QLineEdit

描述

是一个单行文本编辑器

允许用户输入和编辑单行纯文本

自带一组编辑功能

编辑功能

撤消

重做

剪切

粘贴

拖放

继承

QWidget

功能作用

控件的创建

QLineEdit(parent: QWidget = None)

QLineEdit(str, parent: QWidget = None)

文本的设置和获取

API

setText(str)

设置内容文本

insert(newText)

在光标处插入文本

text()

获取真实内容文本

displayText()

获取用户能看到的内容文本

应用场景

通过代码来控制输入文本内容

案例

创建一个窗口, 添加两个文本框一个按钮

要求

点击按钮后

将文本框A内输入的内容

复制到文本框B中

输出模式

API

setEchoMode(QLineEdit.EchoMode)

QLineEdit.EchoMode

NoEcho = 1

不输出

Normal = 0

正常输出

Password = 2

密文形式

PasswordEchoOnEdit = 3

编辑时明文, 结束后密文

echoMode() -> QLineEdit.EchoMode

获取输出模式

应用场景

设置输出模式, 来适应不同的应用场景

明文

密文

保护用户隐私

不输出

案例

模拟用户登录案例

创建一个窗口, 添加两个文本框一个按钮

要求

一个用作账号

另外一个用作密码

点击登录按钮后, 获取账号和密码信息

进行比对账号密码信息

正确账号

sz

正确密码

itlike

如果账号错误, 则清空账号框和密码框

密码错误则清空密码框

占位提示字符串

API

setPlaceholderText(notice\_str)

placeholderText()

应用场景

在用户输入文本内容之前, 给用户的提示语句

文本框内容

空

显示提示文本

不空

隐藏提示文本

案例

完善上述案例

添加账号和密码输入提示

清空按钮显示

API

setClearButtonEnabled(bool)

isClearButtonEnabled() -> bool

应用场景

用作快速清空文本框内容

案例

完善上述案例

为密码文本框添加清空按钮

添加操作行为

API

addAction(QAction, QLineEdit.ActionPosition)

QLineEdit.ActionPosition

QLineEdit.LeadingPosition

搁前面

QLineEdit.TrailingPosition

搁后面

addAction(QIcon, QLineEdit.ActionPosition) -> QAction

应用场景

为文本框添加附加的行为操作

案例

完善上述案例

为密码文本框添加明文和密文切换按钮

图片素材

自动补全

API

setCompleter(QCompleter)

设置完成器

completer() -> QCompleter

应用场景

根据用户已输入的字符串, 快速联想补全

案例

完善上述案例

为账号文本框, 设置补全候选项

Sz

shehui

wangzha

输入限制

API

内容长度限制

setMaxLength(int)

设置限制输入的长度

maxLength()

获取输入长度

只读限制

setReadOnly(bool)

isReadOnly()

规则验证

setValidator(QValidator)

设置验证器

setInputMask(mask\_str)

掩码验证

判定输入文本是否通过验证

hasAcceptableInput()

应用场景

限制用户在文本框中输入的内容

案例

测试以上API

补充

QValidator

描述

验证器

用于验证用户输入数据的合法性

如果一个输入框设置了验证器

到时用户在文本框中输入内容时

首先会将内容传递给验证器进行验证

validate(self, input\_text, pos)

return (QValidator.Acceptable, input\_text, pos)

验证通过

return (QValidator.Intermediate, input\_text, pos)

暂不作判定是否通过验证

return (QValidator.Invalid, input\_text, pos)

验证不通过

如果输入框结束输入后, 上述的验证状态并非有效, 则会调用修复方法

fixup(self, input\_text)

return 修正后文本

是一个抽象类, 使用前需要进行子类化操作

自定义子类

系统提供子类

QIntValidator(bottom, top, parent)

限制整型数据范围

QDoubleValidator

浮点类型数据限制范围

经测试, 无效

需要手动实现

QRegExpValidator

通过正则表达式限定

基本使用

第一步

子类化此类

第二步

实现

validate(self, input\_text, pos)

return (QValidator.Acceptable, input\_text, pos)

验证通过

return (QValidator.Intermediate, input\_text, pos)

暂不作判定是否通过验证

return (QValidator.Invalid, input\_text, pos)

验证不通过

fixup(self, input\_text)

return 修正后文本

掩码含义

掩码可以指定固定位置的固定数据类型, 达到一个格式上的限制

例如

座机号码

四位区号-七位电话

IP地址

XXX.XXX.XXX.XXX

掩码由一串掩码字符和分隔符组成

+

可选的分号; 和 空白占位字符

掩码字符含义

是否被编辑

API

isModified()

setModified(bool)

应用场景

标识文本内容是否被修改

案例

测试以上API

光标控制

API

cursorBackward(bool mark，int steps = 1)

向后(左)移动steps个字符

mark: True

带选中效果

mark: False

不带选中效果

cursorForward(bool mark，int steps = 1)

向前(右)移动steps个字符

mark: True

带选中效果

mark: False

不带选中效果

cursorWordBackward(bool mark)

向后(左)移动一个单词长度

mark: True

带选中效果

mark: False

不带选中效果

cursorWordForward(bool mark)

向前(右)移动一个单词长度

mark: True

带选中效果

mark: False

不带选中效果

home(bool)

移动到行首

True

带选中

False

不带选中

end(bool)

移动到行尾

True

带选中

False

不带选中

setCursorPosition(int)

设置光标位置

cursorPosition()

获取光标位置

cursorPositionAt(const QPoint＆ pos)

获取指定坐标位置对应文本光标位置

应用场景

控制光标, 以及文本选中操作

案例

测试以上API

文本边距设置

API

getTextMargins()

setTextMargins(int left，int top，int right，int bottom)

应用场景

设置文本内容边距

案例

测试以上API

对齐方式

API

setAlignment(Qt.Alignment)

设置输入文本的对齐方式

Qt.Alignment

水平

Qt.AlignLeft

Qt.AlignRight

Qt.AlignHCenter

Qt.AlignJustify

此处同左对齐

垂直

Qt.AlignTop

Qt.AlignBottom

Qt.AlignVCenter

Qt.AlignBaseline

Qt.AlignCenter

等同于

Qt.AlignHCenter | Qt.AlignVCenter

垂直和水平都居中

alignment() -> Qt.Alignment

应用场景

设置用户输入的内容文本对齐方式

案例

测试上述API

常用编辑功能

API

退格

backspace()

删除选中文本

如果有

或

删除光标左侧一个字符

删除

del\_()

删除选中文本

如果有

或

删除光标右侧的一个字符

清空

clear()

删除所有文本框内容

复制

copy()

剪切

cut()

粘贴

paste()

撤消

isUndoAvailable()

undo()

重做

isRedoAvailable()

redo()

拖放

setDragEnabled(bool)

设置选中文本后是否可以拖拽

文本选中

setSelection(start\_pos, length)

选中指定区间的文本

selectAll()

选中所有文本

deselect()

取消选中已选择文本

hasSelectedText()

是否有选中文本

selectedText() -> str

获取选中的文本

selectionStart() -> int

选中的开始位置

selectionEnd() -> int

选中的结束位置

selectionLength() -> int

选中的长度

应用场景

编辑用户输入的文本

案例

测试上述API

信号

textEdited( text)

文本编辑时发射的信号

textChanged(text)

文本框文本发生改变时发出的信号

returnPressed()

按下回车键时发出的信号

editingFinished()

结束编辑时发出的信号

cursorPositionChanged(int oldPos，int newPos)

光标位置发生改变时发出的信号

selectionChanged()

选中的文本发生改变时发出的信号

常用