QLabel Class

QLabel 小部件提供文本或图像显示。更多的...

Header:	**#include < QLabel>**
CMake:	find_package(Qt6 REQUIRED COMPONENTS Widgets) target_link_libraries(mytarget PRIVATE Qt6::Widgets)
qmake:	QT += widgets
Inherits:	QFrame

- 所有成员的列表，包括继承的成员
- 已弃用的成员

特性

alignment : Qt::Alignment
hasSelectedText : const bool
indent : int
margin : int
openExternalLinks : bool
pixmap : QPixmap
scaledContents : bool
selectedText : const QString
text : QString
textFormat : Qt::TextFormat
textInteractionFlags : Qt::TextInteractionFlags
wordWrap : bool

公共函数

	QLabel(QWidget *parent = nullptr, Qt::WindowFlags f = Qt::WindowFlags())
	QLabel(const QString &text, QWidget *parent = nullptr, Qt::WindowFlags f = Qt::WindowFlags())
virtual	**~QLabel**()
Qt::Alignment	**alignment**() const
QWidget *	**buddy**() const
bool	**hasScaledContents**() const
bool	**hasSelectedText**() const
int	**indent**() const
int	**margin**() const
QMovie *	**movie**() const
bool	**openExternalLinks**() const
QPicture	**picture**() const

	QLabel(QWidget **parent* = nullptr, Qt::WindowFlags *f* = Qt::WindowFlags())
QPixmap	**pixmap**() const
QTextDocument::ResourceProvider	**resourceProvider**() const
QString	**selectedText**() const
int	**selectionStart**() const
void	**setAlignment**(Qt::Alignment)
void	**setBuddy**(QWidget **buddy*)
void	**setIndent**(int)
void	**setMargin**(int)
void	**setOpenExternalLinks**(bool *open*)
void	**setResourceProvider**(const QTextDocument::ResourceProvider &*provider*)
void	**setScaledContents**(bool)
void	**setSelection**(int *start*, int *length*)
void	**setTextFormat**(Qt::TextFormat)
void	**setTextInteractionFlags**(Qt::TextInteractionFlags *flags*)
void	**setWordWrap**(bool *on*)
QString	**text**() const
Qt::TextFormat	**textFormat**() const
Qt::TextInteractionFlags	**textInteractionFlags**() const
bool	**wordWrap**() const

重载的公共函数

virtual int	**heightForWidth(int *w*) const override**
virtual QSize	**minimumSizeHint**() const override
virtual QSize	**sizeHint**() const override

公共槽

void	**clear()**
void	**setMovie**(QMovie **movie*)

void	**setNum**(int *num*)
void	**setNum**(double *num*)
void	**setPicture**(const QPicture &*picture*)
void	**setPixmap**(const QPixmap &)
void	**setText**(const QString &)

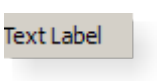
信号

void	**linkActivated**(**const QString &*link***)
void	**linkHovered**(const QString &*link*)

重载的protected function

virtual void	**changeEvent**(**QEvent *ev*) override**
virtual void	**contextMenuEvent**(QContextMenuEvent *ev*) override
virtual bool	**event**(QEvent *e*) override
virtual void	**focusInEvent**(QFocusEvent *ev*) override
virtual bool	**focusNextPrevChild**(bool *next*) override
virtual void	**focusOutEvent**(QFocusEvent *ev*) override
virtual void	**keyPressEvent**(QKeyEvent *ev*) override
virtual void	**mouseMoveEvent**(QMouseEvent *ev*) override
virtual void	**mousePressEvent**(QMouseEvent *ev*) override
virtual void	**mouseReleaseEvent**(QMouseEvent *ev*) override
virtual void	**paintEvent**(QPaintEvent *) override

详细说明



QLabel 用于显示文本或图像。不提供用户交互功能。标签的视觉外观可以通过多种方式进行配置，并且可以用于为另一个小部件指定焦点助记键。

QLabel 可以包含以下任意内容类型：

内容	环境
纯文本	通过一个QString到setText()。
富文本	通过一个QString包含富文本到setText()。
像素图	通过一个QPixmap到setPixmap()。
电影	通过一个QMovie到setMovie()。
一个号码	将*int*或*double*传递给setNum()，将数字转换为纯文本。
没有什么	与空的纯文本相同。这是默认设置。通过设置clear()。

警告：当经过QString到构造函数或调用setText()，请确保清理您的输入，因为 QLabel 会尝试猜测它将文本显示为纯文本还是富文本（HTML 4 标记的子集）。您可能想致电setTextFormat() 明确，例如，如果您希望文本采用纯格式但无法控制文本源（例如，当显示从 Web 加载的数据时）。

当使用这些功能中的任何一个更改内容时，所有以前的内容都会被清除。

默认情况下，显示标签left-aligned, vertically-centered文本和图像，其中要显示的文本中的任何选项卡automatically expanded。然而，QLabel 的外观可以通过多种方式进行调整和微调。

QLabel 小部件区域内内容的定位可以通过以下方式调整setAlignment （） 和setIndent()。文本内容还可以沿单词边界换行setWordWrap()。例如，此代码设置了一个下沉式面板，其右下角有两行文本（两行与标签的右侧齐平）：

```
QLabel *label = new QLabel(this);
label->setFrameStyle(QFrame::Panel | QFrame::Sunken);
label->setText("first line\nsecond line");
label->setAlignment(Qt::AlignBottom | Qt::AlignRight);
```

QLabel 继承的属性和函数QFrame也可用于指定用于任何给定标签的小部件框架。

QLabel 通常用作交互式小部件的标签。为此，QLabel 提供了一种有用的机制来添加助记符（请参阅QKeySequence），这会将键盘焦点设置到另一个小部件（称为 QLabel 的“伙伴”）。例如：

```
QLineEdit *phoneEdit = new QLineEdit(this);
QLabel *phoneLabel = new QLabel("&Phone:", this);
phoneLabel->setBuddy(phoneEdit);
```

在此示例中，键盘焦点转移到标签的好友（QLineEdit）当用户按 Alt+P 时。如果好友是一个按钮（继承自QAbstractButton），触发助记符将模拟按钮单击。

也可以看看QLineEdit,QTextEdit,QPixmap， 和QMovie。

属性文档

alignment : Qt::Alignment

该属性保存标签内容的对齐方式

默认情况下，标签的内容左对齐且垂直居中。

访问功能：

Qt::Alignment	alignment() const
void	**setAlignment**(Qt::Alignment)

也可以看看text。

[read-only]hasSelectedText : const bool

该属性保存是否有任何文本被选中

`true`如果用户选择了部分或全部文本，则hasSelectedText() 返回；否则返回`false`.

默认情况下，该属性为`false`。

注：textInteractionFlags标签上的设置需要包含 TextSelectableByMouse 或 TextSelectableByKeyboard。

访问功能：

bool	hasSelectedText() const

也可以看看selectedText()。

indent : int

该属性保存标签的文本缩进（以像素为单位）

如果标签显示文本，则缩进适用于左边缘，如果alignment（） 是Qt::AlignLeft，到右边缘，如果alignment（） 是Qt::AlignRight，到顶部边缘，如果alignment（） 是Qt::AlignTop，并且到底部边缘，如果alignment（） 是Qt::AlignBottom。

如果缩进为负，或者没有设置缩进，则标签将按如下方式计算有效缩进：如果frameWidth()为0时，有效缩进变为0。如果frameWidth() 大于 0，有效缩进变为小部件当前“x”字符宽度的一半font()。

默认情况下，缩进为-1，表示有效缩进按照上述方式计算。

访问功能：

int	indent() const
void	**setIndent**(int)

也可以看看alignment,margin,frameWidth（），和font()。

margin : int

该属性保存边距的宽度

边距是帧最里面的像素和内容最外面的像素之间的距离。

默认边距为 0。

访问功能：

int	margin() const
void	**setMargin**(int)

也可以看看indent。

openExternalLinks : bool

指定是否QLabel应该使用自动打开链接QDesktopServices::openUrl() 而不是发出linkActivated（） 信号。

注：textInteractionFlags标签上的设置需要包含 LinksAccessibleByMouse 或 LinksAccessibleByKeyboard。

默认值为 false。

访问功能：

bool	openExternalLinks() const
void	**setOpenExternalLinks**(bool *open*)

也可以看看textInteractionFlags()。

pixmap : QPixmap

该属性保存标签的像素图。

设置像素图会清除以前的所有内容。好友快捷方式（如果有）被禁用。

访问功能：

QPixmap	pixmap() const
void	**setPixmap**(const QPixmap &)

scaledContents : bool

该属性保存标签是否缩放其内容以填充所有可用空间。

启用后并且标签显示像素图，它将缩放像素图以填充可用空间。

该属性的默认值是 false。

访问功能：

bool	**hasScaledContents() const**
void	**setScaledContents**(bool)

[read-only]selectedText : const QString

该属性保存选定的文本

如果没有选定的文本，则该属性的值为空字符串。

默认情况下，该属性包含一个空字符串。

注：textInteractionFlags标签上的设置需要包含 TextSelectableByMouse 或 TextSelectableByKeyboard。

访问功能：

QString	**selectedText() const**

也可以看看hasSelectedText()。

text : QString

该属性保存标签的文本

如果没有设置文本，这将返回一个空字符串。设置文本会清除以前的所有内容。

文本将被解释为纯文本或富文本，具体取决于文本格式设置；看setTextFormat()。默认设置是Qt::AutoText; IEQLabel将尝试自动检测文本集的格式。看Supported HTML Subset对于富文本的定义。

如果已经设置了好友，则好友助记词会根据新文本进行更新。

注意QLabel非常适合显示小型富文本文档，例如从标签的调色板和字体属性获取文档特定设置（字体、文本颜色、链接颜色）的小型文档。对于大型文档，请使用QTextEdit改为只读模式。QTextEdit必要时还可以提供滚动条。

注意：此功能启用鼠标跟踪，如果*text*包含丰富的文本。

访问功能：

QString	**text() const**

void	**setText**(const QString &)
QString	**text() const**

也可以看看setTextFormat(),setBuddy（），和alignment。

textFormat : Qt::TextFormat

该属性保存标签的文本格式

请参阅Qt::TextFormat枚举以解释可能的选项。

默认格式是Qt::AutoText。

访问功能：

Qt::TextFormat	**textFormat() const**
void	**setTextFormat**(Qt::TextFormat)

也可以看看text()。

textInteractionFlags : Qt::TextInteractionFlags

指定标签在显示文本时应如何与用户输入交互。

如果标志包含Qt::LinksAccessibleByKeyboard焦点策略也自动设置为Qt::StrongFocus。如果Qt::TextSelectableByKeyboard设置后，焦点策略设置为Qt::ClickFocus。

默认值为Qt::LinksAccessibleByMouse。

访问功能：

Qt::TextInteractionFlags	**textInteractionFlags() const**
void	**setTextInteractionFlags**(Qt::TextInteractionFlags *flags*)

wordWrap : bool

该属性保存标签的自动换行策略

如果此属性为`true`，则标签文本将在必要的断字处换行；否则它根本没有被包裹。

默认情况下，自动换行处于禁用状态。

访问功能：

bool	**wordWrap() const**
void	**setWordWrap**(bool *on*)

也可以看看text。

成员函数文档

*[explicit]QLabel::QLabel(QWidget *parent = nullptr, Qt::WindowFlags f = Qt::WindowFlags())*

构造一个空标签。

这*parent*和小部件标志*f*，参数被传递给QFrame构造函数。

也可以看看setAlignment(),setFrameStyle（），和setIndent()。

*[explicit]QLabel::QLabel(const QString &text, QWidget *parent = nullptr, Qt::WindowFlags f = Qt::WindowFlags())*

构造一个显示文本的标签，*text*。

这*parent*和小部件标志*f*，参数被传递给QFrame构造函数。

也可以看看setText(),setAlignment(),setFrameStyle（），和setIndent()。

[virtual]QLabel::~QLabel()

破坏标签。

*QWidget *QLabel::buddy() const*

返回此标签的好友，如果当前未设置好友，则返回 nullptr。

也可以看看setBuddy()。

*[override virtual protected]void QLabel::changeEvent(QEvent *ev)*

重新实现：QFrame::changeEvent（QEvent *ev）。

[slot]void QLabel::clear()

清除所有标签内容。

[override virtual protected]void QLabel::contextMenuEvent(QContextMenuEvent *ev)

重新实现：QWidget::contextMenuEvent（QContextMenuEvent *事件）。

[override virtual protected]bool QLabel::event(QEvent *e)

重新实现：QFrame::event（QEvent *e）。

[override virtual protected]void QLabel::focusInEvent(QFocusEvent *ev)

重新实现：QWidget::focusInEvent（QFocusEvent *事件）。

[override virtual protected]bool QLabel::focusNextPrevChild(bool next)

重新实现：QWidget::focusNextPrevChild（接下来是布尔值）。

[override virtual protected]void QLabel::focusOutEvent(QFocusEvent *ev)

重新实现：QWidget::focusOutEvent（QFocusEvent *事件）。

[override virtual]int QLabel::heightForWidth(int w) const

重新实现：QWidget::heightForWidth(int w) const。

*[override virtual protected]void QLabel::keyPressEvent(QKeyEvent *ev)*

重新实现：QWidget::keyPressEvent（QKeyEvent *事件）。

[signal]void QLabel::linkActivated(const QString &link)

当用户单击链接时会发出此信号。传入锚点引用的 URL*link*。

也可以看看linkHovered()。

[signal]void QLabel::linkHovered(const QString &link)

当用户将鼠标悬停在链接上时会发出此信号。传入锚点引用的 URL*link*。

也可以看看linkActivated()。

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

重新实现属性的访问函数：QWidget::minimumSizeHint。

*[override virtual protected]void QLabel::mouseMoveEvent(QMouseEvent *ev)*

重新实现：QWidget::mouseMoveEvent（QMouseEvent *事件）。

*[override virtual protected]void QLabel::mousePressEvent(QMouseEvent *ev)*

重新实现：QWidget::mousePressEvent（QMouseEvent *事件）。

*[override virtual protected]void QLabel::mouseReleaseEvent(QMouseEvent *ev)*

重新实现：QWidget::mouseReleaseEvent（QMouseEvent *事件）。

*QMovie *QLabel::movie() const*

返回指向标签电影的指针，如果未设置电影，则返回 nullptr。

也可以看看setMovie()。

*[override virtual protected]void QLabel::paintEvent(QPaintEvent *)*

重新实现：QFrame::paintEvent（QPaintEvent *）。

[since 6.0]QPicture QLabel::picture() const

返回标签的图片。

这个函数是在Qt 6.0中引入的。

[since 6.1]QTextDocument::ResourceProvider QLabel::resourceProvider() const

返回此标签的富文本的资源提供者。

该功能是在 Qt 6.1 中引入的。

也可以看看setResourceProvider()。

int QLabel::selectionStart() const

SelectionStart() 返回标签中第一个选定字符的索引，如果未选择任何文本，则返回 -1。

注：textInteractionFlags标签上的设置需要包含 TextSelectableByMouse 或 TextSelectableByKeyboard。

也可以看看selectedText()。

*void QLabel::setBuddy(QWidget *buddy)*

将此标签的好友设置为*buddy*。

当用户按下该标签指示的快捷键时，键盘焦点将转移到该标签的好友小部件。

好友机制仅适用于包含其中一个字符以与符号“&”为前缀的文本的 QLabels。该字符被设置为快捷键。请参阅 QKeySequence::mnemonic() 文档了解详细信息（要显示实际的 & 符号，请使用“&&”）。

在对话框中，您可以创建两个数据输入小部件并为每个小部件创建一个标签，并设置几何布局，以便每个标签位于其数据输入小部件（其“伙伴”）的左侧，例如：

```
QLineEdit *nameEdit  = new QLineEdit(this);
QLabel    *nameLabel = new QLabel("&Name:", this);
nameLabel->setBuddy(nameEdit);
QLineEdit *phoneEdit  = new QLineEdit(this);
QLabel    *phoneLabel = new QLabel("&Phone:", this);
phoneLabel->setBuddy(phoneEdit);
// (layout setup not shown)
```

使用上面的代码，当用户按 Alt+N 时焦点跳转到“名称”字段，当用户按 Alt+P 时焦点跳转到“电话”字段。

要取消设置先前设置的好友，请使用以下命令调用此函数*buddy*设置为 nullptr。

也可以看看buddy(),setText(),QShortcut， 和setAlignment()。

*[slot]void QLabel::setMovie(QMovie *movie)*

将标签内容设置为*movie*。之前的所有内容都会被清除。唱片公司不拥有电影的所有权。

好友快捷方式（如果有）被禁用。

也可以看看movie （） 和setBuddy()。

[slot]void QLabel::setNum(int num)

将标签内容设置为包含整数文本表示的纯文本*num*。之前的所有内容都会被清除。如果整数的字符串表示形式与标签的当前内容相同，则不执行任何操作。

好友快捷方式（如果有）被禁用。

也可以看看setText(),QString::setNum （） ， 和setBuddy()。

[slot]void QLabel::setNum(double num)

这是一个重载功能。

将标签内容设置为包含 double 文本表示的纯文本*num*。之前的所有内容都会被清除。如果双精度型的字符串表示形式与标签的当前内容相同，则不执行任何操作。

好友快捷方式（如果有）被禁用。

也可以看看setText(),QString::setNum（），和setBuddy()。

[slot]void QLabel::setPicture(const QPicture &picture)

将标签内容设置为*picture*。之前的所有内容都会被清除。

好友快捷方式（如果有）被禁用。

也可以看看picture（） 和setBuddy()。

[since 6.1]void QLabel::setResourceProvider(const QTextDocument::ResourceProvider &provider)

设置*provider*此标签的富文本资源。

注意：该标签*不*拥有所有权*provider*。

该功能是在 Qt 6.1 中引入的。

也可以看看resourceProvider()。

void QLabel::setSelection(int start, int length)

从位置选择文本*start*并为*length*人物。

注：textInteractionFlags标签上的设置需要包含 TextSelectableByMouse 或 TextSelectableByKeyboard。

也可以看看selectedText()。

[override virtual]QSize QLabel::sizeHint() const

重新实现：QFrame::sizeHint() const。