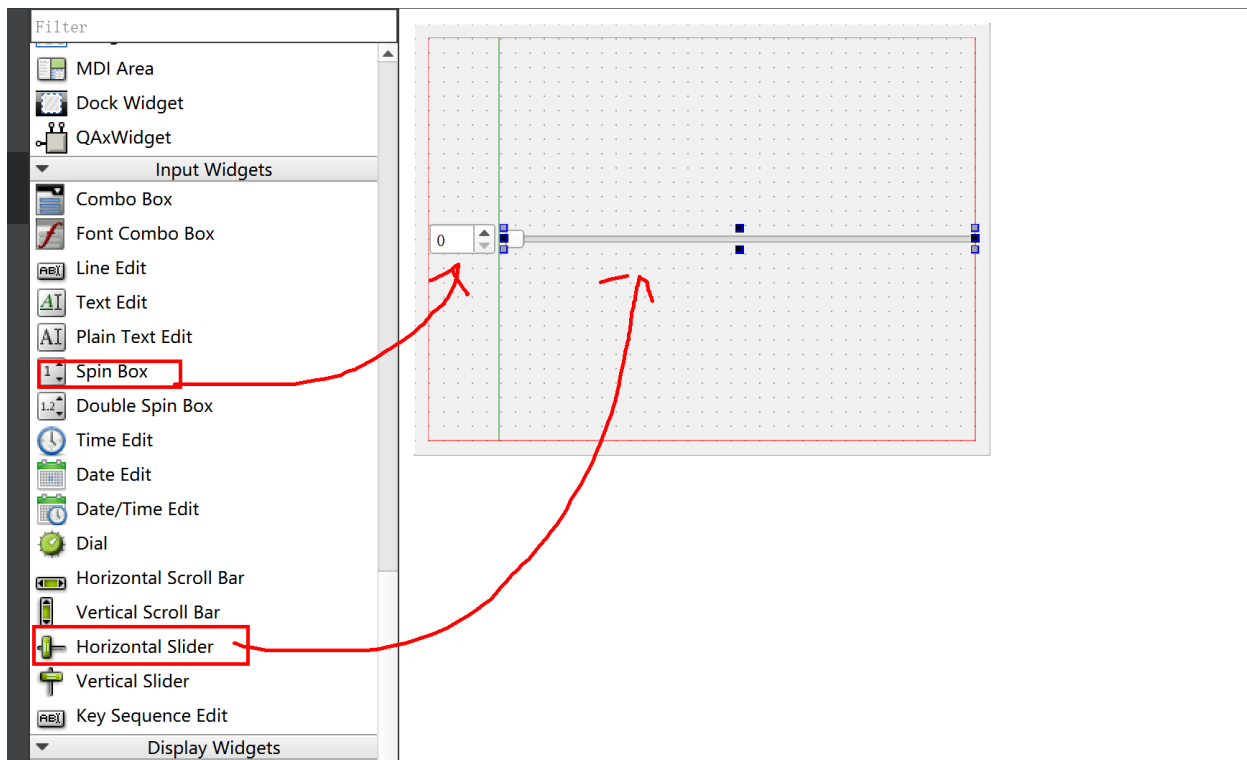
下一步(N)

取消

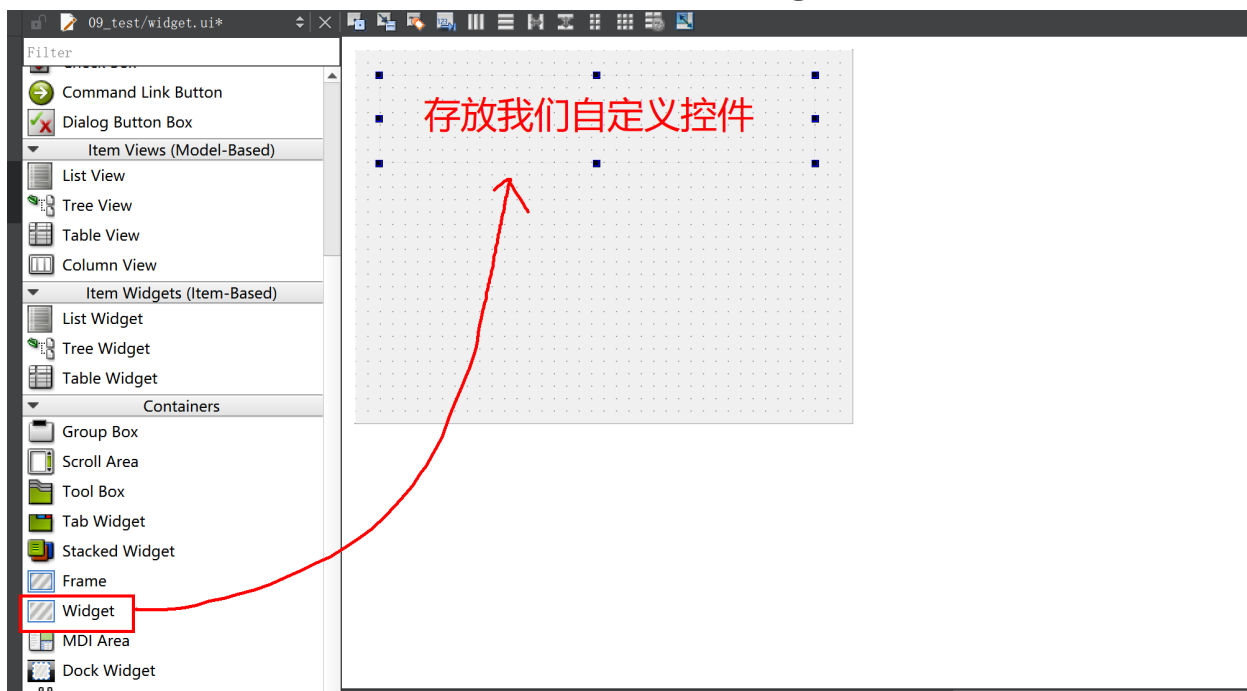


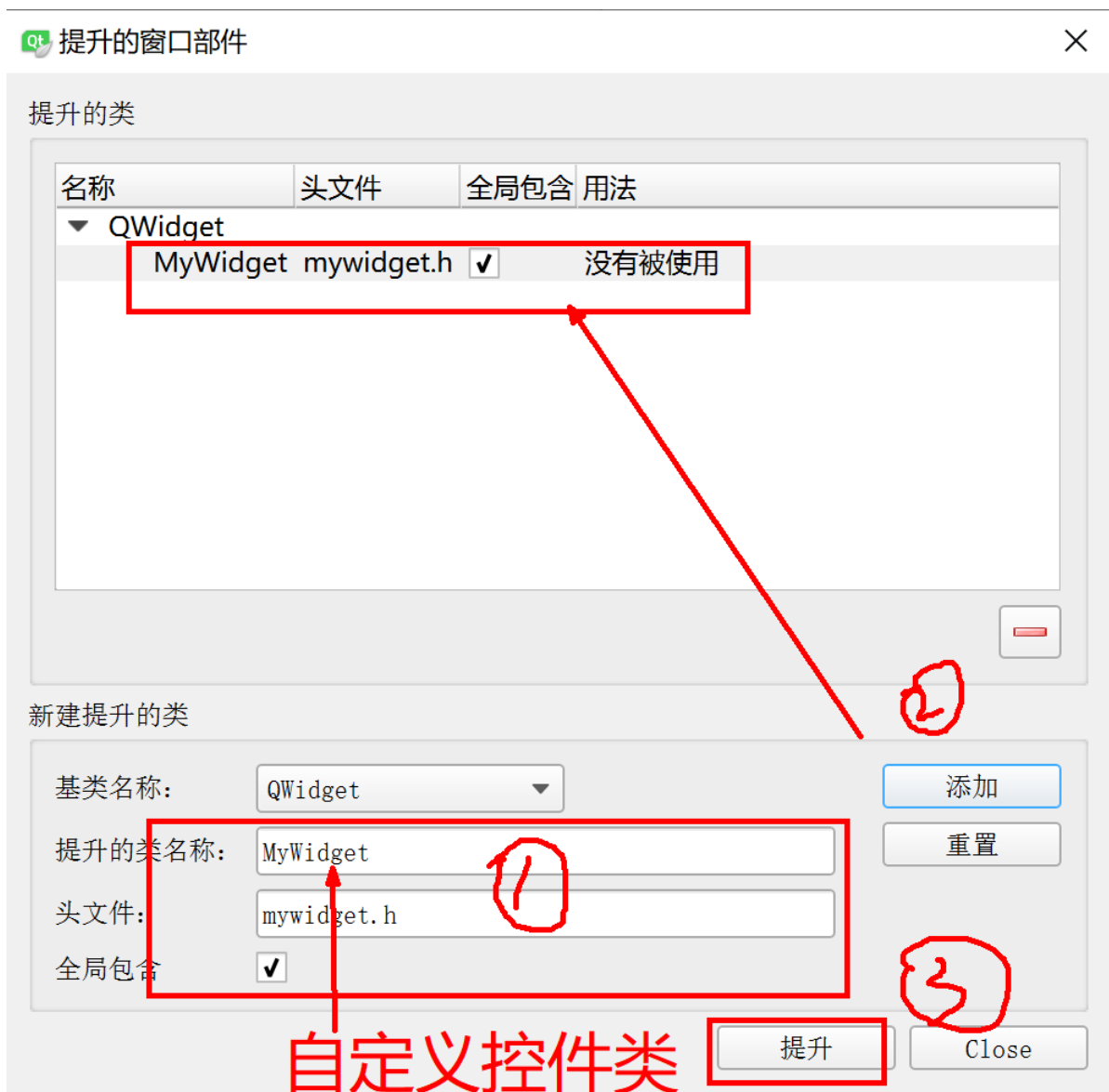
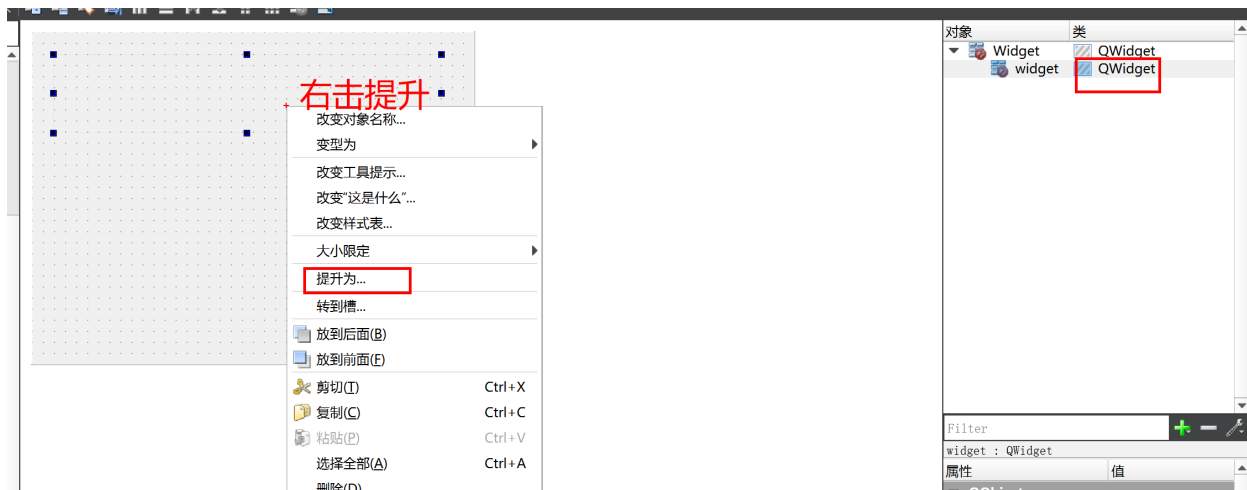
2、给自己的ui文件 添加常用控件

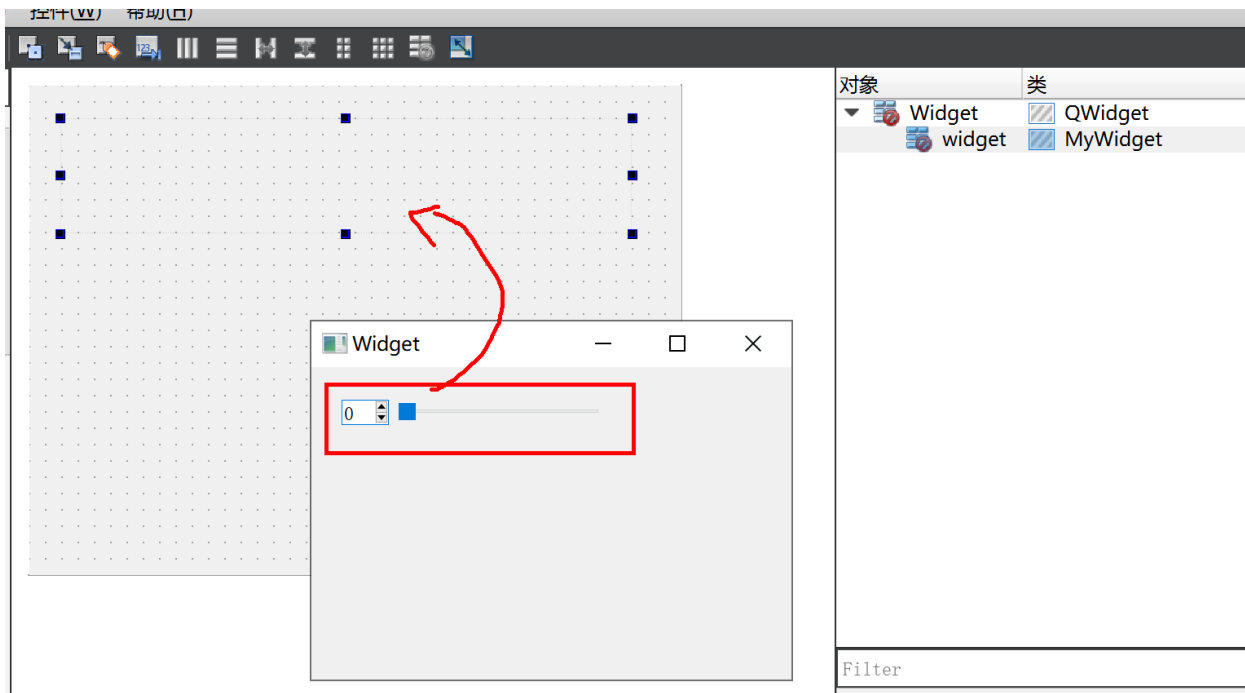
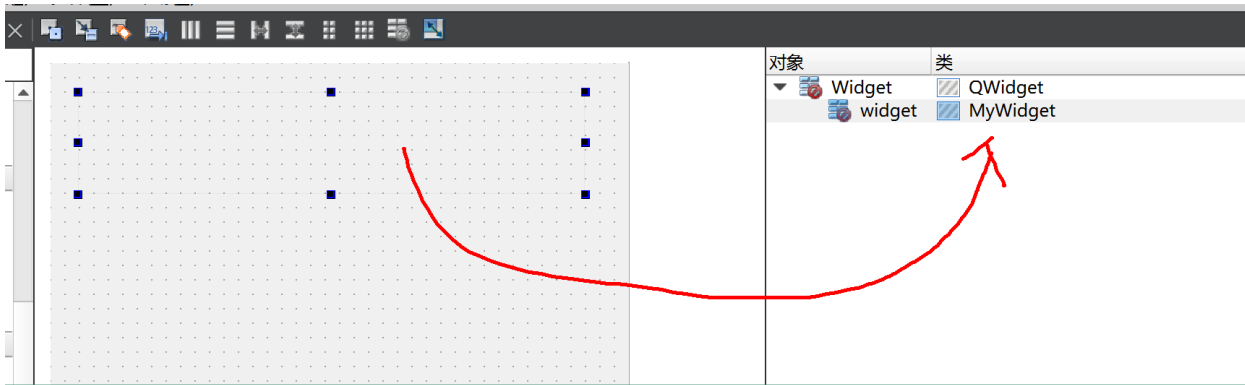




3、在其他ui文件中使用 自定义控件MyWidget







4、改变spinbox的值 进度条移动

spinbox的信号

Signals

void **valueChanged**(int *i*)

void **valueChanged**(const QString &*text*)

- 1 signal inherited from `QAbstractSpinBox`
- 3 signals inherited from `QWidget`
- 2 signals inherited from `QObject`

slider的槽函数

Public Slots

```
void    setOrientation(Qt::Orientation)
```

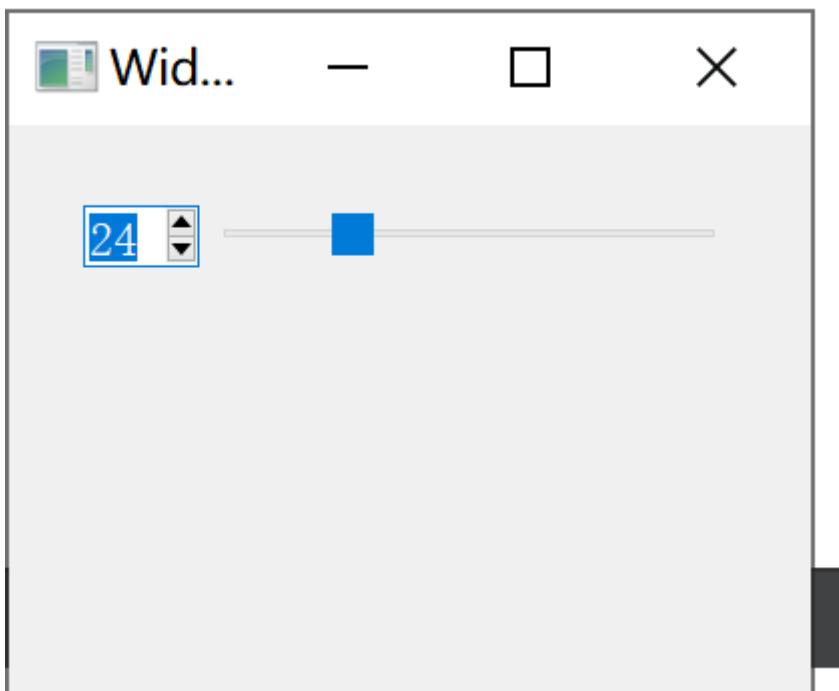
```
void    setRange(int min, int max)
```

```
void    setValue(int)
```

- 19 public slots inherited from `QWidget`
- 1 public slot inherited from `QObject`

在mywidget.cpp的构造函数

```
1 //spinbox改变 导致 slider滑动
2 void (QSpinBox:: *p)(int) = &QSpinBox::valueChanged;
3 connect(ui->spinBox, p,\
4 ui->horizontalSlider, &QSlider::setValue );
```



5、//拖动 slider 更改spinbox

slider的信号:

Signals

```
void    actionTriggered(int action)
void    rangeChanged(int min, int max)
void    sliderMoved(int value)
void    sliderPressed()
void    sliderReleased()
void    valueChanged(int value)
```

- 3 signals inherited from `QWidget`
- 2 signals inherited from `QObject`

Protected Types

spinbox槽函数

Public Slots

```
void    setValue(int val)
```

- 4 public slots inherited from `QAbstractSpinBox`
- 19 public slots inherited from `QWidget`
- 1 public slot inherited from `QObject`

Signals

在mywidget.cpp的函数中

```
//拖动 slider 更改spinbox
connect(ui->horizontalSlider, &QSlider::valueChanged, \
        ui->spinBox, &QSpinBox::setValue );
}

MyWidget::~MyWidget()
{
```

