

去冰三分糖



搜索

找找看

使用总结



随笔分类 (37)

C++(5)

Qt(23)

不才(3)

操作系统(4)

风尘(2)

文本编辑控件。
函数，输入和编辑单行文本，比如撤销、恢复、剪切、粘贴
`echoMode()`，可以设置其属性，比如以密码的形式输入。
`xLength()` 限制，可以通过使用 `validator()` 或
限制它只能输入数字。在对同一个QLineEdit的validator或者
input mask进行替换时，最好先将它的validator或者input mask清除，以避免错误
发生。

与QLineEdit相关的一个类是QTextEdit，它允许多行文字以及富文本编辑。
我们可以使用 `setText()` 或者 `insert()` 改变其中的文本，通过 `text()` 获得文本，通过 `displayText()` 获得显示的文本，使用 `setSelection()` 或者 `selectAll()` 选中文本，选中的文本可以通过`cut()`、`copy()`、`paste()`进行剪切、复制和粘贴，使用 `setAlignment()` 设置文本的位置。
文本改变时会发出 `textChanged()` 信号；如果不是由`setText()`造成文本的改变，那么会发出`textEdit()`信号；鼠标光标改变时会发出`cursorPostionChanged()`信

号；当返回键或者回车键按下时，会发出returnPressed()信号。

当编辑结束，或者LineEdit失去了焦点，或者当返回/回车键按下时，editFinished()信号将会发出。

以上是Qt官方文档对QLineEdit的简要说明，下面根据个人经验，对一些常用的方法作说明：

H1 1.setPlaceholderText()设置提示文字



豆瓣电影的搜索输入框，没有输入任何字符时，显示“电影、影人、影院、电视剧”这些占位文字，对用户输入作相关提示。

```
echoLineEdit->setPlaceholderText("电影、影人、影院、电视剧");
```

H1 2.setEchoMode()设置模式



淘宝登录界面的一部分，用户名可以直接看到，密码一般都用小黑点掩盖。

```
switch (index) {  
    case 0:  
        //默认，输入什么即显示什么  
        echoLineEdit->setEchoMode(QLineEdit::Normal);  
        break;  
    case 1:  
        //密码，一般是用小黑点覆盖你所输入的字符  
        echoLineEdit->setEchoMode(QLineEdit::Password);  
        break;  
    case 2:  
        //编辑时输入字符显示输入内容，否则用小黑点代替  
        echoLineEdit->setEchoMode(QLineEdit::PasswordEchoOnEdit);  
        break;  
    case 3:  
        //任何输入都看不见（只是看不见，不是不能输入）
```

```
echoLineEdit->setEchoMode(QLineEdit::NoEcho);  
}
```

H1 3.setAlignment()设置文本位置

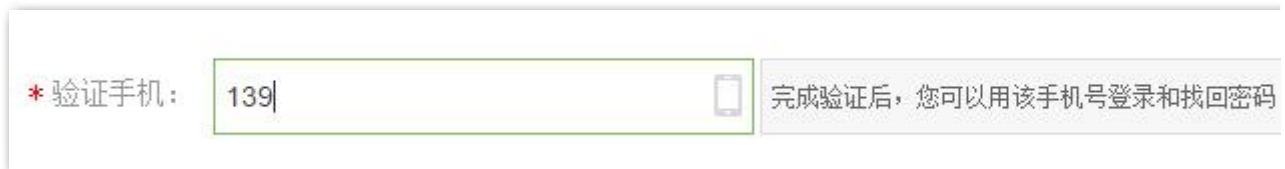
```
switch (index) {  
    case 0:  
        alignmentLineEdit->setAlignment(Qt::AlignLeft);  
        break;  
    case 1:  
        alignmentLineEdit->setAlignment(Qt::AlignCenter);  
        break;  
    case 2:  
        alignmentLineEdit->setAlignment(Qt::AlignRight);  
}
```

H1 4.setReadOnly()设置能否编辑

```
switch (index) {  
    case 0:  
        accessLineEdit->setReadOnly(false);  
        break;  
    case 1:  
        accessLineEdit->setReadOnly(true);  
}
```

H1 5.setValidator()对输入进行限制

这种方式的实质是通过正则表达式限制输入的内容。



比如上面的手机号输入框，控制其不能输入英文汉字等无关字符。

```
switch (index) {  
    case 0:  
        //无限制  
        validatorLineEdit->setValidator(0);  
        break;  
    case 1:  
        //只能输入整数  
        validatorLineEdit->setValidator(new QIntValidator(  
            validatorLineEdit));  
}
```

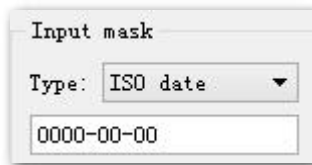
```

        break;
    case 2:
        //实例，只能输入-180到180之间的小数，小数点后最多两位（可用于附
        QDoubleValidator *pDfValidator = new QDoubleValidator(-18
        pDfValidator->setNotation(QDoubleValidator::StandardNotat
        validatorLineEdit->setValidator(pDfValidator);
    }

```

H1 6.setInputMask()对输入进行限制

通过限制格式限制输入，具体怎么格式化可以参考Qt助手。



```

switch (index) {
    case 0:
        inputMaskLineEdit->setInputMask("");
        break;
    case 1:
        inputMaskLineEdit->setInputMask("+99 99 99 99 99;_");
        break;
    case 2:
        inputMaskLineEdit->setInputMask("0000-00-00");
        inputMaskLineEdit->setText("00000000");
        inputMaskLineEdit->setCursorPosition(0);
        break;
    case 3:
        inputMaskLineEdit->setInputMask(">AAAAA-A AAAA-A AAAA-A AAAA
}

```

H1 7.setMaxLength()设置可以输入的最多字符数

```

//最多只能输入9个字符
echoLineEdit->setMaxLength(9);

```

H1 8.validator和inputmask的结合

比如纬度用“度:分:秒”的格式表示，分和秒的范围都是00-59，度的范围是-89到89。

```

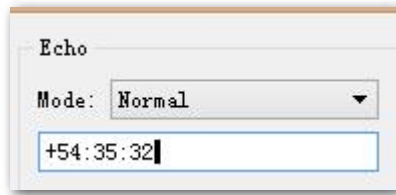
QRegExp rx("(-|\\+)?[0-8]\\d:[0-5]\\d:[0-5]\\d");
echoLineEdit->setValidator(new QRegExpValidator(rx, echoLineEdit)
echoLineEdit->setInputMask("#00:00:00;0");

```

```
echoLineEdit->setText("+00:00:00");
```



如果不控制输入，那么必须在输入后检查输入是否合法，但控制输入后的输入肯定是合法的，可以省去检查合法的繁琐步骤。只需使用正则表达式控制输入的度分秒范围，然后控制输入的格式。



一些测试代码供参考——

头文件：

```
#ifndef WINDOW_H
#define WINDOW_H

#include <QWidget>

QT_BEGIN_NAMESPACE
class QComboBox;
class QLineEdit;
QT_END_NAMESPACE

//! [0]
class Window : public QWidget
{
    Q_OBJECT

public:
    Window();

public slots:
    void echoChanged(int);
    void validatorChanged(int);
    void alignmentChanged(int);
    void inputMaskChanged(int);
    void accessChanged(int);

private:
    QLineEdit *echoLineEdit;
    QLineEdit *validatorLineEdit;
    QLineEdit *alignmentLineEdit;
    QLineEdit *inputMaskLineEdit;
    QLineEdit *accessLineEdit;
};
//! [0]
```

```
#endif
```

实现:

```
#include <QtWidgets>

#include "window.h"

//! [0]
Window::Window()
{
    QGroupBox *echoGroup = new QGroupBox(tr("Echo"));

    QLabel *echoLabel = new QLabel(tr("Mode:"));
    QComboBox *echoComboBox = new QComboBox;
    echoComboBox->addItem(tr("Normal"));
    echoComboBox->addItem(tr("Password"));
    echoComboBox->addItem(tr("PasswordEchoOnEdit"));
    echoComboBox->addItem(tr("No Echo"));

    echoLineEdit = new QLineEdit;
    //test
    /*QRegExp rx("(-|\\+)?[0-8]\\d:[0-5]\\d:[0-5]\\d");
    echoLineEdit->setValidator(new QRegExpValidator(rx, echoLineE
    echoLineEdit->setInputMask("#00:00:00;0");
    echoLineEdit->setText("+00:00:00");*/

    //echoLineEdit->setMaxLength(9);
    echoLineEdit->setPlaceholderText("电影、影人、影院、电视剧");
    echoLineEdit->setFocus();
}

//! [1]
QGroupBox *validatorGroup = new QGroupBox(tr("Validator"));

QLabel *validatorLabel = new QLabel(tr("Type:"));
QComboBox *validatorComboBox = new QComboBox;
validatorComboBox->addItem(tr("No validator"));
validatorComboBox->addItem(tr("Integer validator"));
validatorComboBox->addItem(tr("Double validator"));

validatorLineEdit = new QLineEdit;
validatorLineEdit->setPlaceholderText("Placeholder Text");

//! [2]
QGroupBox *alignmentGroup = new QGroupBox(tr("Alignment"));

QLabel *alignmentLabel = new QLabel(tr("Type:"));
QComboBox *alignmentComboBox = new QComboBox;
```

```
alignmentComboBox->addItem(tr("Left"));
alignmentComboBox->addItem(tr("Centered"));
alignmentComboBox->addItem(tr("Right"));

alignmentLineEdit = new QLineEdit;
alignmentLineEdit->setPlaceholderText("Placeholder Text");
//! [2]

//! [3]
QGroupBox *inputMaskGroup = new QGroupBox(tr("Input mask"));

QLabel *inputMaskLabel = new QLabel(tr("Type:"));
QComboBox *inputMaskComboBox = new QComboBox;
inputMaskComboBox->addItem(tr("No mask"));
inputMaskComboBox->addItem(tr("Phone number"));
inputMaskComboBox->addItem(tr("ISO date"));
inputMaskComboBox->addItem(tr("License key"));

inputMaskLineEdit = new QLineEdit;
inputMaskLineEdit->setPlaceholderText("Placeholder Text");
//! [3]

//! [4]
QGroupBox *accessGroup = new QGroupBox(tr("Access"));

QLabel *accessLabel = new QLabel(tr("Read-only:"));
QComboBox *accessComboBox = new QComboBox;
accessComboBox->addItem(tr("False"));
accessComboBox->addItem(tr("True"));

accessLineEdit = new QLineEdit;
accessLineEdit->setPlaceholderText("Placeholder Text");
//! [4]

//! [5]
connect(echoComboBox, SIGNAL(activated(int)),
        this, SLOT(echoChanged(int)));
connect(validatorComboBox, SIGNAL(activated(int)),
        this, SLOT(validatorChanged(int)));
connect(alignmentComboBox, SIGNAL(activated(int)),
        this, SLOT(alignmentChanged(int)));
connect(inputMaskComboBox, SIGNAL(activated(int)),
        this, SLOT(inputMaskChanged(int)));
connect(accessComboBox, SIGNAL(activated(int)),
        this, SLOT(accessChanged(int)));
//! [5]

//! [6]
QGridLayout *echoLayout = new QGridLayout;
echoLayout->addWidget(echoLabel, 0, 0);
echoLayout->addWidget(echoComboBox, 0, 1);
echoLayout->addWidget(echoLineEdit, 1, 0, 1, 2);
```

```
echoGroup->setLayout(echoLayout);
///  
///  
QGridLayout *validatorLayout = new QGridLayout;
validatorLayout->addWidget(validatorLabel, 0, 0);
validatorLayout->addWidget(validatorComboBox, 0, 1);
validatorLayout->addWidget(validatorLineEdit, 1, 0, 1, 2);
validatorGroup->setLayout(validatorLayout);  
  
QGridLayout *alignmentLayout = new QGridLayout;
alignmentLayout->addWidget(alignmentLabel, 0, 0);
alignmentLayout->addWidget(alignmentComboBox, 0, 1);
alignmentLayout->addWidget(alignmentLineEdit, 1, 0, 1, 2);
alignmentGroup-> setLayout(alignmentLayout);  
  
QGridLayout *inputMaskLayout = new QGridLayout;
inputMaskLayout->addWidget(inputMaskLabel, 0, 0);
inputMaskLayout->addWidget(inputMaskComboBox, 0, 1);
inputMaskLayout->addWidget(inputMaskLineEdit, 1, 0, 1, 2);
inputMaskGroup->setLayout(inputMaskLayout);  
  
QGridLayout *accessLayout = new QGridLayout;
accessLayout->addWidget(accessLabel, 0, 0);
accessLayout->addWidget(accessComboBox, 0, 1);
accessLayout->addWidget(accessLineEdit, 1, 0, 1, 2);
accessGroup->setLayout(accessLayout);
///  
///  
QGridLayout *layout = new QGridLayout;
layout->addWidget(echoGroup, 0, 0);
layout->addWidget(validatorGroup, 1, 0);
layout->addWidget(alignmentGroup, 2, 0);
layout->addWidget(inputMaskGroup, 0, 1);
layout->addWidget(accessGroup, 1, 1);
setLayout(layout);  
  
setWindowTitle(tr("Line Edits"));
}  
///  
///  
void Window::echoChanged(int index)
{
    switch (index) {
        case 0:
            //默认, 输入什么即显示什么
            echoLineEdit->setEchoMode(QLineEdit::Normal);
            break;
        case 1:
            //密码, 一般是用小黑点覆盖你所输入的字符
```



```
        echoLineEdit->setEchoMode(QLineEdit::Password);
        break;
    case 2:
        //编辑时输入字符显示输入内容, 否则用小黑点代替
        echoLineEdit->setEchoMode(QLineEdit::PasswordEchoOnEdit);
        break;
    case 3:
        //任何输入都看不见 (只是看不见, 不是不能输入)
        echoLineEdit->setEchoMode(QLineEdit::NoEcho);
    }
}
//! [9]

//! [10]
void Window::validatorChanged(int index)
{
    switch (index) {
    case 0:
        //无限制
        validatorLineEdit->setValidator(0);
        break;
    case 1:
        //只能输入整数
        validatorLineEdit->setValidator(new QIntValidator(
            validatorLineEdit));
        break;
    case 2:
        //实例, 只能输入-180到180之间的小数, 小数点后最多两位 (可用于附
        QDoubleValidator *pDfValidator = new QDoubleValidator(-18
        pDfValidator->setNotation(QDoubleValidator::StandardNotat
        validatorLineEdit->setValidator(pDfValidator);
    }

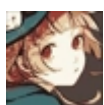
    validatorLineEdit->clear();
}
//! [10]

//! [11]
void Window::alignmentChanged(int index)
{
    switch (index) {
    case 0:
        alignmentLineEdit->setAlignment(Qt::AlignLeft);
        break;
    case 1:
        alignmentLineEdit->setAlignment(Qt::AlignCenter);
        break;
    case 2:
        alignmentLineEdit->setAlignment(Qt::AlignRight);
    }
}
//! [11]
```

```
//! [12]
void Window::inputMaskChanged(int index)
{
    switch (index) {
    case 0:
        inputMaskLineEdit->setInputMask("");
        break;
    case 1:
        inputMaskLineEdit->setInputMask("+99 99 99 99 99;_");
        break;
    case 2:
        inputMaskLineEdit->setInputMask("0000-00-00");
        inputMaskLineEdit->setText("00000000");
        inputMaskLineEdit->setCursorPosition(0);
        break;
    case 3:
        inputMaskLineEdit->setInputMask(">AAAAA-A AAAA-A AAAA-A AAAA-A");
    }
}
//! [12]

//! [13]
void Window::accessChanged(int index)
{
    switch (index) {
    case 0:
        accessLineEdit->setReadOnly(false);
        break;
    case 1:
        accessLineEdit->setReadOnly(true);
    }
}
//! [13]
```

分类: [Qt](#)



去冰三分糖
关注 - 13
粉丝 - 61

[+加关注](#)

3

0

« 上一篇: [Qt——信号槽连接: 基于字符串与基于函数的连接之间的不同](#)

» 下一篇: [Qt——浅谈样式表](#)

posted @ 2016-02-05 20:47 去冰三分糖 阅读(42979) 评论(0) 编辑 收藏

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问](#) 网站首页。

Copyright © 2019 去冰三分糖