第5章

对话框设计

对话框是GUI应用程序中不可或缺的界面组件，它一般以顶层窗体的形式出现在程序的最上层，用于实现短期任务或简洁的用户交互。Qt中的对话框由QDialog类或其子类表示，可以通过它们或其派生类创建自定义的对话框，也可以直接使用Qt预定义的标准对话框。另外，除了QDialog对话框之外，在实际编程过程中还经常使用一些直接从QWidget类或其子类QFrame继承来的窗体，如QSplitter分割窗体、QStackedWidget堆栈窗体、QSplashScreen闪屏窗体和QMdiSubWindow多窗体等。

本章介绍Qt中的对话框以及一些其他常用窗体的设计方法，主要包括自定义对话框、标准对话框、分割窗体、MDI窗体等。

5.1 对话框相关Qt类

在对话框的设计过程中会涉及很多的Qt类，如QWidget类、QDialog类和QDialogButtonBox类等。QWidget类是Qt中所有界面组件的基类，在第3章中已经对其进行了详细的讲解，下面简单介绍QDialog类和QDialogButtonBox类。

5.1.1 QDialog类

QDialog是Qt对话框的基类，它继承自QWidget类。所以，对话框也是属于Qt窗体中的一种类型。QDialog类的继承关系如第3章的图3.1所示。

在QDialog类中，继承或定义了很多的属性、函数、信号和槽函数，用于实现对话框几何尺寸和风格特性等静态属性的存储、动态特性的设置，以及数据的传递等操作。QDialog类的部分成员函数及功能描述如表5.1所示。

表 5.1　QDialog 类的部分成员函数及功能描述

成 员 函 数	功 能 描 述
QDialog()	构造对话框对象
isSizeGripEnabled()、setSizeGripEnabled()	处理对话框的 sizeGripEnabled 属性。该属性控制对话框的尺寸调节器(QSizeGrip 对象),启用该属性时,将调节器放置在对话框的右下角。默认情况下,调节器处于禁用状态
result()、setResult()	处理对话框的 result 属性。该属性表示模态对话框的结果,即 Accepted 或 Rejected
setModal()	设置对话框的 modal 属性。该属性确定对话框的 show()函数是否应以模态或非模态方式弹出对话框
accept()	槽函数。隐藏模态对话框并将结果代码设置为 Accepted
done(int r)	槽函数。关闭对话框并将其结果代码设置为 r,此时会发射 finished()信号并传递参数 r
exec()	槽函数。将对话框显示为模态对话框,直到用户关闭为止。该函数返回对话框的 DialogCode 结果
open()	槽函数。将对话框显示为窗口模式,并立即返回
reject()	槽函数。隐藏模态对话框并将结果代码设置为 Rejected
accepted()	信号函数。当对话框被用户接受时,或者通过使用 QDialog::accepted 参数调用 accept()或 done()函数时发射此信号
finished()	信号函数。当用户或通过调用 done()、accept()或 reject()函数设置对话框的结果代码时发射此信号
rejected()	信号函数。当隐藏模态对话框并将结果代码设置为 Rejected 时发射此信号

QDialog 类的使用非常简单,下面给出一段示例代码。

```
//教材源码 code_5_1_1_1\widget.cpp
QDialog * dlg = new QDialog(this);                    //创建对话框对象
dlg->setWindowTitle(tr("对话框示例"));                 //设置标题
dlg->setFixedSize(320, 240);                          //设置对话框为固定大小
//设置对话框中的 OK 和 Cancel 按钮
QDialogButtonBox * buttonBox = new QDialogButtonBox(dlg);
buttonBox->setGeometry(QRect(10, 200, 301, 32));
buttonBox->setOrientation(Qt::Horizontal);
buttonBox->setStandardButtons(QDialogButtonBox::Cancel | QDialogButtonBox::
OK);
//关联信号和槽
connect(buttonBox,&QDialogButtonBox::accepted,dlg,&QDialog::accept);
connect(buttonBox,&QDialogButtonBox::rejected,dlg,&QDialog::reject);
dlg->exec();                                          //以模态方式打开对话框
QString str;
if(dlg->result()){
    str = "你点击了 OK 按钮!";
}else{
    str = "你点击了 Cancel 按钮!";
}
QMessageBox::information(this,tr("提示信息"),str);
delete buttonBox;                                     //删除按钮框对象
delete dlg;                                           //删除对话框对象
```

上述代码片段创建的对话框如图 5.1 所示。

图 5.1　对话框示例

在上述示例代码中,使用了 QDialogButtonBox 类创建对话框中的 OK 和 Cancel 标准按钮,并实现对话框的 accept()和 reject()操作。当然,也可以使用 QPushButton 类实现这两个按钮的功能。

5.1.2　QDialogButtonBox 类

QDialogButtonBox 类用于表示对话框上的按钮框,它会以适合当前窗体样式的布局显示按钮。QDialogButtonBox 类继承自 QWidget 类,其部分成员函数及功能描述如表 5.2 所示。

表 5.2　QDialogButtonBox 类的部分成员函数及功能描述

成 员 函 数	功 能 描 述
QDialogButtonBox()	构造对象
addButton()	将给定按钮添加到具有指定角色的按钮框中。如果角色无效,则不会添加该按钮
button()	返回与标准按钮对应的 QPushButton 对象指针,如果此按钮框中没有标准按钮,则返回 nullptr 空指针
buttonRole()	返回指定按钮的按钮角色或 InvalidRole
buttons()	返回已添加到按钮框的所有按钮的列表
centerButtons()、setCenterButtons()	处理按钮框的 centerButtons 属性。该属性确定按钮框中的按钮是否居中
clear()	清除按钮框,删除其中的所有按钮
orientation()、setOrientation()	处理按钮框的 orientation 属性。该属性控制按钮框的方向,默认为水平(即按钮并排布置)
removeButton()	从按钮框中移除某个按钮
standardButtons()、setStandardButtons()	处理按钮框的 standarButtons 属性。该属性控制按钮框使用哪些标准按钮
standardButton()	返回与给定按钮相对应的标准按钮枚举值
accepted()	信号函数。当单击按钮框中的以 AcceptRole 或 YesRole 角色定义的按钮时发射此信号
clicked()	信号函数。当单击按钮框内的某个按钮时发射此信号

续表

成 员 函 数	功 能 描 述
helpRequested()	信号函数。当单击按钮框中的以 HelpRole 角色定义的按钮时发射此信号
rejected()	信号函数。当单击按钮框中的以 RejectRole 或 NoRole 角色定义的按钮时发射此信号

QDialogButtonBox 类的使用非常简单，先构造一个 QDialogButtonBox 对象，然后调用其成员函数完成相应功能即可。

5.2　自定义对话框

Qt 中的对话框分为模态对话框和非模态对话框。所谓模态对话框，就是运行时会阻塞同一应用程序中其他窗体的输入的对话框。例如，MS Word 软件中的“打开文件”对话框，当该对话框运行时，用户不能对除此之外的窗体进行操作。与此相反的是非模态对话框，用户可以在运行该对话框的同时继续进行程序的其他操作，如 MS Word 软件中的“查找和替换”对话框。

5.2.1　模态对话框

Qt 中对话框有两种级别的模态，即应用程序级别的模态和窗体级别的模态，默认是应用程序级别的模态。所谓应用程序级别的模态，是指当该种模态的对话框出现时，用户必须首先与对话框进行交互，直到关闭对话框，然后才能访问应用程序的其他窗体；而窗体级别的模态仅阻塞与对话框关联的窗体，它允许用户与其他非关联窗体进行交互。

Qt 使用 QDialog::exec()函数实现应用程序级别的模态对话框，使用 QDialog::open()函数实现窗体级别的模态对话框。

【例 5.1】 编写一个 Qt 应用程序，在其主窗体中添加一个文本编辑器，通过自定义的颜色对话框设置文本编辑器中的字体颜色。程序运行结果如图 5.2 所示。

扫一扫

视频讲解

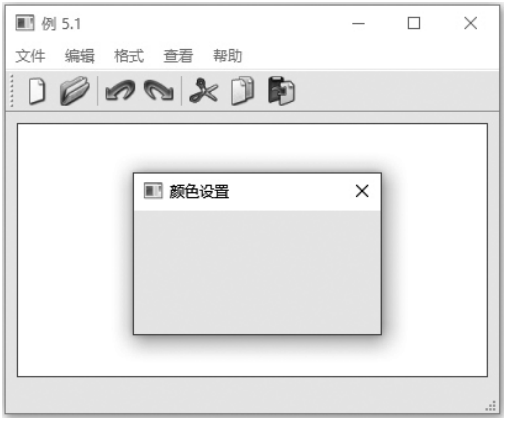

图 5.2　例 5.1 程序运行结果

为了缩减篇幅，也为了让应用程序功能相对完整，本例在例 4.6 应用程序的基础上进行

设计。

(1) 将教材源码中的 examp4_6 项目复制到第 5 章的实例目录 chap05 中,并将项目文件夹名称修改为 examp5_1,项目文件名称修改为 examp5_1.pro。

(2) 在 Qt Creator 中打开 examp5_1 项目,将其主框架标题修改为“例 5.1”,并在“格式”主菜单下添加一个名为“颜色”的 Action。

(3) 为项目添加一个基于 QDialog 类的界面类,类名为 ColorDialog。这里使用 Qt Creator 向导同时生成类文件及界面文件,如图 5.3 所示。

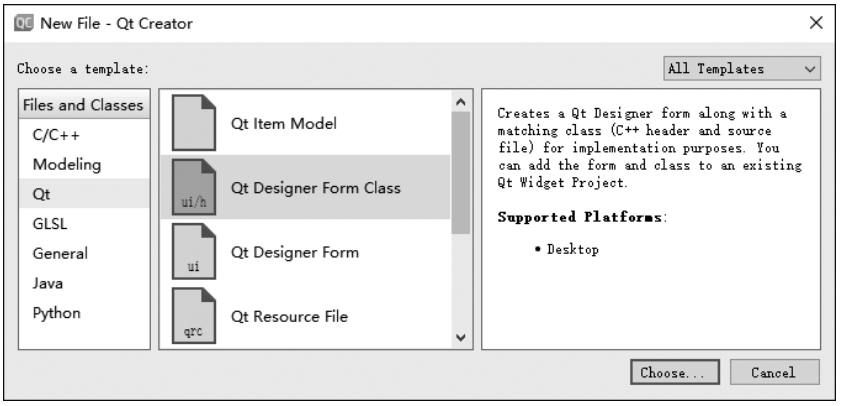

图 5.3 添加新文件

(4) 将新对话框的标题设置为“颜色对话框”,大小设置为 200×100。

(5) 为步骤(2)中的“颜色”Action 添加信号与槽,并在槽函数中添加代码。

```
void MainWindow::on_action_color_triggered()
{
    ColorDialog * dlg = new ColorDialog(this);      //语句 1
    dlg -> exec();                                  //语句 2
}
```

上述代码中的语句 1 创建一个 dlg 对话框对象,并设置应用程序主窗体为其父窗体;语句 2 通过 dlg 对象调用 QDialog 的 exec()成员函数,显示对话框界面。

(6) 构建并运行程序。执行程序的“格式”→“颜色”菜单命令,结果如图 5.2 所示。

从程序运行结果可以看出,当自定义的“颜色设置”对话框被激活后,程序的主框架窗体随即变为灰色。此时,不能对主窗体中的任何元素进行操作。

如果把步骤(5)中的语句 2 代码修改为

```
dlg -> open();
```

运行程序会得到相同的结果。

上面的示例说明,在 Qt 中使用 QDialog::exec()和 QDialog::open()函数均能实现模态对话框。至于这两个函数的区别,限于篇幅,这里不再展开讨论,请大家参考相关的技术文档自行思考。

5.2.2 非模态对话框

Qt 使用 QDialog::show()函数实现非模态对话框。将例 5.1 步骤(5)中的语句 2 修

改为

```
dlg -> show();     //语句 2
```

即可实现“颜色设置”对话框的非模态显示。程序运行结果如图 5.4 所示。

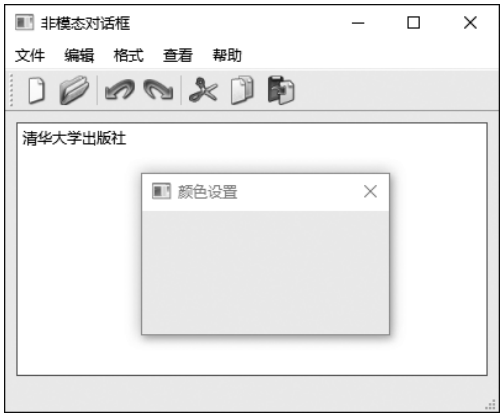

图 5.4　非模态对话框示例

从程序运行的结果可以看出，当“颜色对话框”被激活时，是可以对主窗体进行操作的。下面尝试将例 5.1 步骤(5)的代码修改为

```
ColorDialog dlg(this);          //语句 1
dlg.show();                     //语句 2
```

构建并运行程序。执行应用程序的“格式”→“颜色”菜单命令，可以看到“颜色设置”对话框竟然一闪而过。

这是因为 QDialog::show()函数并没有阻塞当前线程，当对话框显示出来后，show()函数会立即返回，继续执行语句 2 后面的代码。注意，这里 dlg 是建立在栈上的，show()函数运行结束后，MainWindow::on_action_color_triggered()槽函数也执行完成了。此时，作为槽函数局部变量的 dlg 随即被析构清除，因此“颜色设置”对话框瞬间就消失了。所以，要实现非模态对话框，对话框对象必须建立在堆上，也就是要用 new 方法构建对话框对象，然后调用 QDialog::show()函数将其显示即可。

5.2.3　数据交换

在应用程序中使用对话框，往往都是想通过它传递数据，也就是使用它与主窗体之间进行数据的交换。

1. 获取模态对话框数据

在例 5.1 应用程序中，我们只是实现了对话框的模态与非模态显示，并没有真正通过该对话框设置主窗体中文本编辑器的字体颜色。下面通过实例说明如何从模态对话框中获取数据。

【例 5.2】 继续实现例 5.1 应用程序功能，使用“颜色设置”对话框设置主窗体中文本编辑器字体颜色。程序运行结果如图 5.5 所示。

扫一扫

视频讲解

(1) 复制 examp5_1 项目到第 5 章的实例目录 chap05 中，并将项目文件夹名称修改为 examp5_2，项目文件名称修改为 examp5_2.pro。

图 5.5 例 5.2 程序运行结果

(2) 在 Qt Creator 中打开 examp5_2 项目,将其主框架标题修改为"例 5.2",并在"颜色设置"对话框中添加 3 个 QFrame 对象和 3 个 QRadioButton 对象。3 个 QFrame 对象的名称分别为 redFrame、greenFrame 和 blueFrame;3 个 QRadioButton 对象的名称分别为 redBtn、greenBtn 和 blueBtn。

(3) 在 ColorDialog 中添加一个 private 访问权限的名称为 color 的 QColor 对象,用于存储用户选择的颜色;另外,再添加一个 public 访问权限的 getColor()成员函数,该函数返回一个 QColor 颜色对象。

(4) 在类 ColorDialog 构造方法中编写代码,对 3 个 QFrame 对象初始化,将它们填充为相应的色块。代码如下。

```
ui->red->setAutoFillBackground(true);
ui->red->setPalette(QPalette(QColor(255,0,0)));
ui->green->setAutoFillBackground(true);
ui->green->setPalette(QPalette(QColor(0,255,0)));
ui->blue->setAutoFillBackground(true);
ui->blue->setPalette(QPalette(QColor(0,0,255)));
color = QColor(0,0,0);
```

(5) 编写 ColorDialog 类的 getColor()成员函数的实现代码。

```
QColor ColorDialog::getColor(){
if(ui->redBtn->isChecked()){
    color = QColor(255,0,0);
}
if(ui->greenBtn->isChecked()){
    color = QColor(0,255,0);
}
if(ui->blueBtn->isChecked()){
    color = QColor(0,0,255);
}
return color;
}
```

(6) 为 ColorDialog 类的 3 个 QRadioButton 对象添加 clicked 信号与槽函数,并编写槽函数代码。

```
void ColorDialog::on_redBtn_clicked()
{
    ui->redBtn->setChecked(true);
    this->accept();
}
void ColorDialog::on_greenBtn_clicked()
{
    ui->greenBtn->setChecked(true);
    this->accept();
}
void ColorDialog::on_blueBtn_clicked()
{
    ui->blueBtn->setChecked(true);
    this->accept();
}
```

(7) 修改主窗体“格式”→“颜色”菜单命令槽函数代码,完成文本编辑器文字颜色的设置。代码如下。

```
void MainWindow::on_action_color_triggered()
{
    ColorDialog * dlg = new ColorDialog(this);
    if(dlg->exec() == QDialog::Accepted){
        ui->textEdit->setTextColor(dlg->getColor());
    }
}
```

(8) 构建并运行程序。程序运行后的测试结果如图 5.5 所示。

从上述实例代码可以看出,获取模态对话框中的数据一般使用对话框的公有成员函数。模态对话框使用 QDialog::exec()函数来实现,该函数的真正含义是开启一个新的事件循环(参见第 6 章)。所谓事件循环,可以理解成一个无限循环。Qt 在开启了事件循环之后,系统发出的各种事件才能够被程序监听到。

2. 获取非模态对话框数据

非模态对话框使用 QDialog::show()函数来实现,与 QDialog::exec()函数不同的是,show()函数没有返回值,它的作用仅仅是将对话框显示出来而已。因此,从非模态对话框中获取数据不能采用 dlg->show()= =QDialog::Accepted 这样的代码。

从非模态对话框中获取数据最好采用信号与槽的通信机制,通过信号函数与槽函数进行数据的传递。例如,对于例 5.2 中的应用程序,建立“颜色设置”对话框和主窗体之间信号与槽通信机制,当对话框执行某个操作(如单击单选按钮)时,将对话框中需要传送的数据放到信号中发射出去,主窗体中的槽函数接收这一信号并进行相应的处理,从而实现数据从对话框向主窗口的传递。

【例 5.3】 使用非模态对话框实现例 5.2 中应用程序功能。程序运行结果如图 5.6 所示。

(1) 复制 examp5_2 项目到第 5 章的实例目录 chap05 中,并将项目文件夹名称修改为 examp5_3,项目文件名称修改为 examp5_3.pro。

(2) 在 Qt Creator 中打开 examp5_3 项目,将其主框架标题修改为“例 5.3”。

(3) 在 ColorDialog 类的头文件 colordialog.h 中添加信号声明代码。

图 5.6 例 5.3 程序运行结果

```
signals:
    void sendData(QColor);
```

(4) 打开 ColorDialog 类的实现文件 colordialog.cpp,修改其中 3 个单选按钮的槽函数代码。

```
void ColorDialog::on_redBtn_clicked()
{
    emit sendData(QColor(255,0,0));
}
void ColorDialog::on_greenBtn_clicked()
{
    emit sendData(QColor(0,255,0));
}
void ColorDialog::on_blueBtn_clicked()
{
    emit sendData(QColor(0,0,255));
}
```

(5) 在 MainWindow 类的头文件 mainwindow.h 中添加槽函数声明代码。

```
private slots:
    void receiveData(QColor);
```

(6) 打开 MainWindow 类的实现文件 mainwindow.cpp,添加槽函数的实现代码。

```
void MainWindow::receiveData(QColor c){
    ui->textEdit->setTextColor(c);
}
```

(7) 修改 mainwindow.cpp 文件中 on_action_color_triggered()槽函数代码。

```
void MainWindow::on_action_color_triggered()
{
    ColorDialog *dlg = new ColorDialog(this);
    connect(dlg,SIGNAL(sendData(QColor)),this,SLOT(receiveData(QColor)));
    dlg->show();
}
```

(8) 构建并运行程序,结果如图 5.6 所示。注意图中“颜色设置”对话框中单选按钮的状态与图 5.5 中的区别。

5.3　标准对话框

Qt 为应用程序设计提供了一些常用的标准对话框，如文件对话框、颜色对话框、字体对话框、输入对话框以及消息对话框等，用于实现应用程序中的一些常用功能。另外，这些标准对话框也为应用程序提供了一致的界面观感。

Qt 为每个标准对话框定义了相关的类，这些类全部继承于 QDialog 类，如第 3 章的图 3.1 所示。下面对几个常用的标准对话框类进行简单的介绍，它们分别是 QColorDialog、QFileDialog、QFontDialog、QInputDialog 和 QMessageBox。

5.3.1　颜色对话框

颜色对话框用于选取颜色值，由 QColorDialog 类实现，其界面效果如图 5.7 所示。

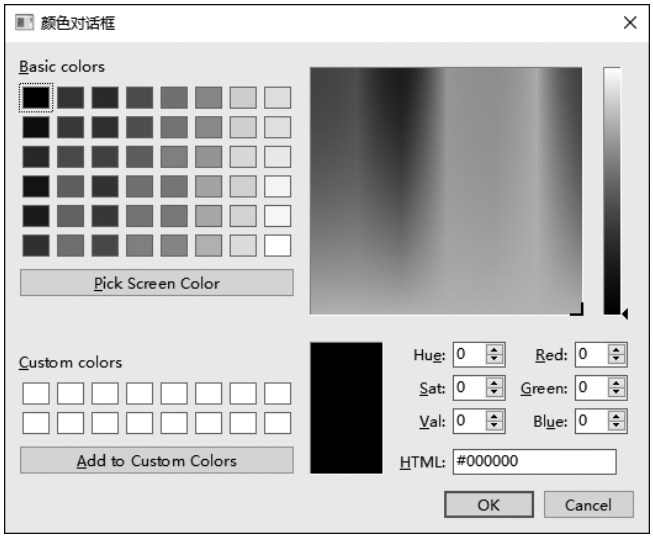

图 5.7　颜色对话框

颜色对话框的使用非常简单，打开对话框后，可以通过使用鼠标选择颜色色块、使用颜色拾取器在屏幕中拾取颜色或自定义颜色分量值等方法获取颜色值。

【例 5.4】 使用 Qt 的颜色对话框实现例 5.1 应用程序功能。

扫一扫

视频讲解

(1) 复制 examp5_1 项目到第 5 章的实例目录 chap05 中，并将项目文件夹名称修改为 examp5_4，项目文件名称修改为 examp5_4.pro。

(2) 在 Qt Creator 中打开 examp5_4 项目，删除 colordialog.ui 界面文件，同时删除该界面文件对应的 colordialog.h 和 colordialog.cpp 类文件。

(3) 打开 mainwindow.cpp 文件，修改 on_action_color_triggered()槽函数中的代码。

```
void MainWindow::on_action_color_triggered()
{
    //打开颜色对话框方法一
    QColor c = QColorDialog::getColor(Qt::black,this,tr("颜色对话框"));
    /* 打开颜色对话框方法二
```