

qt实战之表格控制 及对话框

表格控制及对话框

Widget

	姓名	性别	出生日期	民族	分数
1	张三0	男	1982-10-10	汉族	51
2	张三1	男	1982-10-10	汉族	52
3	张三2	男	1982-10-10	汉族	53
4	张三3	男	1982-10-10	汉族	54
5	张三4	男	1982-10-10	汉族	55
6	张三5	男	1982-10-10	汉族	56
7	张三6	男	1982-10-10	汉族	57
8	张三7	男	1982-10-10	汉族	58

初始化

增加

清空

删除

姓名

性别

出生日期

民族

分数

Widget

	姓名	性别	出生日期	民族	分数
1	张三0	男	1982-10-10	汉族	51
2	张三1	男	1982-10-10	汉族	52
3	张三2	男	1982-10-10	汉族	53
4	张三3	男	1982-10-10	汉族	54
5	张三4	男	1982-10-10	汉族	55
6	张三5	男	1982-10-10	汉族	56
7	张三6	男	1982-10-10	汉族	57
8	张三7	男	1982-10-10	汉族	58

信息

 张三1,男,1982-10-10,汉族,52

OK

初始化

增加

清空

删除

姓名

性别

出生日期

民族

分数

案例知识点

- 1. QTableWidgetItem表格组件
- 2. QTableWidgetItem表格项目
- 3. 其他数据类型的综合应用
- 4. QMessageBox对话框

QTableWidget

QTableWidget 是 Qt 中的表格组件类。在窗体上放置一个 QTableWidget 组件后,可以在 Property Editor 里对其进行属性设置,双击这个组件,可以打开一个编辑器,对其 Column、Row 和 Item 进行编辑。一个 QTableWidget 组件的界面基本结构如图 4-17 所示,这个表格设置为 6 行 5 列。

表格的第 1 行称为行表头,用于设置每一列的标题,第 1 列称为列表头,可以设置其标题,但一般使用缺省的标题,即为行号。行表头和列表头一般是不可编辑的。

除了行表头和列表头之外的表格区域是内容区,内容区是规则的网格状,如同一个二维数组,每个网格单元称为一个单元格。每个单元格有一个行号、列号,图 4-17 表示了行号、列号的变化规律。

在 QTableWidget 表格中,每一个单元格是一个 QTableWidgetItem 对象,可以设置文字内容、字体、前景色、背景色、图标,也可以设置编辑和显示标记。每个单元格还可以存储一个 QVariant 数据,用于设置用户自定义数据。

	列1	列2	列3	列4	列5
1	0行, 0列	0行, 1列			0行, 4列
2	1行, 0列				
3	2行, 0列				
4					
5					
6	5行, 0列				5行, 4列

图 4-17 一个 QTableWidget 表格的基本结构和工作区的行、列索引号

设置表格头部

QTableWidgetItem是一个表示单元格的对象

```
void Widget::init_table() {  
    QTableWidgetItem *headerItem;  
  
    QStringList headerText;  
    headerText.append("姓名");  
    headerText.append("性别");  
    headerText.append("出生日期");  
    headerText.append("民族");  
    headerText.append("分数");  
  
    ui->tableWidget->setColumnCount(headerText.count());  
    for(int i=0; i< ui->tableWidget->columnCount(); i++) {  
        headerItem = new QTableWidgetItem(headerText.at(i));  
  
        QFont font = headerItem->font();  
        font.setBold(true);  
        font.setPointSize(16);  
        headerItem->setTextColor(Qt::blue);  
        headerItem->setFont(font);  
  
        ui->tableWidget->setHorizontalHeaderItem(i, headerItem);  
    }  
}
```

将表头内容存储到一个List中

设置列的数量

设置单元格的字体

初始化表格数据

```
void Widget::on_initButton_clicked()
{
    std::cout << "ext" << std::endl;

    QDate birthday;
    float score = 50;
    ui->tableWidget->setRowCount(10);
    for(int i=0; i<10;i++) {
        birthday.setDate(1982,10,10);
        QString name = "张三" + QString::asprintf("%d", i);
        create_row(i, name, "男", birthday, "汉族", ++score);
        birthday = birthday.addDays(1);
    }
}
```



设置表格行数

初始化表格数据

```
void Widget::create_row(int rowNo, QString name, QString sex, QDate birthday, QString national, float score)
{
    QTableWidgetItem *item;

    // 姓名
    item = new QTableWidgetItem(name, 1001);
    item->setTextAlignment(Qt::AlignHCenter | Qt::AlignVCenter);
    ui->tableWidget->setItem(rowNo, 0, item);

    // 性别
    item = new QTableWidgetItem(sex, 1002);
    item->setTextAlignment(Qt::AlignHCenter | Qt::AlignVCenter);
    ui->tableWidget->setItem(rowNo, 1, item);

    // 出生日期
    item = new QTableWidgetItem(birthday.toString("yyyy-MM-dd"), 1003);
    item->setTextAlignment(Qt::AlignHCenter | Qt::AlignVCenter);
    ui->tableWidget->setItem(rowNo, 2, item);

    // 民族
    item = new QTableWidgetItem(national, 1004);
    item->setTextAlignment(Qt::AlignHCenter | Qt::AlignVCenter);
    ui->tableWidget->setItem(rowNo, 3, item);

    // 分数
    QString str;
    item = new QTableWidgetItem(str.setNum(score), 1005);
    item->setTextAlignment(Qt::AlignHCenter | Qt::AlignVCenter);
    ui->tableWidget->setItem(rowNo, 4, item);
}
```

设置一个单元格

初始化表格数据

创建 QTableWidgetItem 使用的构造函数的原型为：

```
QTableWidgetItem::QTableWidgetItem(const QString &text, int type = Type)
```

其中，第一个参数作为单元格的显示文字，第二个参数作为节点的类型。

QTableWidgetItem 有一些函数对单元格进行属性设置，如下。


- setTextAlignment(int alignment): 设置文字对齐方式。
- setBackground(const QBrush &brush): 设置单元格背景颜色。
- setForeground(const QBrush &brush): 设置单元格前景色。
- setIcon(const QIcon &icon): 为单元格设置一个显示图标。
- setFont(const QFont &font): 为单元格显示文字设置字体。

```
ui->tableInfo->setItem(rowNo,MainWindow::colName,item);
```

其中，MainWindow::colName 是定义的枚举类型 FieldColNum 的一个常量值。

清空数据

```
void Widget::on_addButton_2_clicked()
{
    ui->tableWidget->clearContents();
    ui->tableWidget->setRowCount(0);
}
```



QTableWidget::clearContents()函数清除表格数据区的所有内容，但是不清除表头。

QTableWidget::rowCount()函数返回表格数据区的行数。

从输入控件增加一行

```
void Widget::on_addButton_clicked()
{
    int rowCount = ui->tableWidget->rowCount();
    ui->tableWidget->insertRow(rowCount);

    QString name = ui->nameEdit->text();
    QString sex = ui->sexEdit->text();
    QDate birthday = ui->dateEdit->date();
    QString national = ui->nationalEdit->text();
    QString score = ui->scoreBox->text();
    create_row(rowCount, name, sex, birthday, national, score.toFloat());
}
```

增加一个空行

从输入控件中获得一行数据

删除一行

```
void Widget::on_deleteButton_clicked()
{
    int currentRow = ui->tableWidget->currentRow();
    ui->tableWidget->removeRow(currentRow);
}
```

获得当前选中行

QTableWidget 处理行操作的函数如下。

- `insertRow(int row)`: 在行号为 `row` 的行前面插入一行，如果 `row` 等于或大于总行数，则在表格最后添加一行。`insertRow()` 函数只是插入一个空行，不会为单元格创建 `QTableWidgetItem` 对象，需要手工为单元格创建。
- `removeRow(int row)`: 删除行号为 `row` 的行。

间隔行底色

- 间隔行底色

setAlternatingRowColors()函数可以设置表格的行是否用交替底色显示，若为交替底色，则间隔的一行会用灰色作为底色。具体底色的设置需要用 styleSheet，在 16.2 节有介绍。

```
void MainWindow::on_chkBoxRowColor_clicked(bool checked)
{
    ui->tableInfo->setAlternatingRowColors(checked);
}
```

	姓名	性别	出生日期	民族	分数	
1	张三0	男	1982-10-10	汉族	51	
2	张三1	男	1982-10-10	汉族	52	
3	张三2	男	1982-10-10	汉族	53	
4	张三3	男	1982-10-10	汉族	54	
5	张三4	男	1982-10-10	汉族	55	
6	张三5	男	1982-10-10	汉族	56	
7	张三6	男	1982-10-10	汉族	57	
8	张三7	男	1982-10-10	汉族	58	
9	张三8	男	1982-10-10	汉族	59	

选择一行并显示

当鼠标在表格上单击单元格时，被选中的单元格是当前单元格。通过 `QTableWidget` 的 `currentColumn()`和 `currentRow()`可以获得当前单元格的列编号和行编号。

当前单元格发生切换时，会发射 `currentCellChanged()`信号和 `currentItemChanged()`信号，两个信号都可以利用，只是传递的参数不同。

```
void Widget::on_tableWidget_itemSelectionChanged()
{
    int currentRow = ui->tableWidget->currentRow();

    QTableWidgetItem *cellItem = ui->tableWidget->item(currentRow, 0);
    QString name = cellItem->text();

    cellItem = ui->tableWidget->item(currentRow, 1);
    QString sex = cellItem->text();

    cellItem = ui->tableWidget->item(currentRow, 2);
    QString birthday = cellItem->text();

    cellItem = ui->tableWidget->item(currentRow, 3);
    QString national = cellItem->text();

    cellItem = ui->tableWidget->item(currentRow, 4);
    QString score = cellItem->text();

    QMessageBox::information(this, "信息", name + "," + sex + "," + birthday + "," + national + "," + score , QMessageBox::Ok, QMessageBox::NoButton);
    //std::cout << data.toStdString() << std::endl;
}
```

通过item后的当前单元格，
再通过data或者text获得数据

QMessageBox对话框

消息对话框 QMessageBox 用于显示提示、警告、错误等信息，或进行确认选择，由几个静态函数实现这些功能（详见表 6-1）。其中 warning()、information()、critical()和 about()这几个函数的输入参数和使用方法相同，只是信息提示的图标有区别。例如，warning()的函数原型是：

```
StandardButton QMessageBox::warning(QWidget *parent, const QString &title, const  
QString &text, StandardButtons buttons = Ok, StandardButton defaultButton = NoButton)
```

其中，parent 是对话框的父窗口，指定父窗口之后，打开对话框时，对话框将自动显示在父窗口的上方中间位置；title 是对话框标题字符串；text 是对话框需要显示的信息字符串；buttons 是对话框提供的按钮，缺省只有一个 OK 按钮；defaultButton 是缺省选择的按钮，缺省表示没有选择。

warning()函数的返回结果是 StandardButton 类型。对话框上显示的按钮和缺省选中按钮也是 StandardButton 类型。

StandardButton 是各种按钮的定义，如 OK、Yes、No、Cancel 等，其枚举取值是 QMessageBox::Ok、QMessageBox::Cancel、QMessageBox::Close 等，详见 Qt 帮助文档中的 StandardButton 类型的说明。

QMessageBox对话框

对于 warning()、information()、critical()和 about()这几种对话框，它们一般只有一个 OK 按钮，且无须关心对话框的返回值。所以，使用缺省的按钮设置即可。例如，下面是程序中调用 QMessageBox 信息显示的代码，显示的几个对话框如图 6-3 所示。



图 6-3 QMessageBox 的几种消息提示对话框

QMessageBox对话框例子

```
void Dialog::on_btnMsgInformation_clicked()
{
    //information
    QString dlgTitle="information 消息框";
    QString strInfo="文件已经打开, 字体大小已设置";
    QMessageBox::information(this, dlgTitle, strInfo,
                             QMessageBox::Ok, QMessageBox::NoButton);
}

void Dialog::on_btnMsgWarning_clicked()
{
    // warning
    QString dlgTitle="warning 消息框";
    QString strInfo="文件内容已经被修改";
    QMessageBox::warning(this, dlgTitle, strInfo);
}
```


导入的其他头文件

```
widget.h
#ifndef WIDGET_H
#define WIDGET_H

#include <QWidget>
#include <QTableWidgetItem>
```

QTableWidgetItem头文件

```
具(I) 控件(W) 帮助(H)
widget.cpp

#include "widget.h"
#include "ui_widget.h"
#include "qdatetime.h"
#include "iostream"
#include <QMessageBox>
```

时间对象头文件

输出流头文件

对话框头文件