

QInputDialog Class

QInputDialog 类提供了一个简单方便的对话框来从用户处获取单个值。 [更多的...](#)

Header:	#include
CMake:	find_package(Qt6 REQUIRED COMPONENTS Widgets) target_link_libraries(mytarget PRIVATE Qt6::Widgets)
qmake:	QT += widgets
Inherits:	QDialog

- [所有成员的列表](#)，包括继承的成员
- QDialog 是标准对话框的一部分。

公共类型

enum	InputDialogOption { NoButtons, UseListViewForComboBoxItems, UsePlainTextEditForTextInput }
flags	InputDialogOptions
enum	InputMode { TextInput, IntInput, DoubleInput }

特性

cancelButtonText : QString comboBoxEditable :	intMaximum : int intMinimum : int intStep :
boolcomboBoxItems : QStringList doubleDecimals :	intintValue : int labelText : QString okButtonText :
intdoubleMaximum : double doubleMinimum :	QString options :
doubledoubleStep : double doubleValue : int inputMode :	InputDialogOption textEchoMode :
InputMode	QLineEdit::EchoMode textValue : QString

公共方法

	QInputDialog (QWidget *parent = nullptr, Qt::WindowFlags flags = Qt::WindowFlags())
virtual	~QInputDialog()

QString	cancelButtonText() const
QInputDialog::QInputDialog(QWidget *parent, Qt::WindowFlags flags = Qt::WindowFlags())	
QStringList	comboBoxItems() const
int	doubleDecimals() const
double	doubleMaximum() const
double	doubleMinimum() const
double	doubleStep() const
double	doubleValue() const
QInputDialog::InputMode	inputMode() const
int	intMaximum() const
int	intMinimum() const
int	intStep() const
int	intValue() const
bool	isComboBoxEditable() const
QString	labelText() const
QString	okButtonText() const
void	open (QObject receiver*, const char member*)
QInputDialog::InputDialogOptions	options() const
void	setCancelButtonText (const QString &text)
void	setComboBoxEditable (bool editable)
void	setComboBoxItems (const QStringList &items)
void	setDoubleDecimals (int decimals)
void	setDoubleMaximum (double max)
void	setDoubleMinimum (double min)
void	setDoubleRange (double min, double max)
void	setDoubleStep (double step)
void	setDoubleValue (double value)
void	setInputMode (QInputDialog::InputMode mode)
void	setIntMaximum (int max)
void	setIntMinimum (int min)

void	setMaximum (int <i>max</i>)
void	QInputDialog (QWidget <i>*parent</i> = nullptr, Qt::WindowFlags <i>flags</i> = Qt::WindowFlags(), int <i>max</i>)
void	setIntStep (int <i>step</i>)
void	setIntValue (int <i>value</i>)
void	setLabelText (const QString & <i>text</i>)
void	setOkButtonText (const QString & <i>text</i>)
void	setOption (QInputDialog::InputDialogOption <i>option</i> , bool <i>on</i> = true)
void	setOptions (QInputDialog::InputDialogOptions <i>options</i>)
void	setTextEchoMode (QLineEdit::EchoMode <i>mode</i>)
void	setTextValue (const QString & <i>text</i>)
bool	testOption (QInputDialog::InputDialogOption <i>option</i>) const
QLineEdit::EchoMode	textEchoMode () const
QString	textValue () const

重载的公共方法

virtual void	done (int <i>result</i>) override
virtual QSize	minimumSizeHint () const override
virtual void	setVisible (bool <i>visible</i>) override
virtual QSize	sizeHint () const override

信号

void	doubleValueChanged (double <i>value</i>)
void	doubleValueSelected (double <i>value</i>)
void	intValueChanged (int <i>value</i>)
void	intValueSelected (int <i>value</i>)
void	textValueChanged (const QString & <i>text</i>)
void	textValueSelected (const QString & <i>text</i>)

静态公共成员

double	getDouble (QWidget parent, <i>const QString &title</i> , <i>const QString &label*</i> , double value = 0, double min = -2147483647, double max = 2147483647, int decimals = 1, bool ok* = nullptr, Qt::WindowFlags flags = Qt::WindowFlags(), double step = 1)
int	getInt (QWidget parent, <i>const QString &title</i> , <i>const QString &label*</i> , int value = 0, int min = -2147483647, int max = 2147483647, int step = 1, bool ok* = nullptr, Qt::WindowFlags flags = Qt::WindowFlags())
QString	getItem (QWidget parent, <i>const QString &title</i> , <i>const QString &label</i> , <i>const QStringList &items*</i> , int current = 0, bool editable = true, bool ok* = nullptr, Qt::WindowFlags flags = Qt::WindowFlags(), Qt::InputMethodHints inputMethodHints = Qt::ImhNone)
QString	getMultiLineText (QWidget parent, <i>const QString &title</i> , <i>const QString &label</i> , <i>const QString &text*</i> = QString(), bool ok* = nullptr, Qt::WindowFlags flags = Qt::WindowFlags(), Qt::InputMethodHints inputMethodHints = Qt::ImhNone)
QString	getText (QWidget parent, <i>const QString &title</i> , <i>const QString &label*</i> , QLineEdit::EchoMode mode = QLineEdit::Normal, <i>const QString &text</i> = QString(), bool ok* = nullptr, Qt::WindowFlags flags = Qt::WindowFlags(), Qt::InputMethodHints inputMethodHints = Qt::ImhNone)

详细说明

输入值可以是字符串、数字或列表中的项目。必须设置标签来告诉用户应该输入什么。

提供了五个静态便利函数：[getText\(\)](#)、[getMultiLineText\(\)](#)、[getInt\(\)](#)、[getDouble\(\)](#) 和 [getItem\(\)](#)。所有函数都可以以类似的方式使用，例如：

```
bool ok;
QString text = QInputDialog::getText(this, tr("QInputDialog::getText()"),
                                     tr("User name:"), QLineEdit::Normal,
                                     QDir::home().dirName(), &ok);

if (ok && !text.isEmpty())
    textLabel->setText(text);
```

如果用户单击，则该ok变量设置为 trueOK; 否则，它被设置为 false。

这[Standard Dialogs](#)示例展示了如何使用 QInputDialog 以及其他内置 Qt 对话框。

也可以看看[QMessageBox](#)和[Standard Dialogs Example](#)。

会员类型文档

枚举 `QInputDialog::InputDialogOption` 标志 `QInputDialog::InputDialogOptions`

该枚举指定影响输入对话框外观的各种选项。

持续的	价值	描述
<code>QInputDialog::NoButtons</code>	<code>0x00000001</code>	不显示 OK 和 Cancel 按钮（对于“实时对话框”有用）。
<code>QInputDialog::UseListViewForComboBoxItems</code>	<code>0x00000002</code>	用一个 QListView 而不是不可编辑的 QComboBox 用于显示设置的项目 setComboBoxItems() 。
<code>QInputDialog::UsePlainTextEditForTextInput</code>	<code>0x00000004</code>	用一个 QPlainTextEdit 用于多行文本输入。该值是在 5.2 中引入的。

`InputDialogOptions` 类型是**[QFlags](#)** 的 typedef 。它存储 `InputDialogOption` 值的 OR 组合。

也可以看看**[options](#)**,**[setOption \(\)](#)** , 和**[testOption\(\)](#)**。

enum QInputDialog::InputMode

该枚举描述了可以为对话框选择的不同输入模式。

持续的	价值	描述
<code>QInputDialog::TextInput</code>	<code>0</code>	用于输入文本字符串。
<code>QInputDialog::IntInput</code>	<code>1</code>	用于输入整数。
<code>QInputDialog::DoubleInput</code>	<code>2</code>	用于输入双精度浮点数。

也可以看看**[inputMode](#)**。

财产文件

cancelButtonText : [QString](#)

该属性保存用于取消对话框的按钮的文本

访问功能:

<code>QString</code>	<code>cancelButtonText() const</code>
<code>void</code>	<code>setCancelButtonText(const QString &text)</code>

comboBoxEditable : bool

该属性保存输入对话框中使用的组合框是否可编辑

访问功能:

bool	isComboBoxEditable() const
void	setComboBoxEditable (bool <i>editable</i>)

comboBoxItems : QStringList

该属性保存输入对话框的组合框中使用的项目

访问功能:

QStringList	comboBoxItems() const
void	setComboBoxItems (const QStringList & <i>items</i>)

doubleDecimals : int

设置双旋转框的小数精度

访问功能:

int	doubleDecimals() const
void	setDoubleDecimals (int <i>decimals</i>)

也可以看看[QDoubleSpinBox::setDecimals\(\)](#)。

doubleMaximum : double

该属性保存接受为输入的最大双精度浮点值

该属性仅在使用输入对话框时相关[DoubleInput](#)模式。

访问功能:

double	doubleMaximum() const
void	setDoubleMaximum (double <i>max</i>)

doubleMinimum : double

该属性保存接受作为输入的最小双精度浮点值

该属性仅在使用输入对话框时相关[DoubleInput](#)模式。

访问功能：

double	doubleMinimum() const
void	setDoubleMinimum (double <i>min</i>)

doubleStep : double

该属性保存双精度值增加和减少的步长

该属性仅在使用输入对话框时相关[DoubleInput](#)模式。

访问功能：

double	doubleStep() const
void	setDoubleStep (double <i>step</i>)



doubleValue : int

该属性保存当前接受为输入的双精度浮点值

该属性仅在使用输入对话框时相关[DoubleInput](#)模式。

访问功能：

double	doubleValue() const
void	setDoubleValue (double <i>value</i>)

通知器信号：

void	doubleValueChanged (double <i>value</i>)
-------------	--

inputMode : *InputMode*

该属性保存用于输入的模式

此属性有助于确定使用哪个小部件将输入输入到对话框中。

访问功能:

QInputDialog::InputMode	inputMode() const
void	setInputMode (QInputDialog::InputMode <i>mode</i>)

intMaximum : *int*

该属性保存接受为输入的最大整数值

该属性仅在使用输入对话框时相关[IntInput](#)模式。

访问功能:

int	intMaximum() const
void	setIntMaximum (int <i>max</i>)



intMinimum : *int*

该属性保存接受为输入的最小整数值

该属性仅在使用输入对话框时相关[IntInput](#)模式。

访问功能:

int	intMinimum() const
void	setIntMinimum (int <i>min</i>)


intStep : *int*

该属性保存整数值增加和减少的步长

该属性仅在使用输入对话框时相关[IntInput](#)模式。

访问功能:

int	intStep() const
void	setIntStep (int <i>step</i>)



intValue : int

该属性保存当前接受为输入的整数值

该属性仅在使用输入对话框时相关[IntInput](#)模式。

访问功能：

int	intValue() const
void	setIntValue(int value)

通知器信号：

void	intValueChanged (int value)
------	---

labelText : QString

该属性保存标签的文本，描述需要输入的内容

访问功能：

QString	labelText() const
void	setLabelText(const QString &text)

okButtonText : QString

该属性保存用于接受对话框中的条目的按钮的文本

访问功能：

QString	okButtonText() const
void	setOkButtonText(const QString &text)



options : InputDialogOptions

该属性包含影响对话框外观的各种选项

默认情况下，所有选项均被禁用。

访问功能：

QInputDialog::InputDialogOptions	options() const
void	setOptions (QInputDialog::InputDialogOptions <i>options</i>)

也可以看看[setOption](#) () 和[testOption](#)()。

textEchoMode : QLineEdit::EchoMode

该属性保存文本值的回显模式

该属性仅在使用输入对话框时相关[TextInput](#)模式。

访问功能：

QLineEdit::EchoMode	textEchoMode() const
void	setTextEchoMode (QLineEdit::EchoMode <i>mode</i>)

textValue : QString

该属性保存输入对话框的文本值

该属性仅在使用输入对话框时相关[TextInput](#)模式。

访问功能：

QString	textValue() const
void	setTextValue (const QString & <i>text</i>)

通知器信号：

void	textValueChanged (const QString & <i>text</i>)
------	--

成员函数文档

QInputDialog::QInputDialog([QWidget](#) **parent* = nullptr, [Qt::WindowFlags](#) *flags* = [Qt::WindowFlags\(\)](#))

使用给定的构造一个新的输入对话框*parent*和窗户*flags*。

[virtual]QInputDialog::~QInputDialog()

销毁输入对话框。

[override virtual]void QInputDialog::done(int result)

重新实现： `QDialog::done` (int r) 。

关闭对话框并将其结果代码设置为`result`。如果此对话框显示为`exec()`、`done()` 导致本地事件循环完成，并且`exec()` 回来`result`。

也可以看看`QDialog::done()`。

[signal]void QInputDialog::doubleValueChanged(double value)

每当对话框中的双精度值发生变化时，就会发出此信号。当前值由下式指定`value`。

该信号仅当输入对话框用于`DoubleInput`模式。

注意： 属性的通知程序信号`doubleValue`。

[signal]void QInputDialog::doubleValueSelected(double value)

每当用户通过接受对话框选择双精度值时，就会发出此信号；例如，通过单击OK按钮。所选值由下式指定`value`。

该信号仅当输入对话框用于`DoubleInput`模式。

[static]double QInputDialog::getDouble(QWidget parent, const QString &title, const QString &label, double value = 0, double min = -2147483647, double max = 2147483647, int decimals = 1, bool ok* = nullptr, Qt::WindowFlags flags = Qt::WindowFlags(), double step = 1)*

用于从用户处获取浮点数的静态便利函数。

`title`是显示在对话框标题栏中的文本。`label`是向用户显示的文本（应该说明应输入的内容）。`value`是行编辑将设置为的默认浮点数。`min`和`max`是用户可以选择的最小值和最大值。`decimals`是该数字可以具有的最大小数位数。`step`是当用户按箭头按钮增加或减少值时值更改的量。

如果`ok`为非空，`ok*`如果用户按下，将被设置为 `true`OK如果用户按下，则返回 `false`Cancel*。对话框的父级是 `parent`。该对话框将是模态的并使用小部件`flags*`。

该函数返回用户输入的浮点数。

像这样使用这个静态函数：

```
bool ok;
double d = QInputDialog::getDouble(this, tr("QInputDialog::getDouble()"),
                                   tr("Amount:"), 37.56, -10000, 10000, 2, &ok,
                                   Qt::WindowFlags(), 1);

if (ok)
    doubleLabel->setText(QString("%1").arg(d));
```

也可以看看[getText\(\)](#),[getInt\(\)](#),[getItem \(\)](#) , 和[getMultiLineText\(\)](#)。

[static]int QInputDialog::getInt([QWidget](#) parent, const [QString](#) &title, const [QString](#) &label, int value = 0, int min = -2147483647, int max = 2147483647, int step = 1, bool ok* = nullptr, [Qt::WindowFlags](#) flags = [Qt::WindowFlags\(\)](#))*

用于获取用户输入的整数的静态便捷函数。

*title*是显示在对话框标题栏中的文本。*label*是向用户显示的文本（应该说明应输入的内容）。*value*是旋转框将设置为的默认整数。*min*和*max*是用户可以选择的最小值和最大值。*step*是当用户按箭头按钮增加或减少值时值更改的量。

如果*ok*为非空 *ok****如果用户按下，将被设置为 true**OK**如果用户按下，则返回 false**Cancel*。对话框的父级是parent。该对话框将是模态的并使用小部件*flags**。

成功时，该函数返回用户输入的整数；失败时，它返回初始值*value*。

像这样使用这个静态函数：

```
bool ok;
int i = QInputDialog::getInt(this, tr("QInputDialog::getInt()"),
                             tr("Percentage:"), 25, 0, 100, 1, &ok);

if (ok)
    integerLabel->setText(tr("%1%").arg(i));
```

也可以看看[getText\(\)](#),[getDouble\(\)](#),[getItem \(\)](#) , 和[getMultiLineText\(\)](#)。

```
[static]QString QInputDialog::getItem(QWidget parent, const QString
&title, const QString &label, const QStringList &items*, int current = 0,
bool editable = true, bool ok* = nullptr, Qt::WindowFlags flags =
Qt::WindowFlags(), Qt::InputMethodHints inputMethodHints =
Qt::ImhNone)
```

静态便利函数让用户从字符串列表中选择个项目。

*title*是显示在对话框标题栏中的文本。*label*是向用户显示的文本（应该说明应输入的内容）。*items*是插入到组合框中的字符串列表。*current*是当前项目的项目编号。*inputMethodHints*是组合框可编辑且输入法处于活动状态时将使用的输入法提示。

如果`editable`为 `true` 用户可以输入自己的文本；否则，用户只能选择现有项目之一。

如果`ok`如果用户按下，则为非空`ok`将设置为 `true`*OK如果用户按下，则返回 `false`Cancel*。对话框的父级是`parent`。该对话框将是模态的并使用小部件`flags`。

该函数返回当前项目的文本，或者如果`editable`为 `true` 时，组合框的当前文本。

像这样使用这个静态函数：

```
QStringList items;
items << tr("Spring") << tr("Summer") << tr("Fall") << tr("Winter");

bool ok;
QString item = QInputDialog::getItem(this, tr("QInputDialog::getItem()"),
                                     tr("Season:"), items, 0, false, &ok);

if (ok && !item.isEmpty())
    itemLabel->setText(item);
```

也可以看看`getText()`,`getInt()`,`getDouble ()` , 和`getMultiLineText()`。

```
[static]QString QInputDialog::getMultiLineText(QWidget parent, const
QString &title, const QString &label, const QString &text* = QString(),
bool ok* = nullptr, Qt::WindowFlags flags = Qt::WindowFlags(),
Qt::InputMethodHints inputMethodHints = Qt::ImhNone)
```

用于从用户获取多行字符串的静态便捷函数。

*title*是显示在对话框标题栏中的文本。*label*是向用户显示的文本（应该说明应输入的内容）。*text*是放置在纯文本编辑中的默认文本。*inputMethodHints*是输入法提示，如果输入法处于活动状态，将在编辑小部件中使用。

如果`ok`如果用户按下，则为非空`ok`将设置为 `true`*OK如果用户按下，则返回 `false`Cancel*。对话框的父级是`parent`。该对话框将是模态的并使用指定的小部件`flags`。

如果对话框被接受，则此函数返回对话框的纯文本编辑中的文本。如果对话框被拒绝，则返回 `null``QString`被返回。

像这样使用这个静态函数：

```

bool ok;
QString text = QInputDialog::getMultiLineText(this, tr("QInputDialog::getMultiLineText()"),
                                              tr("Address:"), "John Doe\nFreedom Street",
                                              &ok);
if (ok && !text.isEmpty())
    multiLineTextLabel->setText(text);

```

也可以看看[getInt\(\)](#),[getDouble\(\)](#),[getItem \(\)](#) , 和[getText\(\)](#)。

[static]QString QInputDialog::getText(QWidget parent, const QString &title, const QString &label, QLineEdit::EchoMode mode = QLineEdit::Normal, const QString &text = QString(), bool ok* = nullptr, Qt::WindowFlags flags = Qt::WindowFlags(), Qt::InputMethodHints inputMethodHints = Qt::ImhNone)*

从用户处获取字符串的静态便捷函数。

*title*是显示在对话框标题栏中的文本。*label*是向用户显示的文本（应该说明应输入的内容）。*text*是放置在行编辑中的默认文本。*mode*是行编辑将使用的回显模式。*inputMethodHints*是输入法提示，如果输入法处于活动状态，将在编辑小部件中使用。

如果`ok`如果用户按下，则为非空`ok`将设置为 `true`*OK如果用户按下，则返回 `false`Cancel*。对话框的父级是`parent`。该对话框将是模态的并使用指定的小部件`flags`。

如果接受对话框，则此函数返回对话框行编辑中的文本。如果对话框被拒绝，则返回 `null`[QString](#)被返回。

像这样使用这个静态函数：

```

bool ok;
QString text = QInputDialog::getText(this, tr("QInputDialog::getText()"),
                                     tr("User name:"), QLineEdit::Normal,
                                     QDir::home().dirName(), &ok);

if (ok && !text.isEmpty())
    textLabel->setText(text);

```

也可以看看[getInt\(\)](#),[getDouble\(\)](#),[getItem \(\)](#) , 和[getMultiLineText\(\)](#)。

[signal]void QInputDialog::intValueChanged(int value)

每当对话框中的整数值发生变化时，就会发出此信号。当前值由下式指定`value`。

该信号仅当输入对话框用于[IntInput](#)模式。

注意：属性的通知程序信号[intValue](#)。

[signal] void QInputDialog::intValueSelected(int value)

每当用户通过接受对话框选择整数值时，就会发出此信号；例如，通过单击OK按钮。所选值由下式指定`value`。

该信号仅当输入对话框用于IntInput模式。

[override virtual] QSize QInputDialog::minimumSizeHint() const

重新实现：QDialog::minimumSizeHint() const。

void QInputDialog::open(QObject receiver, const char member*)*

该函数将其信号之一连接到由`receiver`和`member`。具体信号取决于中指定的参数`member`。这些都是：

- `textValueSelected` () 如果`member`有一个QString对于它的第一个参数。
- `intValueSelected` () 如果`member`它的第一个参数是 `int` 。
- `doubleValueSelected` () 如果`member`它的第一个参数有一个 `double` 。
- `accepted` () 如果`member`没有参数。

当对话框关闭时，信号将从插槽断开。

void QInputDialog::setDoubleRange(double min, double max)

设置对话框在使用时接受的双精度浮点值的范围DoubleInput模式，最小值和最大值由`min`和`max`分别。

void QInputDialog::setIntRange(int min, int max)

设置对话框使用时接受的整数值范围IntInput模式，最小值和最大值由`min`和`max`分别。

*void QInputDialog::setOption(QInputDialog::InputDialogOption option,
bool on = true)*

设置给定的`option`启用如果`on`是真的; 否则，清除给定的`option`。

也可以看看`options`和`testOption()`。

[override virtual]void QInputDialog::setVisible(bool visible)

重新实现： [QDialog::setVisible](#) (布尔值可见) 。

[override virtual][QSize](#) QInputDialog::sizeHint() const

重新实现： [QDialog::sizeHint\(\)](#) const。

*bool QInputDialog::testOption([QInputDialog::InputDialogOption](#) option)
const*

true如果给定则返回`option`已启用；否则，返回 false。

也可以看看[options](#)和[setOption\(\)](#)。

[signal]void QInputDialog::textValueChanged(const [QString](#) &text)

每当对话框中的文本字符串发生变化时，就会发出此信号。当前字符串由以下方式指定`text`。

该信号仅当输入对话框用于[TextInput](#)模式。

注意：属性的通知程序信号`textValue`。

[signal]void QInputDialog::textValueSelected(const [QString](#) &text)

每当用户通过接受对话框选择文本字符串时，就会发出此信号；例如，通过单击**OK**按钮。所选字符串由以下方式指定`text`。

该信号仅当输入对话框用于[TextInput](#)模式。