QLayout Class

QLayout 类是几何管理器的基类。更多的...

Header:	**#include < QLayout>**
CMake:	find_package(Qt6 REQUIRED COMPONENTS Widgets) target_link_libraries(mytarget PRIVATE Qt6::Widgets)
qmake:	QT += widgets
Inherits:	QObject and QLayoutItem
Inherited By:	QBoxLayout, QFormLayout, QGridLayout, and QStackedLayout

- 所有成员的列表，包括继承的成员

公共类型

enum	**SizeConstraint { SetDefaultConstraint, SetFixedSize, SetMinimumSize, SetMaximumSize, SetMinAndMaxSize, SetNoConstraint }**

特性

- **sizeConstraint** : SizeConstraint
- **spacing** : int

公共函数

	QLayout(QWidget **parent* = nullptr)
bool	**activate**()
virtual void	**addItem**(QLayoutItem **item*) = 0
void	**addWidget**(QWidget **w*)
QMargins	**contentsMargins**() const

	QLayout(QWidget **parent* = nullptr)
QRect	**contentsRect**() const
virtual int	**count**() const = 0
void	**getContentsMargins**(int **left*, int** top*, int **right*, int **bottom*) const
virtual int	**indexOf**(const QWidget **widget*) const
virtual int	**indexOf**(const QLayoutItem **layoutItem*) const
bool	**isEnabled**() const
virtual QLayoutItem *	**itemAt**(int *index*) const = 0
QWidget *	**menuBar**() const
QWidget *	**parentWidget**() const
void	**removeItem**(QLayoutItem **item*)
void	**removeWidget**(QWidget **widget*)
virtual QLayoutItem *	**replaceWidget**(QWidget **from*, QWidget** to*, Qt::FindChildOptions *options* = Qt::FindChildrenRecursively)
bool	**setAlignment**(QWidget **w*, Qt::Alignment *alignment*)
bool	**setAlignment**(QLayout **l*, Qt::Alignment *alignment*)
void	**setContentsMargins**(int *left*, int *top*, int *right*, int *bottom*)
void	**setContentsMargins**(const QMargins &*margins*)
void	**setEnabled**(bool *enable*)
void	**setMenuBar**(QWidget **widget*)
void	**setSizeConstraint**(QLayout::SizeConstraint)
virtual void	**setSpacing**(int)
QLayout::SizeConstraint	**sizeConstraint**() const
virtual int	**spacing**() const
virtual QLayoutItem *	**takeAt**(int *index*) = 0
void	**unsetContentsMargins**()
void	**update**()

重载的公共函数

virtual QSizePolicy::ControlTypes	**controlTypes() const override**

virtual QSizePolicy::ControlTypes	**controlTypes() const override**
virtual Qt::Orientations	**expandingDirections**() const override
virtual QRect	**geometry**() const override
virtual void	**invalidate**() override
virtual bool	**isEmpty**() const override
virtual QLayout *	**layout**() override
virtual QSize	**maximumSize**() const override
virtual QSize	**minimumSize**() const override
virtual void	**setGeometry**(const QRect &*r*) override

静态公共成员

QSize	**closestAcceptableSize**(**const QWidget** ***widget*, const QSize &*size*)**

protected function

void	**addChildLayout**(**QLayout** ***childLayout*)**
void	**addChildWidget**(QWidget **w*)
QRect	**alignmentRect**(const QRect &*r*) const

重载的protected function

virtual void	**childEvent**(**QChildEvent** ***e*) override**

详细说明

这是一个由具体类继承的抽象基类QBoxLayout,QGridLayout,QFormLayout， 和QStackedLayout。

对于 QLayout 子类或QMainWindow很少需要用到QLayout提供的基本功能，比如setSizeConstraint （） 或者setMenuBar()。看Layout Management了解更多信息。

要制作自己的布局管理器，请实现以下功能addItem(),sizeHint(),setGeometry(),itemAt（） 和takeAt()。您还应该实施minimumSize() 以确保如果空间太小，您的布局不会调整为零大小。为了支持身高取决于宽度的儿童，请实施hasHeightForWidth（） 和heightForWidth()。请参阅Border Layout和Flow Layout有关实现自定义布局管理器的更多信息的示例。

当布局管理器被删除时，几何管理将停止。

也可以看看 QLayoutItem,Layout Management,Basic Layouts Example,Border Layout Example， 和 Flow Layout Example。

会员类型文档

enum QLayout::SizeConstraint

可能的值为：

持续的	价值	描述
`QLayout::SetDefaultConstraint`	`0`	主部件的最小尺寸设置为minimumSize()，除非小部件已经有最小尺寸。
`QLayout::SetFixedSize`	`3`	主小部件的大小设置为sizeHint(); 它根本无法调整大小。
`QLayout::SetMinimumSize`	`2`	主部件的最小尺寸设置为minimumSize(); 它不能更小。
`QLayout::SetMaximumSize`	`4`	主部件的最大尺寸设置为maximumSize(); 它不能更大。
`QLayout::SetMinAndMaxSize`	`5`	主部件的最小尺寸设置为minimumSize() 其最大尺寸设置为maximumSize()。
`QLayout::SetNoConstraint`	`1`	小部件不受限制。

也可以看看setSizeConstraint()。

属性文档

sizeConstraint : SizeConstraint

该属性保存布局的调整大小模式

默认模式是SetDefaultConstraint。

访问功能：

QLayout::SizeConstraint	sizeConstraint() const
void	**setSizeConstraint**(QLayout::SizeConstraint)

spacing : int

该属性保存布局内小部件之间的间距

如果未显式设置值，则布局的间距将从父布局或父窗口小部件的样式设置继承。

为了 QGridLayout 和 QFormLayout，可以使用设置不同的水平和垂直间距 setHorizontalSpacing() 和 setVerticalSpacing()。在这种情况下，spacing() 返回 -1。

访问功能：

virtual int	**spacing() const**
virtual void	**setSpacing**(int)

也可以看看contentsRect(),getContentsMargins(),QStyle::layoutSpacing()，和QStyle::pixelMetric()。

成员函数文档

*[explicit]QLayout::QLayout(QWidget *parent = nullptr)*

构造一个新的顶级 QLayout，带有父级*parent*。

该布局直接设置为顶层布局*parent*。一个小部件只能有一个顶级布局。它由以下方式返回QWidget::layout()。

如果*parent*是`nullptr`，那么您必须将此布局插入到另一个布局中，或者使用以下命令将其设置为小部件的布局 QWidget::setLayout()。

也可以看看QWidget::setLayout()。

bool QLayout::activate()

重做布局parentWidget() 如有必要。

您通常不需要调用它，因为它会在最合适的时间自动调用。如果重做布局，则返回 true。

也可以看看update() 和QWidget::updateGeometry()。

*[protected]void QLayout::addChildLayout(QLayout *childLayout)*

`addLayout()`从子类或子类中调用此函数`insertLayout()`以添加布局*childLayout*作为子布局。

唯一需要直接调用它的场景是如果您实现支持嵌套布局的自定义布局。

也可以看看QBoxLayout::addLayout(),QBoxLayout::insertLayout()，和QGridLayout::addLayout()。

[protected]void QLayout::addChildWidget(QWidget *w)

从子类中的函数调用此函数addWidget()来添加*w*作为布局的托管小部件。

如果*w*已经由布局管理，此函数将产生警告，并删除*w*从那个布局。因此必须在添加之前调用此函数*w*布局的数据结构。

[pure virtual]void QLayout::addItem(QLayoutItem *item)

在子类中实现以添加*item*。它的添加方式特定于每个子类。

该函数通常不会在应用程序代码中调用。要将小部件添加到布局，请使用addWidget（） 功能; 要添加子布局，请使用相关提供的 addLayout() 函数QLayout子类。

注：所有权*item*被转移到布局中，布局负责删除它。

也可以看看addWidget(),QBoxLayout::addLayout（），和QGridLayout::addLayout()。

void QLayout::addWidget(QWidget *w)

添加小部件*w*以特定于该布局的方式对该布局进行修改。该函数使用addItem()。

[protected]QRect QLayout::alignmentRect(const QRect &r) const

返回当此布局的几何形状设置为时应覆盖的矩形*r*，前提是该布局支持setAlignment()。

结果源自sizeHint（） 和expandingDirections()。它永远不会大于*r*。

[override virtual protected]void QLayout::childEvent(QChildEvent *e)

重新实现：QObject::childEvent（QChildEvent *事件）。

[static]QSize QLayout::closestAcceptableSize(const QWidget *widget, const QSize &size)

返回满足所有大小限制的大小*widget*， 包括heightForWidth() 并且尽可能接近*size*。

QMargins QLayout::contentsMargins() const

返回布局周围使用的边距。

默认情况下，QLayout使用样式提供的值。在大多数平台上，所有方向的边距均为 11 像素。

注意：属性contentsMargins的Getter函数。

也可以看看setContentsMargins()。

QRect QLayout::contentsRect() const

返回布局的geometry() 矩形，但考虑到内容边距。

也可以看看setContentsMargins （） 和getContentsMargins()。

[override virtual]QSizePolicy::ControlTypes QLayout::controlTypes() const

重新实现：QLayoutItem::controlTypes() const。

[pure virtual]int QLayout::count() const

必须在子类中实现以返回布局中的项目数。

也可以看看itemAt()。

[override virtual]Qt::Orientations QLayout::expandingDirections() const

重新实现：QLayoutItem::expandingDirections() const。

返回此布局是否可以利用比sizeHint()。值为Qt::Vertical或者Qt::Horizontal意味着它只想在一个维度上增长，而Qt::Vertical|Qt::Horizontal意味着它想要在两个维度上都增长。

默认实现返回Qt::Horizontal|Qt::Vertical。子类重新实现它以根据其子窗口小部件返回有意义的值size policies。

也可以看看sizeHint()。

[override virtual]QRect QLayout::geometry() const

重新实现：QLayoutItem::geometry() const。

也可以看看setGeometry()。

*void QLayout::getContentsMargins(int **left*, int** *top, int **right, int *bottom) const*

对于每个*left*,*top*,*right*和*bottom*那不是`nullptr`，存储在指针引用的位置中命名的边距的大小。

默认情况下，QLayout使用样式提供的值。在大多数平台上，所有方向的边距均为 11 像素。

也可以看看setContentsMargins(),QStyle::pixelMetric(),PM_LayoutLeftMargin,PM_LayoutTopMargin,PM_LayoutRightMargin，和PM_LayoutBottomMargin。

*[virtual]int QLayout::indexOf(const QWidget *widget) const*

搜索小部件*widget*在此布局中（不包括子布局）。

返回索引*widget*，或 -1 如果*widget*没有找到。

默认实现使用迭代所有项目itemAt()。

*[virtual]int QLayout::indexOf(const QLayoutItem *layoutItem) const*

搜索布局项*layoutItem*在此布局中（不包括子布局）。

返回索引*layoutItem*，或 -1 如果*layoutItem*没有找到。

[override virtual]void QLayout::invalidate()

重新实现：QLayoutItem::invalidate()。

[override virtual]bool QLayout::isEmpty() const

重新实现：QLayoutItem::isEmpty() const。

bool QLayout::isEnabled() const

返回`true`布局是否启用；否则返回`false`.

也可以看看setEnabled()。

*[pure virtual]QLayoutItem *QLayout::itemAt(int index) const*

必须在子类中实现才能返回布局项*index*。如果没有这样的项目，该函数必须返回`nullptr`。项目从 0 开始连续编号。如果删除某个项目，其他项目将重新编号。

此函数可用于迭代布局。以下代码将为小部件布局结构中的每个布局项绘制一个矩形。

```
static void paintLayout(QPainter *painter, QLayoutItem *item)
{
    QLayout *layout = item->layout();
    if (layout) {
        for (int i = 0; i < layout->count(); ++i)
            paintLayout(painter, layout->itemAt(i));
    }
    painter->drawRect(item->geometry());
}

void MyWidget::paintEvent(QPaintEvent *)
{
    QPainter painter(this);
    if (layout())
        paintLayout(&painter, layout());
}
```

也可以看看count（） 和takeAt()。

*[override virtual]QLayout *QLayout::layout()*

重新实现：QLayoutItem::layout()。

[override virtual]QSize QLayout::maximumSize() const

重新实现：QLayoutItem::maximumSize() const。

返回此布局的最大尺寸。这是布局在仍遵守规范的情况下可以具有的最大尺寸。

返回值不包含所需的空间QWidget::setContentsMargins（） 或者menuBar()。

默认实现允许无限制地调整大小。

QWidget *QLayout::menuBar() const*

返回为此布局设置的菜单栏，或者nullptr如果未设置菜单栏。

也可以看看setMenuBar()。

[override virtual]QSize QLayout::minimumSize() const

重新实现：QLayoutItem::minimumSize() const。

返回此布局的最小尺寸。这是布局在仍遵守规范的情况下可以具有的最小尺寸。

返回值不包含所需的空间QWidget::setContentsMargins（） 或者menuBar()。

默认实现允许无限制地调整大小。

QWidget *QLayout::parentWidget() const*

返回此布局的父窗口小部件，或者nullptr如果此布局未安装在任何窗口小部件上。

如果布局是子布局，则此函数返回父布局的父窗口小部件。

也可以看看parent()。

*void QLayout::removeItem(QLayoutItem *item)*

删除布局项*item*从布局来看。调用者有责任删除该项目。

请注意*item*可以是一个布局（因为QLayout继承QLayoutItem）。

也可以看看removeWidget（） 和addItem()。

*void QLayout::removeWidget(QWidget *widget)*

删除小部件*widget*从布局来看。在此调用之后，调用者有责任为小部件提供合理的几何形状，或者将小部件放回到布局中，或者在必要时显式隐藏它。

注：所有权*widget*与添加时保持相同。

也可以看看removeItem(),QWidget::setGeometry（），和addWidget()。

[virtual] QLayoutItem QLayout::replaceWidget(QWidget **from, QWidget *to, Qt::FindChildOptions options = Qt::FindChildrenRecursively)

搜索小部件*from*并将其替换为小部件*to*如果找到的话。返回包含小部件的布局项*from*关于成功。否则nullptr返回。如果*options*包含Qt::FindChildrenRecursively（默认），搜索子布局以进行替换。任何其他标志*options*被忽略。

请注意，因此返回的项目可能不属于此布局，而是属于子布局。

返回的布局项不再属于该布局，应将其删除或插入到另一个布局中。小部件*from*不再由布局管理，可能需要删除或隐藏。小部件的父级*from*保持不变。

此函数适用于内置 Qt 布局，但可能不适用于自定义布局。

也可以看看indexOf()。

bool QLayout::setAlignment(QWidget *w, Qt::Alignment alignment)

设置小部件的对齐方式*w*到*alignment*并返回 true 如果*w*在此布局中找到（不包括子布局）；否则返回false.

bool QLayout::setAlignment(QLayout *l, Qt::Alignment alignment)

这是一个过载功能。

设置布局的对齐方式*l*到*alignment*并返回true如果*l*在此布局中找到（不包括子布局）；否则返回false.

void QLayout::setContentsMargins(int left, int top, int right, int bottom)

设置*left*,*top*,*right*， 和*bottom*布局周围使用的边距。

默认情况下，QLayout使用样式提供的值。在大多数平台上，所有方向的边距均为 11 像素。

注意：属性的 Setter 函数contentsMargins。

也可以看看contentsMargins(),getContentsMargins(),QStyle::pixelMetric(),PM_LayoutLeftMargin,PM_LayoutTopMargin,PM_LayoutRightMargin， 和PM_LayoutBottomMargin。

void QLayout::setContentsMargins(const QMargins &margins)

设置*margins*在布局周围使用。

默认情况下，QLayout使用样式提供的值。在大多数平台上，所有方向的边距均为 11 像素。

注意：属性的 Setter 函数contentsMargins。

也可以看看contentsMargins()。

void QLayout::setEnabled(bool enable)

如果满足以下条件，则启用此布局*enable*为 true，否则禁用它。

启用的布局会根据变化动态调整；禁用的布局就像不存在一样。

默认情况下，所有布局均已启用。

也可以看看isEnabled()。

[override virtual]void QLayout::setGeometry(const QRect &r)

重新实现：QLayoutItem::setGeometry（常量 QRect &r）。

也可以看看geometry()。

*void QLayout::setMenuBar(QWidget *widget)*

告诉几何管理器放置菜单栏*widget*在顶端parentWidget（），外部QWidget::contentsMargins()。所有子小部件都放置在菜单栏底部边缘的下方。

也可以看看menuBar()。

*[pure virtual]QLayoutItem *QLayout::takeAt(int index)*

必须在子类中实现以删除位于*index*从布局中，并返回该项目。如果不存在这样的项目，则该函数必须不执行任何操作并返回 0。项目从 0 开始连续编号。如果删除某个项目，则其他项目将重新编号。

以下代码片段显示了从布局中删除所有项目的安全方法：

```
QLayoutItem *child;
while ((child = layout->takeAt(0)) != nullptr) {
    ...
    delete child->widget(); // delete the widget
    delete child;   // delete the layout item
}
```

也可以看看itemAt（） 和count()。

[since 6.1]void QLayout::unsetContentsMargins()

取消设置布局周围任何用户定义的边距。布局将使用样式提供的默认值。

注：属性重置功能contentsMargins。

该功能是在 Qt 6.1 中引入的。

也可以看看setContentsMargins()。

void QLayout::update()

更新布局parentWidget()。

您通常不需要调用它，因为它会在最合适的时间自动调用。

也可以看看activate （） 和invalidate()。