



北京大学
PEKING UNIVERSITY

Qt 编程

课前多吼歪

- 期中考试

- 4月27日(周三) 上午10-12点
- 考试地点：计算中心456机房, 具体考位当天机房外看
- 考试题型：选择题 & 程序题 (注意POJ上两场比赛, 不要漏答)
- 考试范围：C++部分 (含C++11, 不含高级)
- 考场要求：不允许带任何电子设备, 会发草稿纸, 带学生证
- 考试纪律
- 考试技巧与问题

课前多吼歪

- 本周日上机 继续模考 & 助教答疑, 有考勤
- 本周日教学组答疑
 - 17:30 理科1号楼1223
- 五一都不上机, 但要记得写魔兽和研究Qt
- 程序设计竞赛

示例4-6 信号和槽

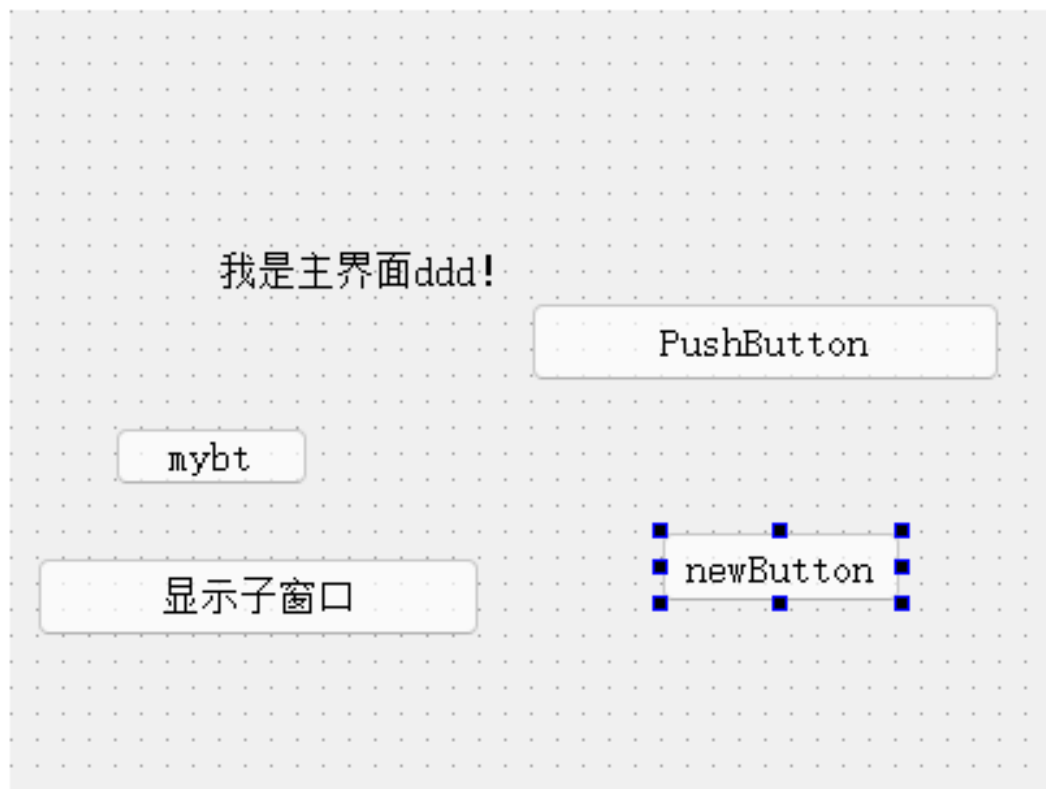
- Qt中的信号槽机制

在信号槽机制中, 某个事件发生会广播一个**信号**,

如果一个对象对这个信号有兴趣就会使用连接 (connect) 函数,

→ 用自己的一个事件响应函数 (称为slot, 也就是**槽**) 来处理这个信号

示例4-6 信号和槽



以按钮组件为例,
在添加一个新的“newButton”
(名为btNewy) 的按钮组件后,
通过**信号槽机制**,可以实现**点击**
这个按钮执行特定的函数的功能

示例4 信号和槽的自动关联

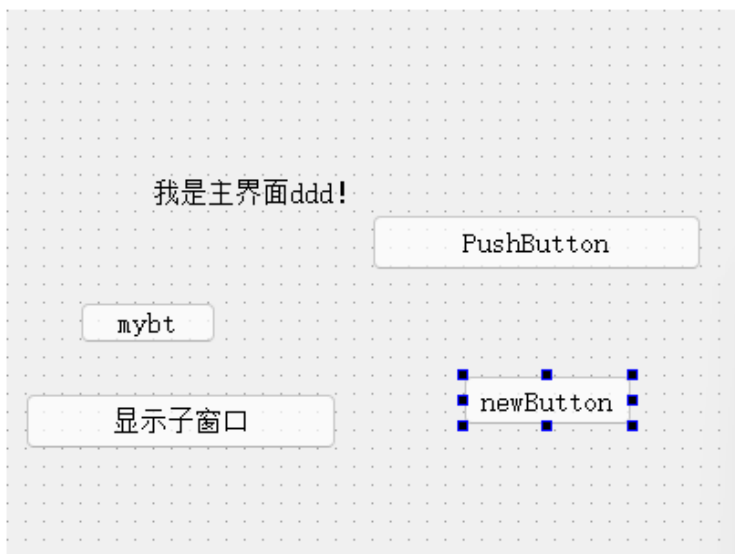
- 信号槽机制的关键，在于如何让程序知道对于某个信号，应该调用什么槽函数进行处理，即信号和槽的**关联**
- 接下来的示例将展示如何通过设计界面，**自动**实现信号和槽的关联

示例4 信号和槽的自动关联

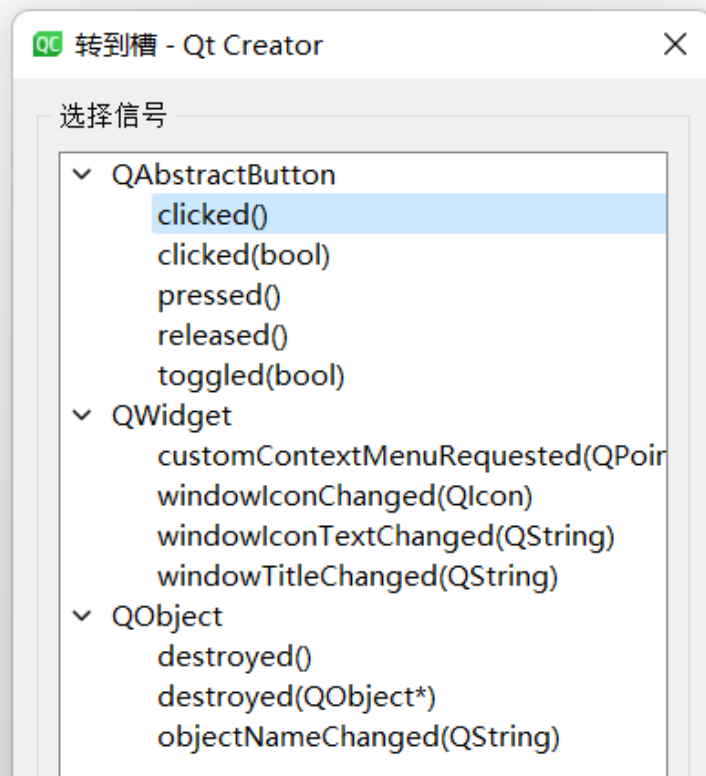


右键打开按钮组件的
菜单栏，点击转到槽

示例4 信号和槽的自动关联



这里选择了 **clicked()** 信号, 表示按钮在被点击时就会向其他组件发送这个信号



- 会出现一个选择**信号**的界面, 用来选择按钮在什么状态下会执行特定函数
- 在达到这种状态时, 按钮组件就会在全局发送一个其他所有组件都能捕获的**标识**, 这就是信号槽机制中的**信号**

示例4 信号和槽的自动关联

- 在mywidget.h中

```
private slots:  
    void on_mybutton_clicked();  
    void on_btNewy_clicked();
```

- 在mywidget.cpp中

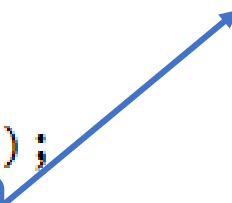
```
void MyWidget::on_btNewy_clicked()  
{  
    ui->label->setText("btnew clicked");  
}
```

在选择好信号后, QtCreator会自动在MyWidget类中生成一个私有的响应函数on_btNewy_clicked(), 捕获到信号后这个函数就会被执行, 这个响应函数就叫槽函数

示例4 信号和槽的自动关联

- 在mywidget.h中

```
private slots:  
    void on_mybutton_clicked();  
    void on_btNewy_clicked();
```



- 在mywidget.cpp中

```
void MyWidget::on_btNewy_clicked()  
{  
    ui->label->setText("btnew clicked");  
}
```

- QtCreator自动生成的槽函数的名字格式是on_xxx_yyy(), 其中xxx为发出信号的**组件名**, yyy为**信号名**, 这里程序判断信号和槽的是否关联的依据也是**槽函数的名字**
- 注意自动生成的槽函数是私有成员函数, 只能和类的成员组件发出的信号关联, 所以不会出现**命名冲突**的问题

示例4 信号和槽的自动关联

- 在mywidget.h中

```
private slots:  
    void on_mybutton_clicked();  
    void on_btNewy_clicked();
```

- 在mywidget.cpp中

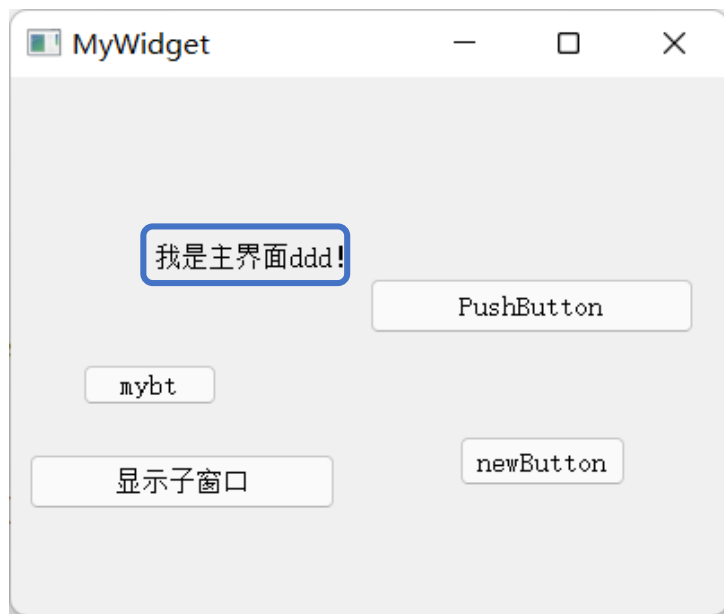
```
void MyWidget::on_btNewy_clicked()  
{  
    ui->label->setText("btnew clicked");  
}
```

- 自动生成的槽函数是没有内容的，可以给这个函数添加一个改变对话框的文本的行为来进行测试

示例4 信号和槽的自动关联

- 可以看到点击newButton后
MyWidget显示的文本按照
槽函数的设置进行了改变

```
void MyWidget::on_btNewy_clicked()  
{  
    ui->label->setText("btnew clicked");  
}
```



运行效果

示例5 信号和槽的手动关联

- 上一个示例通过设计界面的“转到槽”功能，自动进行了信号和槽函数的关联
- 接下来的示例将展示如何**手动**进行信号和槽之间的关联

示例5 信号和槽的手动关联

- 在mywidget.cpp中

```
connect(ui->showChildButton, &QPushButton::clicked,  
        this, &MyWidget::showChildDialog);
```

```
//connect是QObject成员函数 各种类都继承 QObject  
//手动关联信号和槽  
//指定 ui->showChildButton按钮上面的 clicked事件,  
//将由本对象的 showChildDialog响应
```

除了利用 QtCreator 自动生成槽函数外，也可以在MyWidget类内部初始化时利用**connect**函数**手动**将某个组件的信号与自身成员函数或者其他类的公有成员函数进行关联

示例5 信号和槽的手动关联

- **connect函数**

第一个参数：发出信号的组件

第二个参数：发出的信号，这里设置为

点击“显示子窗口”按钮时发出的信号

```
connect(ui->showChildButton, &QPushButton::clicked,  
        this, &MyWidget::showChildDialog);  
//connect是QObject成员函数 各种类都继承 QObject  
//手动关联信号和槽  
//指定 ui->showChildButton按钮上面的 clicked事件,  
//将由本对象的 showChildDialog响应
```



示例5 信号和槽的手动关联

第三个参数：捕获信号的对象，也就是执行槽函数的对象，
`this`指代的MyWidget类

```
connect(ui->showChildButton, &QPushButton::clicked,  
        this, &MyWidget::showChildDialog);  
//connect是QObject成员函数 各种类都继承 QObject  
//手动关联信号和槽  
//指定 ui->showChildButton按钮上面的 clicked事件，  
//将由本对象的 showChildDialog响应
```

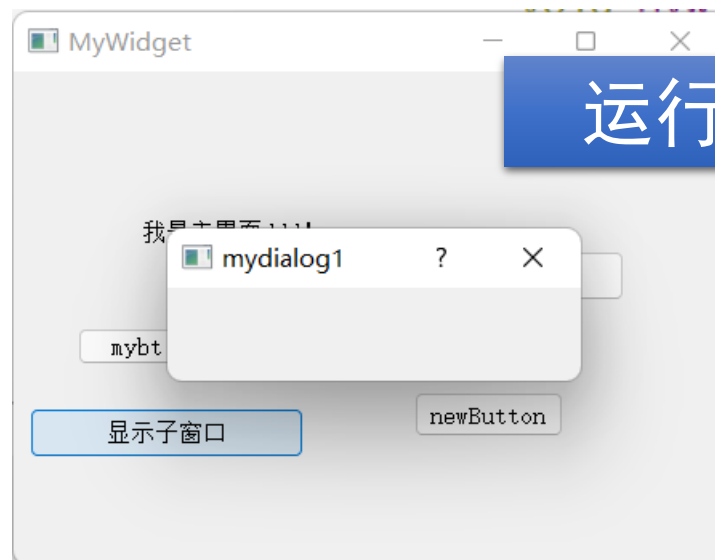
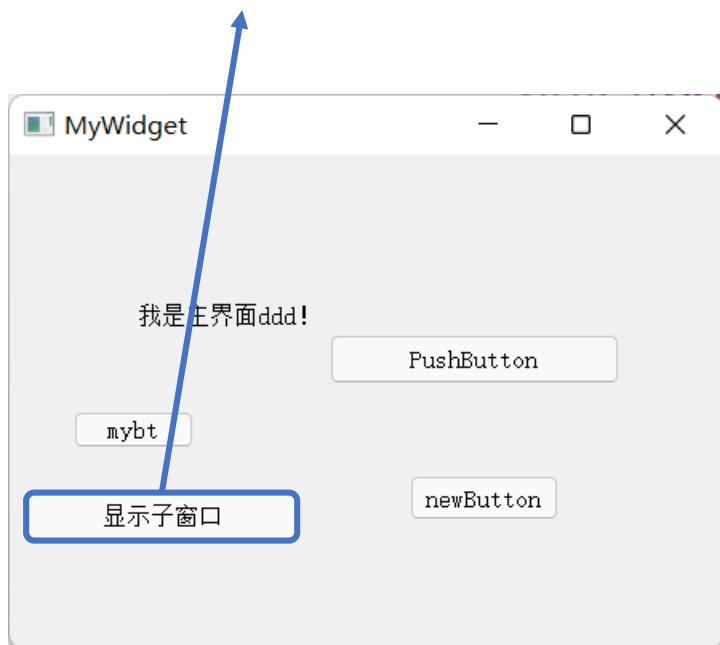
```
void MyWidget::showChildDialog()  
{  
    QDialog *dialog = new QDialog(this);  
    dialog->show(); //弹出非模态对话框  
}
```

第四个参数：槽函数，这里的功能就是 new 一个新的Qdialog对话框并且显示出来

示例5 信号和槽的手动关联

点击按钮后执行槽函数弹出
一个新的对话框

```
void MyWidget::showChildDialog()  
{  
    QDialog *dialog = new QDialog(this);  
    dialog->show(); //弹出非模态对话框  
}
```



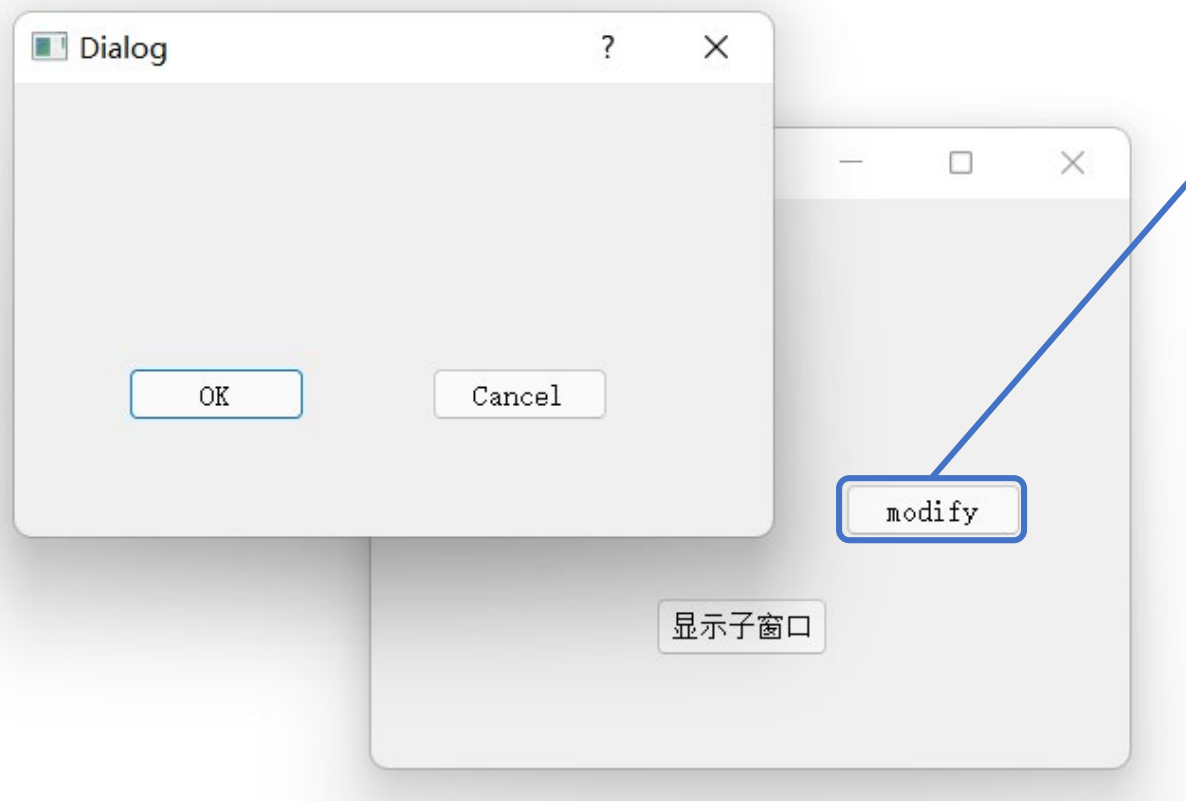
运行效果

示例6 编辑槽映射

- 信号槽**自动**关联中，QtCreator会自动生成槽函数，但也需要自己写槽函数内容
- 信号槽**手动**关联中，利用connect函数允许随意设置信号和槽的关联，但需要在类的初始化阶段写相应代码，比较麻烦

接下来的示例将展示如何通过UI设计界面编辑**槽映射**，实现一个组件的信号和另外组件的槽函数的关联

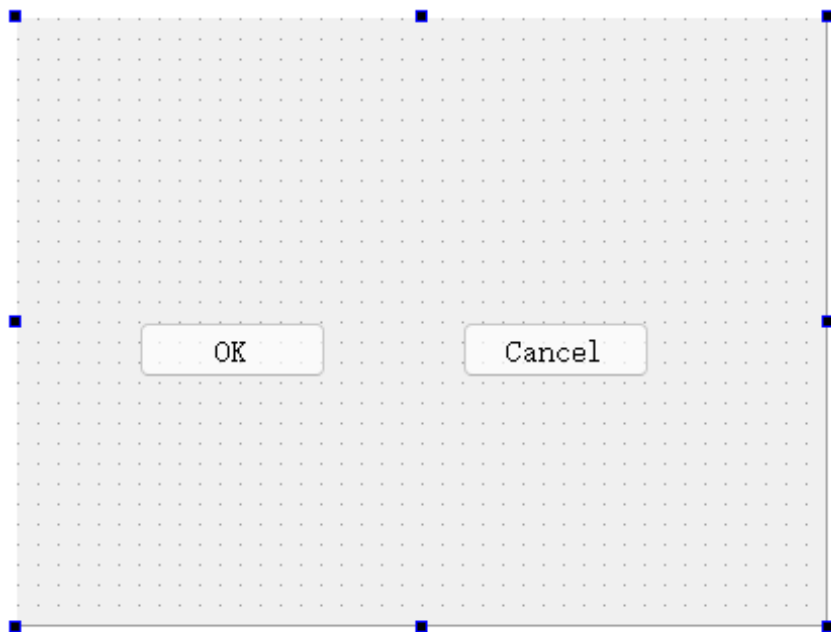
示例6 编辑槽映射



- 本示例的程序由下方的父窗口和上方的对话框组成
- 点击父窗口中的modify按钮将弹出对话框，接着点击对话框的OK按钮后，对话框与父窗口均会被关闭

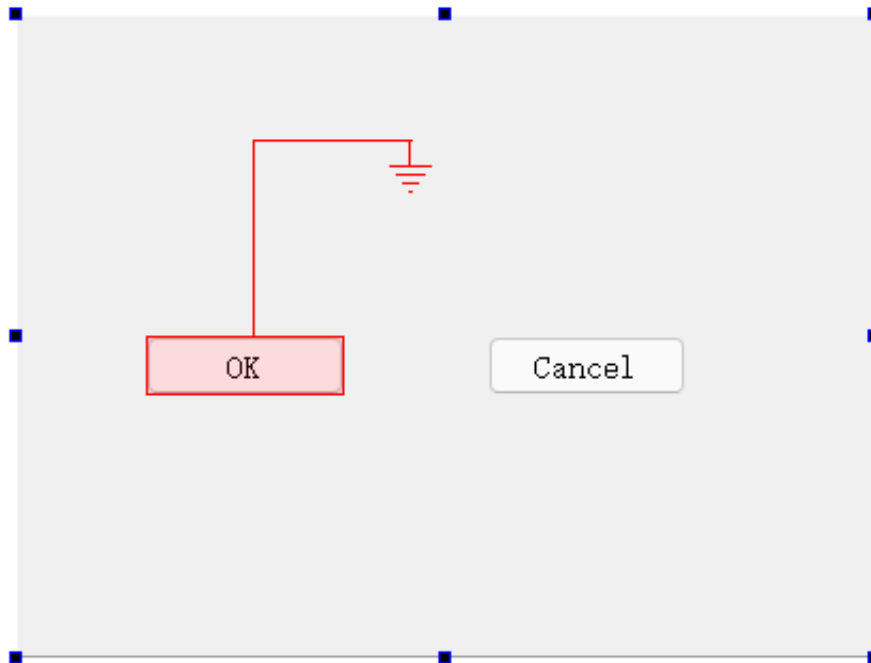
本示例将借助**编辑槽映射**的方式来实现上述效果

示例6 编辑槽映射



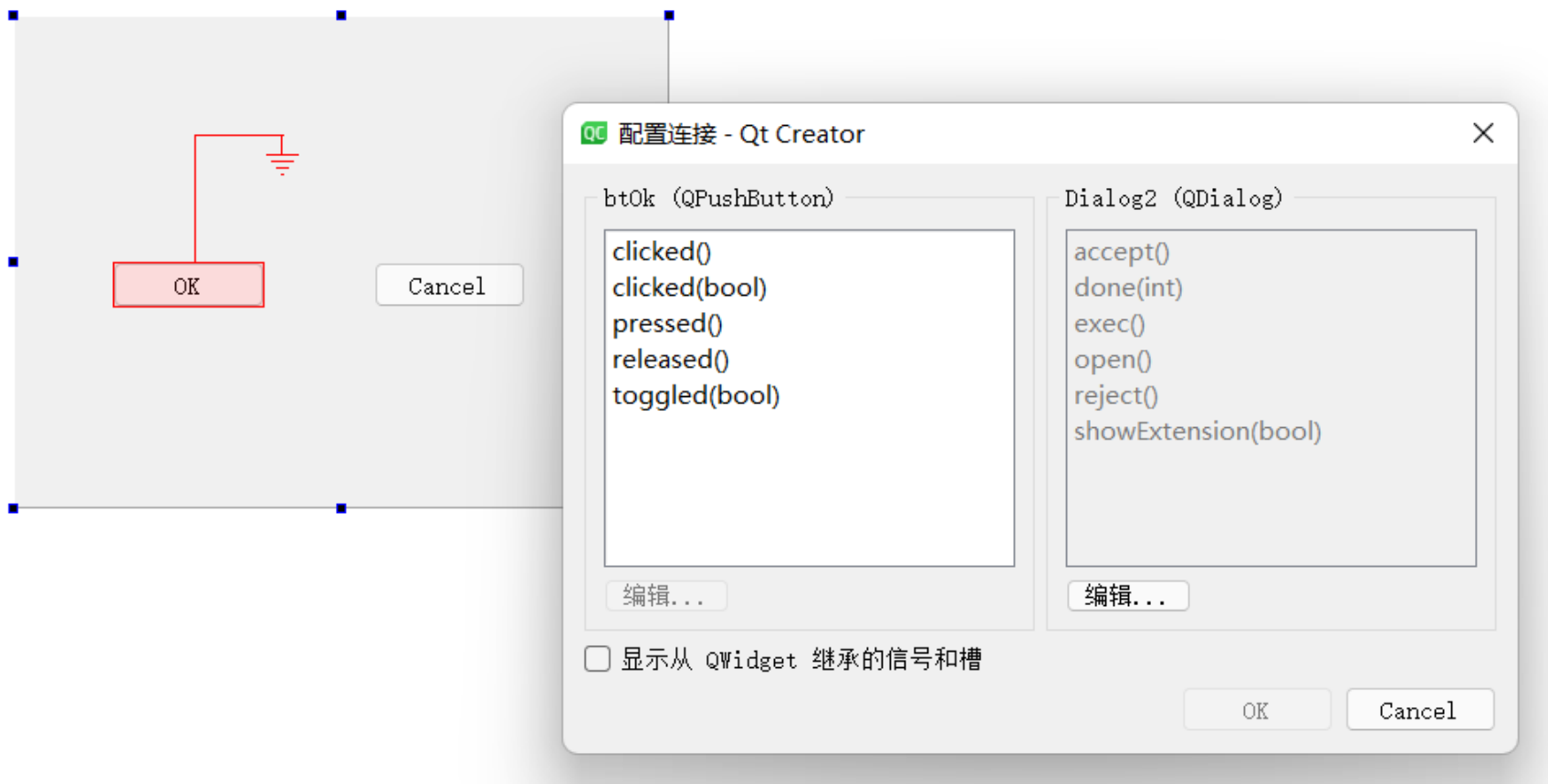
- 这个是MyWidget主界面弹出的QDialog类型的对话框, 在对话框的添加两个原始的QPushButton类的按钮组件

示例6 编辑槽映射



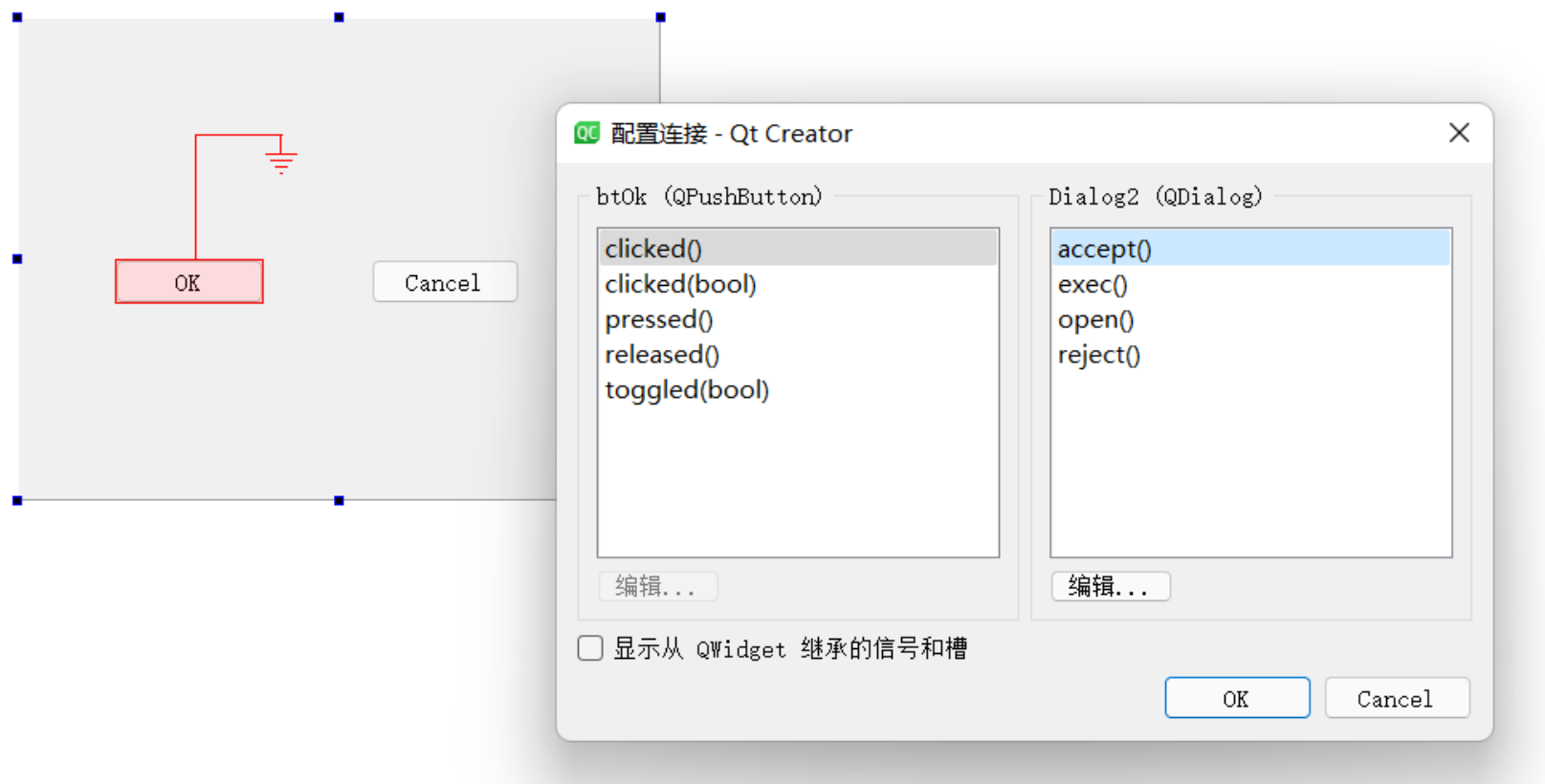
点击F4，选中发送信号的组件，拖动连线停在接收信号的组件，这里发送信号的按钮被命名为**btOK**，接收信号的对话框被命名为**Dialog2**

示例6 编辑槽映射



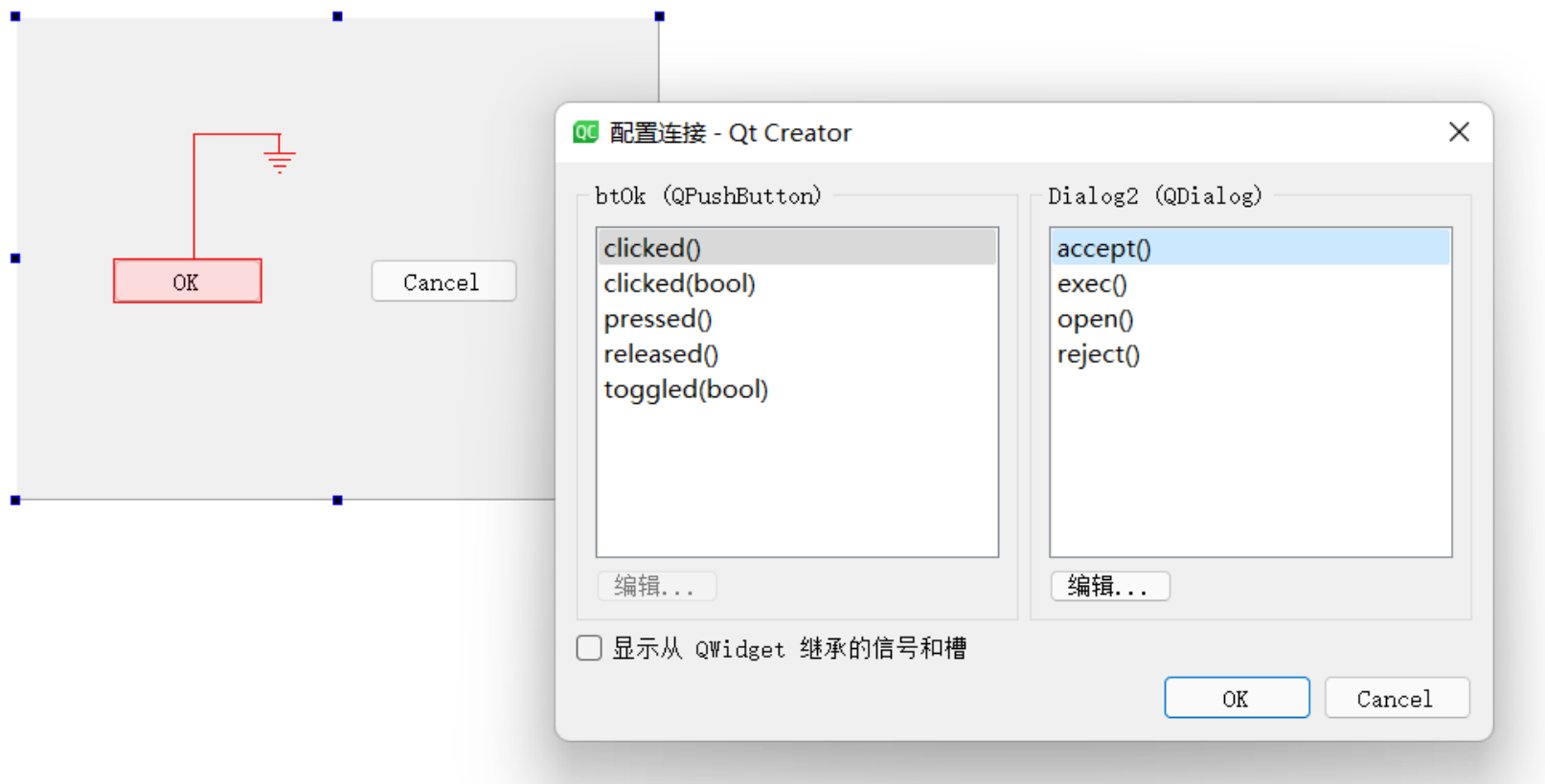
- 松开鼠标后会出现配置连接的选项
- 可以在这里编辑槽映射

示例6 编辑槽映射



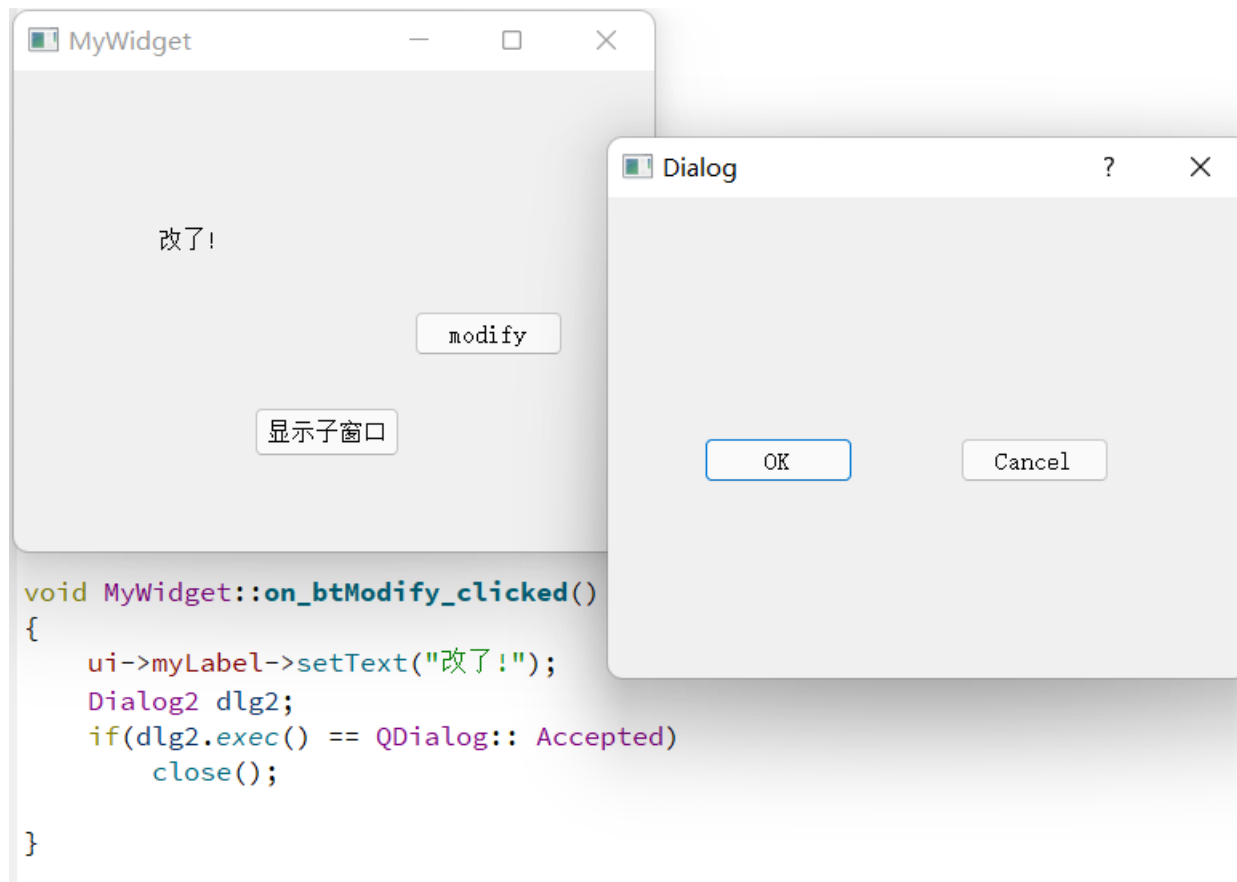
这里选择发送方**btOK**的信号为**clicked()**, 接收方**Dialog2**接收到信号后需要执行的槽函数为**accept()**

示例6 编辑槽映射



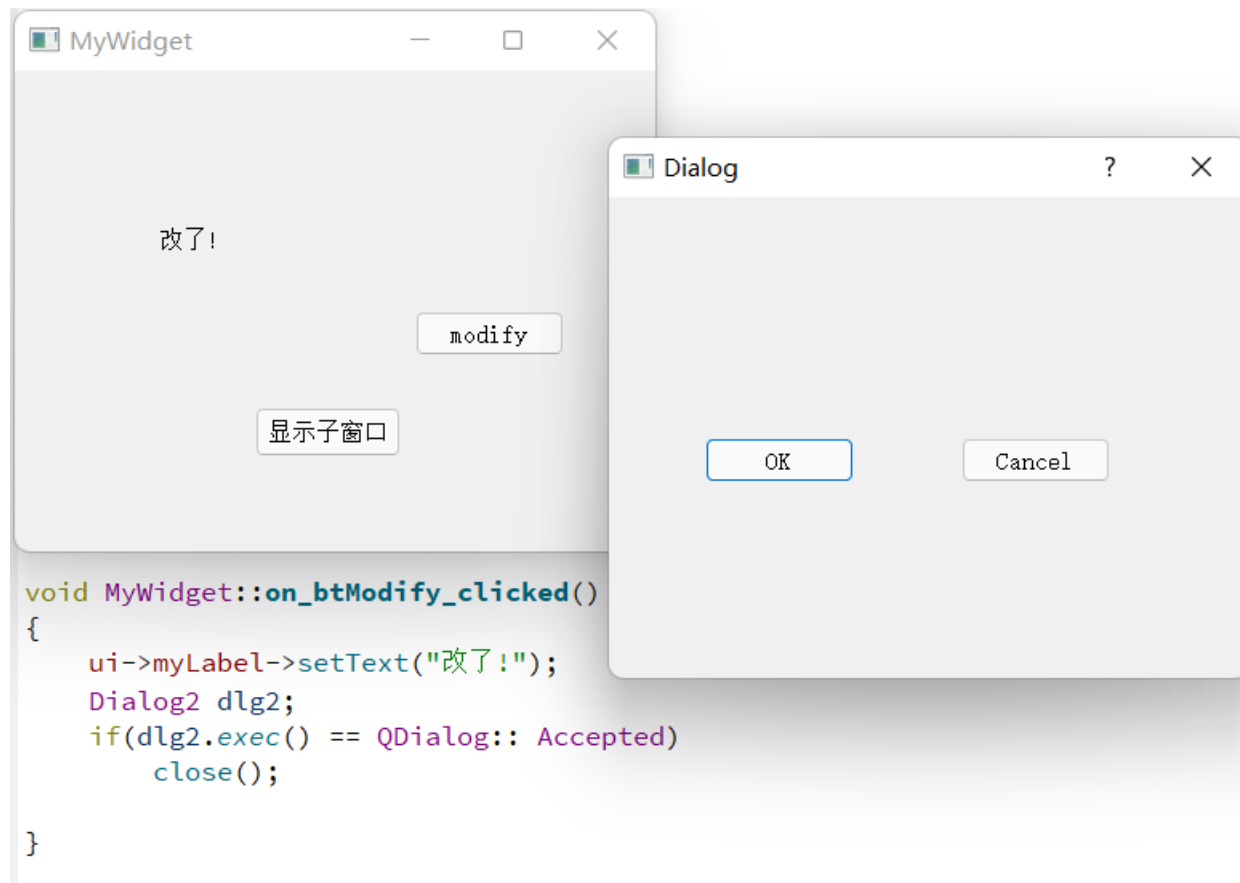
- 行为逻辑变成了点击按钮,按钮会发出信号使对话框执行accept()函数
- accept() 函数会把对话框Dialog2的返回值设置为Accepted成员变量

示例6 编辑槽映射



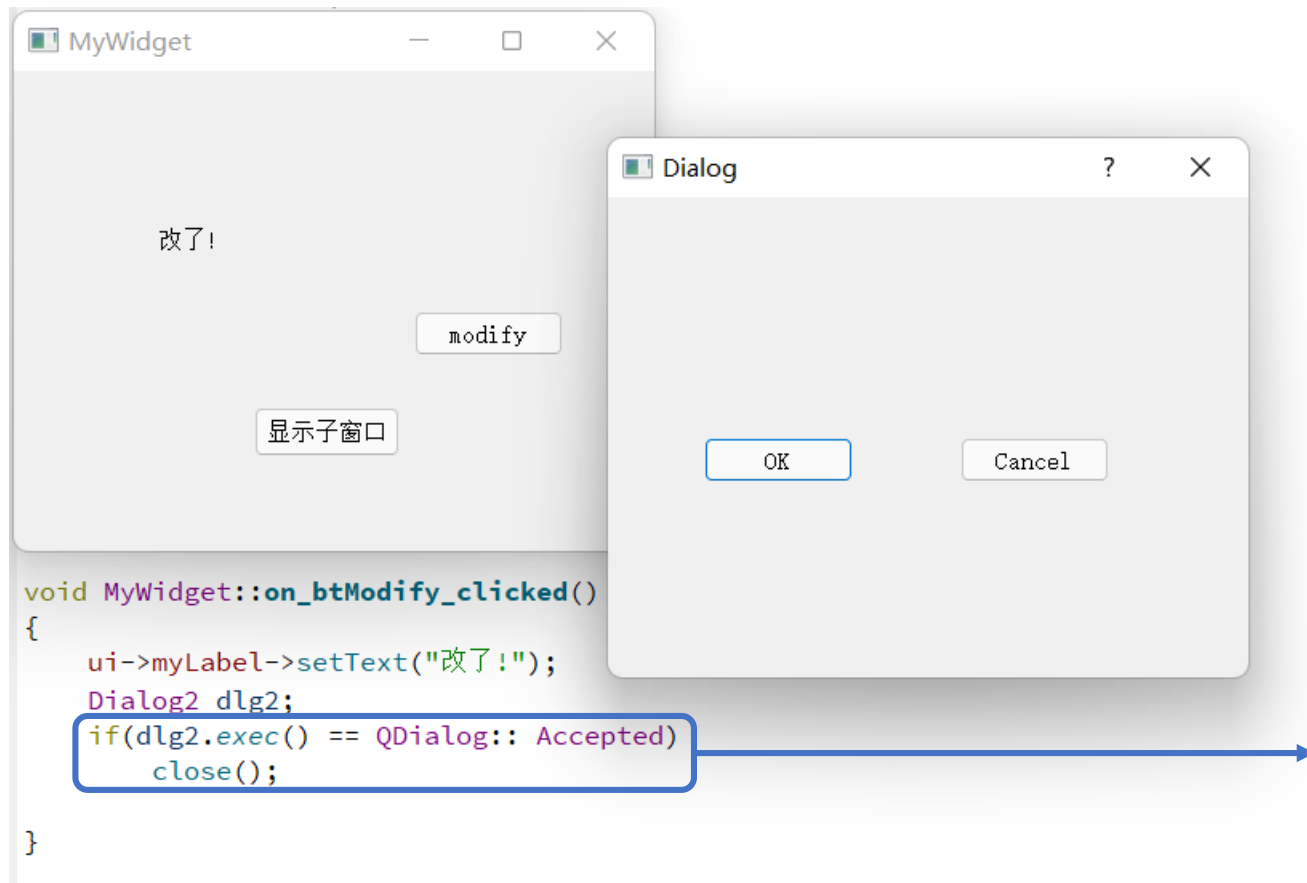
点击btModify按钮创建并
弹出对话框Dialog2

示例6 编辑槽映射



点击Dialog2对话框的btOK按钮，根据刚才编辑槽映射的结果，Dialog2会执行完毕并返回Accepted变量

示例6 编辑槽映射



过程都发生在btModify的槽函数中，这是MyWidget类的成员函数，所以当Dialog2对话框返回Accepted时，槽函数执行close()，主页面MyWidget也随之关闭

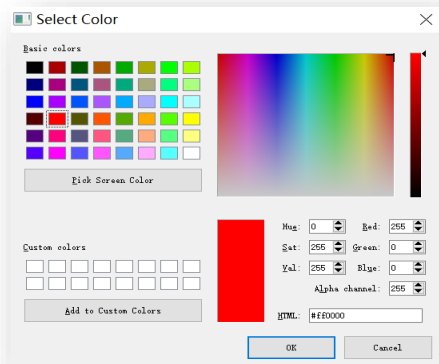
示例7 常用对话框介绍

本示例介绍以下八个常用的对话框

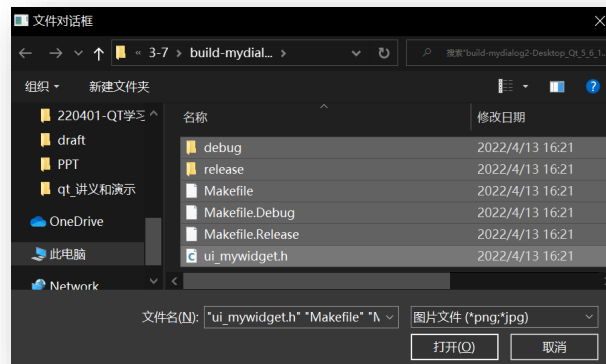
- 颜色对话框
- 文件对话框
- 字体对话框
- 输入对话框
- 消息对话框
- 进度对话框
- 错误信息对话框
- 向导对话框

示例7 常用对话框介绍

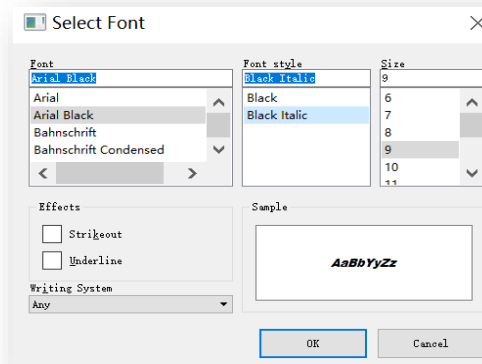
颜色对话框



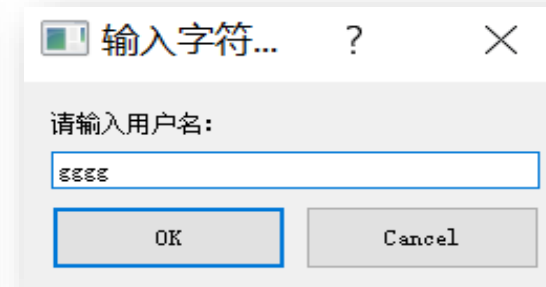
文件选择对话框



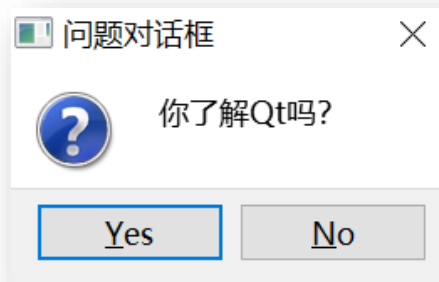
字体对话框



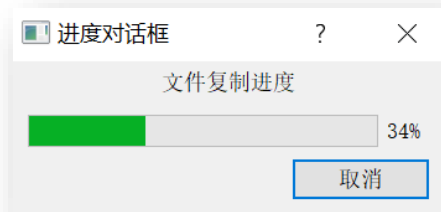
输入对话框



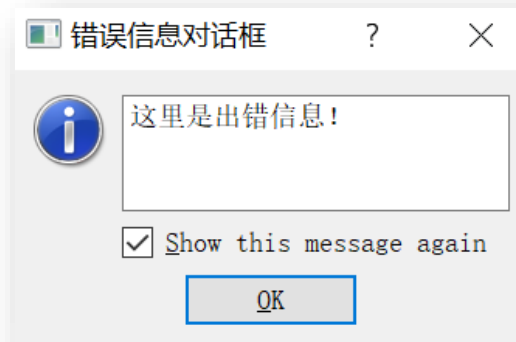
消息对话框



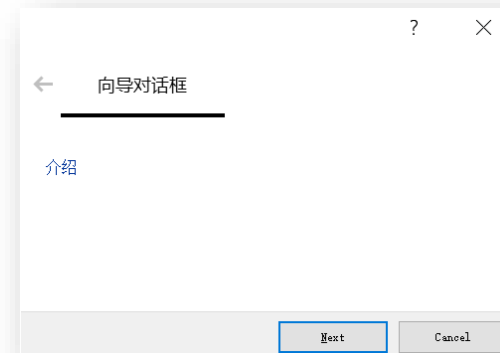
进度对话框



错误信息对话框



向导对话框

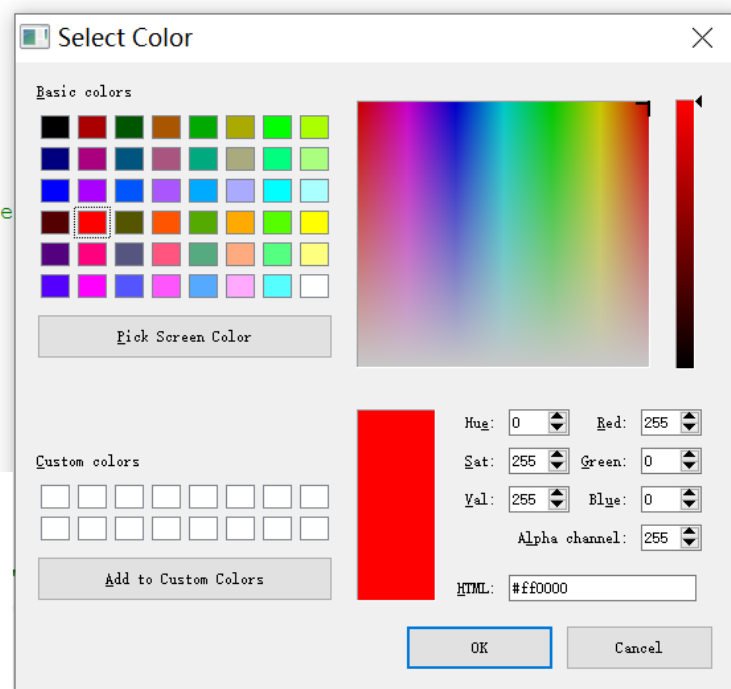


示例7 常用对话框介绍

- 颜色对话框

QColorDialog为Qt中的颜色对话框类
设置Qt::red为默认选择的颜色

```
26 // 颜色对话框
27 void MyWidget::on_pushButton_clicked()
28 {
29     // QColor color = QColorDialog::getColor(Qt::red, this, tr("颜色对话框"),
30     //                                     QColorDialog::ShowAlphaChannel);
31
32     QColorDialog dialog(Qt::red, this); // 创建对象
33     dialog.setOption(QColorDialog::ShowAlphaChannel); // 显示alpha选项
34     dialog.exec(); // 以模态方式运行对话框
35     QColor color = dialog.currentColor(); // 获取当前颜色
36
37     qDebug() << "color: " << color;
38 }
39
```



应用程序输出
mydialog2 X

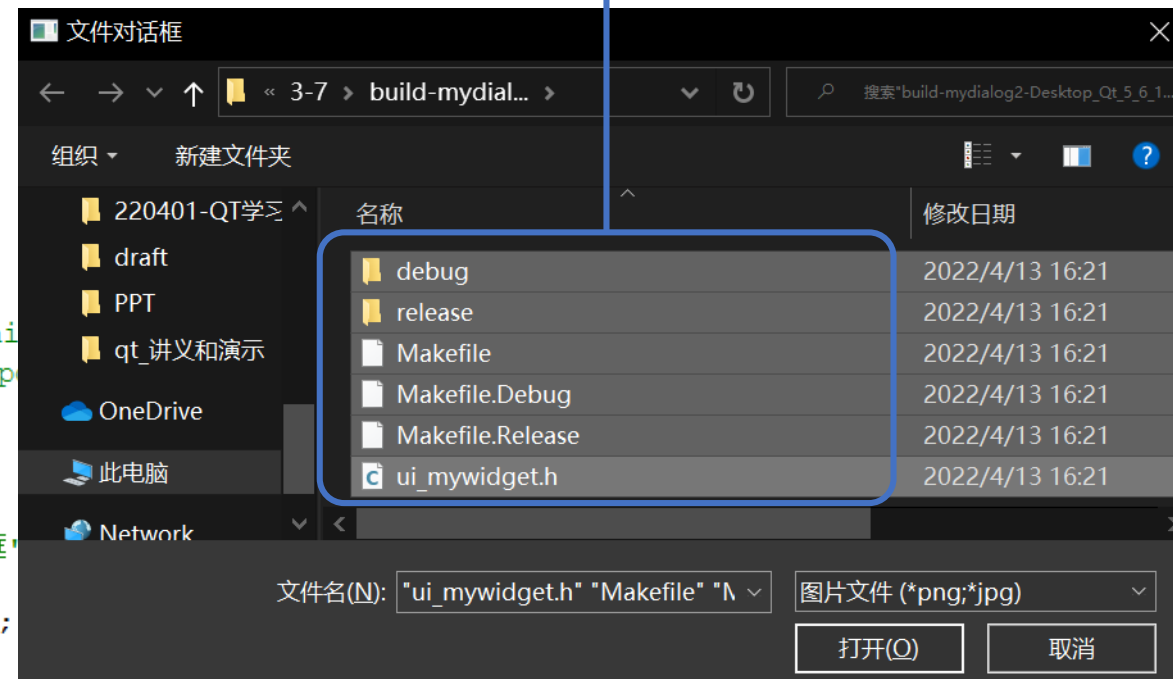
Starting C:\DISKD\qtthing\booksample\03\3-7\build-mydialog2-Desktop_Qt_5_6_1_MinGW_32bit-Debug\debug\mydialog2.exe...
color: QColor(ARGB 1, 1, 0, 0)

示例7 常用对话框介绍

- 文件对话框

可以同时选择多个文件

```
39
40 // 文件对话框
41 void MyWidget::on_pushButton_2_clicked()
42 {
43     // QString fileName = QFileDialog::getOpenFileName(this,
44     //                                     "D:", tr("图片文件(*png *jpg)"));
45
46     // qDebug() << "fileName:" << fileName;
47
48     QStringList fileNames =
49         QFileDialog::getOpenFileNames(this, tr("文件对话框"),
50                                     "D:", tr("图片文件(*png *jpg);"));
51
52     qDebug() << "fileNames:" << fileNames;
53 }
54
```



```
Starting C:\DISKD\qtthing\booksample\03\3-7\build-mydialog2-Desktop_Qt_5_6_1_MinGW_32bit-Debug\debug\mydialog2.exe...
fileNames: ("C:/diskd/qtthing/booksample/03/3-7/build-mydialog2-Desktop_Qt_5_6_1_MinGW_32bit-Debug/Makefile", "C:/diskd/qtthing/booksample/03/3-7/build-mydialog2-Desktop_Qt_5_6_1_MinGW_32bit-Debug/Makefile.Debug", "C:/diskd/qtthing/booksample/03/3-7/build-mydialog2-Desktop_Qt_5_6_1_MinGW_32bit-Debug/Makefile.Release", "C:/diskd/qtthing/booksample/03/3-7/build-mydialog2-Desktop_Qt_5_6_1_MinGW_32bit-Debug/ui_mywidget.h")
```

输出所有文件名

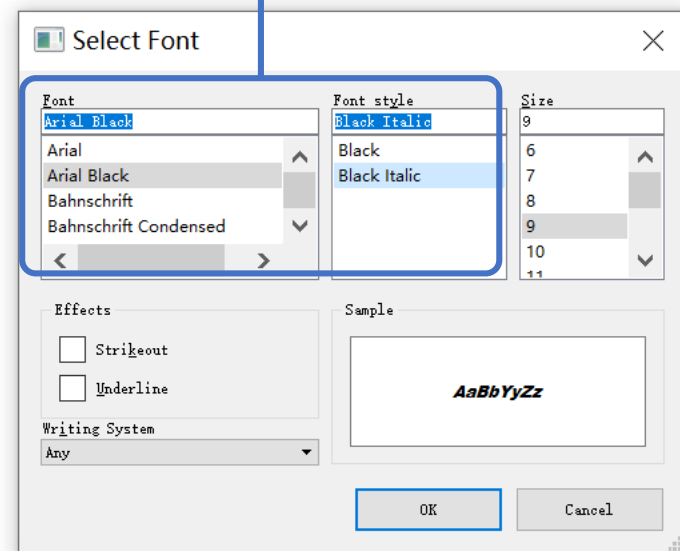
示例7 常用对话框介绍

- 字体对话框

```
// 字体对话框
void MyWidget::on_pushButton_3_clicked()
{
    // ok用于标记是否按下了“OK”按钮
    bool ok;
    QFont font = QFontDialog::getFont(&ok, this);
    // 如果按下“OK”按钮，那么让“字体对话框”按钮使用新字体
    // 如果按下“Cancel”按钮，那么输出信息
    if (ok) ui->pushButton_3->setFont(font);
    else qDebug() << tr("没有选择字体!");
}
```



1. 选择某种字体



2. 对应文字的字体发生改变

示例7 常用对话框介绍

- 输入对话框

```
66
67 // 输入对话框
68 void MyWidget::on_pushButton_4_clicked()
69 {
70     bool ok;
71     // 获取字符串
72     QString string =
73         QInputDialog::getText(this, tr("输入字符串对话框"),
74                               tr("请输入用户名: "),
75                               QLineEdit::Normal, tr("admin"), &ok);
76     if(ok) qDebug() << "string:" << string;
77     // 获取整数
78     int value1 = QInputDialog::getInt(this, tr("输入整数对话框"),
79                                       tr("请输入-1000到1000之间的数值"), 100, -1000, 1000, 10, &ok);
79     if(ok) qDebug() << "value1:" << value1;
80     // 获取浮点数
81     double value2 = QInputDialog::getDouble(this, tr("输入浮点数对话框"),
82                                              tr("请输入-1000到1000之间的数值"), 0.00, -1000, 1000, 2, &ok);
83     if(ok) qDebug() << "value2:" << value2;
84     QStringList items;
85     items << tr("条目1") << tr("条目2");
86     // 获取条目
87     QString item =
88         QInputDialog::getItem(this, tr("输入条目对话框"),
89                               tr("请选择或输入一个条目"), items, 0, true, &ok);
90     if(ok) qDebug() << "item:" << item;
91 }
92
```

对话框1：输入字符串对话框

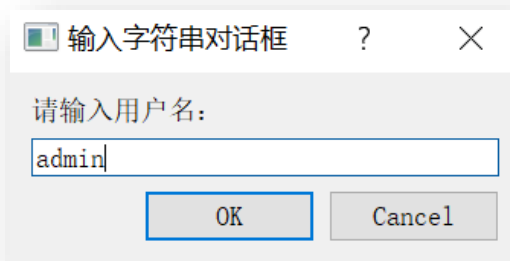
对话框2：输入整数/
浮点数对话框

对话框3：输入条目对话框

示例7 常用对话框介绍

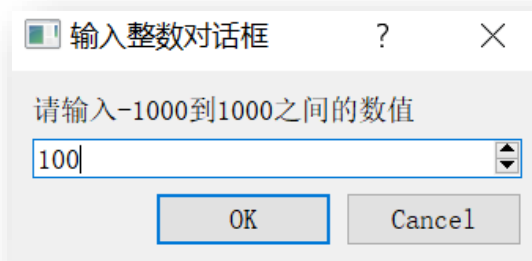
- 输入对话框

输入字符串对话框



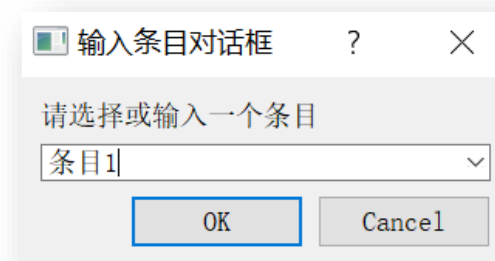
A screenshot of a standard Windows-style input dialog box. The title bar reads "输入字符串对话框" (Input String Dialog Box). The main text area contains the prompt "请输入用户名:" (Please enter a username:). Below the prompt is a single-line text input field containing the text "admin". At the bottom of the dialog are two buttons: "OK" and "Cancel".

输入整数对话框



A screenshot of a standard Windows-style input dialog box for integers. The title bar reads "输入整数对话框" (Input Integer Dialog Box). The main text area contains the prompt "请输入-1000到1000之间的数值" (Please enter a value between -1000 and 1000). Below the prompt is a numeric input field with a spinner control on the right, containing the value "100". At the bottom of the dialog are two buttons: "OK" and "Cancel".

输入条目对话框



A screenshot of a standard Windows-style input dialog box for selecting an item. The title bar reads "输入条目对话框" (Input Item Dialog Box). The main text area contains the prompt "请选择或输入一个条目" (Please select or enter an item). Below the prompt is a list box with a dropdown arrow on the right, showing the selected item "条目1" (Item 1). At the bottom of the dialog are two buttons: "OK" and "Cancel".

示例7 常用对话框介绍

- 消息对话框

```
// 消息对话框
```

```
void MyWidget::on_pushButton_5_clicked()
{
```

```
    // 问题对话框
```

```
    int ret1 = QMessageBox::question(this, tr("问题对话框"),  
                                     tr("你了解Qt吗? "), QMessageBox::Yes, QMessageBox::No);
```

```
    if(ret1 == QMessageBox::Yes) qDebug() << tr("问题!");
```

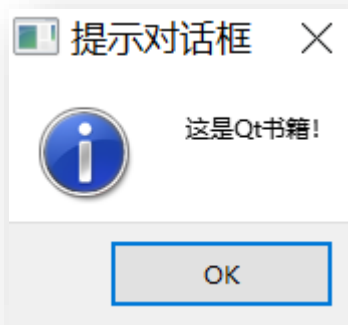
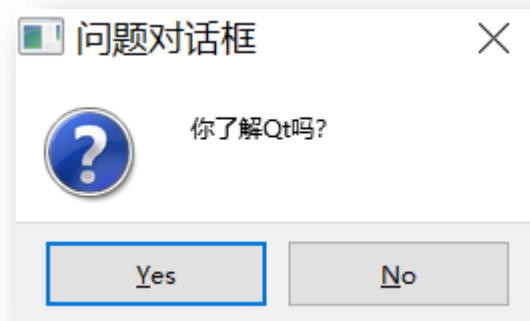
```
    // 提示对话框
```

```
    int ret2 = QMessageBox::information(this, tr("提示对话框"),  
                                       tr("这是Qt书籍! "), QMessageBox::Ok);
```

```
    if(ret2 == QMessageBox::Ok) qDebug() << tr("提示!");
```

添加问题
对话框

添加提示对话框



示例7 常用对话框介绍

- 消息对话框

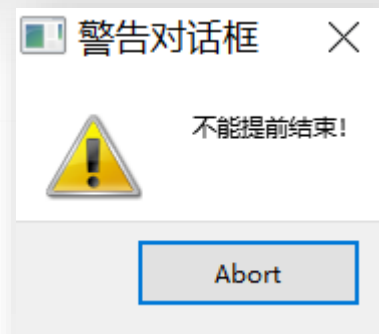
```
// 警告对话框
int ret3 = QMessageBox::warning(this, tr("警告对话框"),
                                tr("不能提前结束!"), QMessageBox::Abort);
if(ret3 == QMessageBox::Abort) qDebug() << tr("警告!");

// 错误对话框
int ret4 = QMessageBox::critical(this, tr("严重错误对话框"),
                                  tr("发现一个严重错误! 现在要关闭所有文件!"), QMessageBox::YesAll);
if(ret4 == QMessageBox::YesAll) qDebug() << tr("错误");

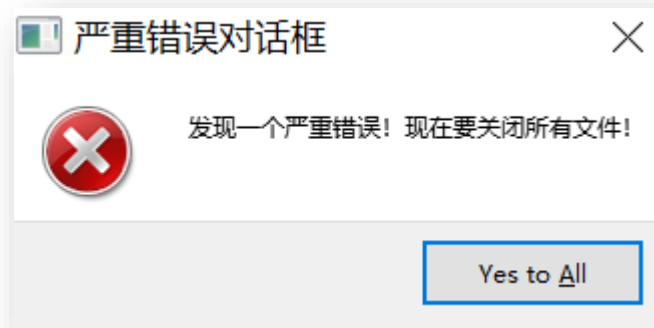
// 关于对话框
QMessageBox::about(this, tr("关于对话框"),
                   tr("yafeilinux致力于Qt及Qt Creator的普及工作!"));

}
```

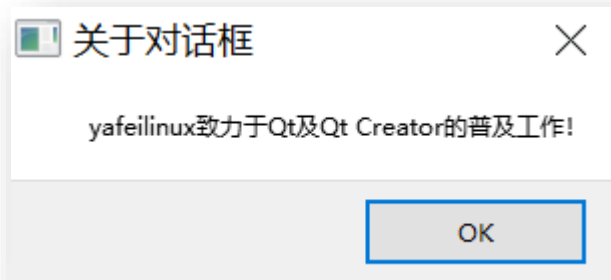
添加警告对话框



添加错误对话框



添加关于对话框

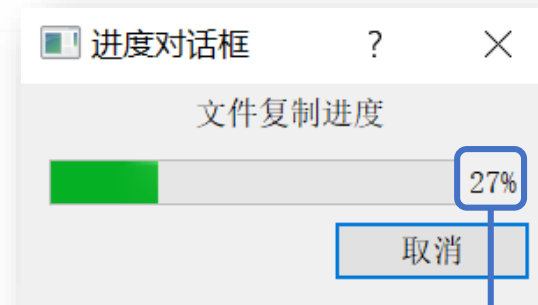


示例7 常用对话框介绍

- 进度对话框

```
118 // 进度对话框
119 void MyWidget::on_pushButton_6_clicked()
120 {
121     QProgressDialog dialog(tr("文件复制进度"),
122                             tr("取消"), 0, 50000, this);
123     dialog.setWindowTitle(tr("进度对话框")); // 设置窗口标题
124     dialog.setWindowModality(Qt::WindowModal); // 将对话框设置为模态
125     dialog.show();
126     for(int i=0; i<50000; i++) {
127         // 演示复制进度
128         qDebug() << i;
129         dialog.setValue(i); // 设置进度条的当前值
130         QCoreApplication::processEvents(); // 避免界面冻结
131         if(dialog.wasCanceled()) break; // 按下取消按钮则中断
132     }
133     dialog.setValue(50000); // 这样才能显示100%，因为for循环中少加了一个数
134     qDebug() << tr("复制结束！");
135 }
136
```

不断向dialog发消息，阻塞并更新dialog进度



根据setValue函数更新进度

示例7 常用对话框介绍

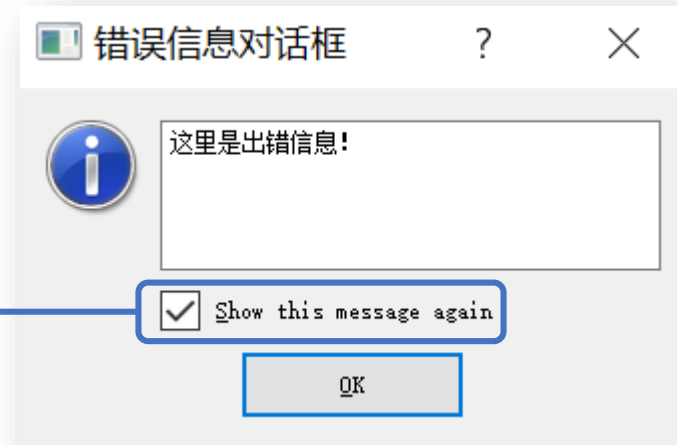
- 错误信息对话框

```
137 // 错误信息对话框
138 void MyWidget::on_pushButton_7_clicked()
139 {
140     errorDlg->setWindowTitle(tr("错误信息对话框"));
141     errorDlg->showMessage(tr("这里是出错信息!"));
142 }
```

设置对话框标题

设置错误信息

该选项会更新成员变量errorDlg状态，
从而决定是否不再显示该信息



示例7 常用对话框介绍

- 向导对话框

```
145
146 ▼ QWizardPage * MyWidget::createPage1() // 向导页面1
147 {
148     QWizardPage *page = new QWizardPage;
149     page->setTitle(tr("介绍"));
150     return page;
151 }
152 ▼ QWizardPage * MyWidget::createPage2() // 向导页面2
153 {
154     QWizardPage *page = new QWizardPage;
155     page->setTitle(tr("用户选择信息"));
156     return page;
157 }
158 ▼ QWizardPage * MyWidget::createPage3() // 向导页面3
159 {
160     QWizardPage *page = new QWizardPage;
161     page->setTitle(tr("结束"));
162     return page;
163 }
164
165 // 向导对话框
166 ▼ void MyWidget::on_pushButton_8_clicked()
167 {
168     QWizard wizard(this);
169     wizard.setWindowTitle(tr("向导对话框"));
170     wizard.addPage(createPage1()); // 添加向导页面
171     wizard.addPage(createPage2());
172     wizard.addPage(createPage3());
173     wizard.exec();
174 }
```

→ 创建多个向导页面

→ 添加多个向导页面

示例7 常用对话框介绍

- 向导对话框

