# QSplitter Class

QSplitter 类实现了一个分割器小部件。更多的...

| | |
|---|---|
| **Header:** | **#include < QSplitter>** |
| 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

## 特性

- **childrenCollapsible** : bool
- **handleWidth** : int
- **opaqueResize** : bool
- **orientation** : Qt::Orientation

## 公共职能

| | |
|---|---|
| | **QSplitter**(**QWidget** ***parent** = **nullptr**) |
| | **QSplitter**(Qt::Orientation *orientation*, QWidget **parent* = nullptr) |
| virtual | **~QSplitter**() |
| void | **addWidget**(QWidget **widget*) |
| bool | **childrenCollapsible**() const |
| int | **count**() const |
| void | **getRange**(int *index*, int **min*, **int** max*) const |
| QSplitterHandle * | **handle**(int *index*) const |
| int | **handleWidth**() const |
| int | **indexOf**(QWidget **widget*) const |
| void | **insertWidget**(int *index*, QWidget **widget*) |

| | |
|---|---|
| bool | **QSplitter**(QWidget **parent* = nullptr) |
| bool | **isCollapsible**(int *index*) const |
| bool | **opaqueResize**() const |
| Qt::Orientation | **orientation**() const |
| void | **refresh**() |
| QWidget * | **replaceWidget**(int *index*, QWidget **widget*) |
| bool | **restoreState**(const QByteArray &*state*) |
| QByteArray | **saveState**() const |
| void | **setChildrenCollapsible**(bool) |
| void | **setCollapsible**(int *index*, bool *collapse*) |
| void | **setHandleWidth**(int) |
| void | **setOpaqueResize**(bool *opaque* = true) |
| void | **setOrientation**(Qt::Orientation) |
| void | **setSizes**(const QList &*list*) |
| void | **setStretchFactor**(int *index*, int *stretch*) |
| QList | **sizes**() const |
| QWidget * | **widget**(int *index*) const |

## 重载的公共职能

| | |
|---|---|
| **virtual QSize** | **minimumSizeHint**() const override |
| virtual QSize | **sizeHint**() const override |

## 信号

| | |
|---|---|
| **void** | **splitterMoved**(**int *pos*, int *index***) |

## protected function

| int | closestLegalPosition(int *pos*, int *index*) |
| --- | --- |
| virtual QSplitterHandle * | createHandle() |
| void | moveSplitter(int *pos*, int *index*) |
| void | setRubberBand(int *pos*) |

## 重新实现protected function

| virtual void | changeEvent(QEvent **ev*) override |
| --- | --- |
| virtual void | childEvent(QChildEvent **c*) override |
| virtual bool | event(QEvent **e*) override |
| virtual void | resizeEvent(QResizeEvent *) override |

## 详细说明

拆分器允许用户通过拖动子部件之间的边界来控制子部件的大小。单个分离器可以控制任意数量的小部件。QSplitter 的典型用途是创建多个小部件并使用添加它们insertWidget（） 或者addWidget()。

下面的例子将展示一个QListView,QTreeView， 和QTextEdit并排，带有两个分离手柄：

```
QSplitter *splitter = new QSplitter(parent);
QListView *listview = new QListView;
QTreeView *treeview = new QTreeView;
QTextEdit *textedit = new QTextEdit;
splitter->addWidget(listview);
splitter->addWidget(treeview);
splitter->addWidget(textedit);
```

如果一个小部件已经在 QSplitter 中insertWidget（） 或者addWidget() 被调用，它将移动到新的位置。这可用于稍后在拆分器中重新排序小部件。您可以使用indexOf(),widget（），和count() 来访问拆分器内的小部件。

默认的 QSplitter 水平布置其子级（并排）；您可以使用setOrientation（Qt::Vertical）将其子项垂直放置。

默认情况下，所有小部件都可以根据用户的意愿设置为大或小，介于minimumSizeHint（） （或者minimumSize（）） 和maximumSize() 的小部件。

默认情况下，QSplitter 会动态调整其子级的大小。如果您希望 QSplitter 仅在调整大小操作结束时调整子级的大小，请调用setOpaqueResize（错误的）。

小部件之间的初始大小分布是通过将初始大小乘以拉伸因子来确定的。您还可以使用setSizes() 设置所有小部件的大小。功能sizes() 返回用户设置的尺寸。或者，您可以从以下位置保存和恢复小部件的大小：QByteArray使用saveState（） 和restoreState（） 分别。

当你hide() 一个孩子，它的空间将分配给其他孩子。当您show() 再说一遍。

**注意：**添加一个QLayout不支持 QSplitter（通过setLayout() 或使 QSplitter 成为QLayout）；使用addWidget() 代替（参见上面的示例）。

**也可以看看**QSplitterHandle,QHBoxLayout,QVBoxLayout， 和QTabWidget。

# 属性文档

## *childrenCollapsible : bool*

该属性保存用户是否可以将子窗口小部件调整为大小 0

默认情况下，子项是可折叠的。可以使用以下命令启用和禁用单个子项的折叠setCollapsible()。

**访问功能：**

| | |
|---|---|
| **bool** | **childrenCollapsible() const** |
| void | **setChildrenCollapsible**(bool) |

**也可以看看**setCollapsible()。

## *handleWidth : int*

该属性保存分割器手柄的宽度

默认情况下，此属性包含一个取决于用户平台和样式首选项的值。

如果将 handleWidth 设置为 1 或 0，则实际抓取区域将增长以重叠其各自小部件的几个像素。

**访问功能：**

| | |
|---|---|
| **int** | **handleWidth() const** |
| void | **setHandleWidth**(int) |

## *opaqueResize : bool*

返回`true`是否在交互式移动拆分器时动态（不透明）调整小部件的大小。否则返回`false`。

默认调整大小行为取决于样式（由 SH_Splitter_OpaqueResize 样式提示确定）。但是，您可以通过调用 setOpaqueResize() 来覆盖它

**访问功能：**

| | |
|---|---|
| **bool** | **opaqueResize() const** |
| void | **setOpaqueResize**(bool *opaque* = true) |

**也可以看看**QStyle::StyleHint。

### *orientation : Qt::Orientation*

该属性保存分离器的方向

默认情况下，方向是水平的（即，小部件并排布置）。可能的方向是Qt::Horizontal和Qt::Vertical。

**访问功能：**

| Qt::Orientation | orientation() const |
| --- | --- |
| void | **setOrientation**(Qt::Orientation) |

**也可以看看**QSplitterHandle::orientation()。

## 成员函数文档

### *[explicit]QSplitter::QSplitter(QWidget *parent = nullptr)*

构造一个水平分离器*parent*论点传递给QFrame构造函数。

**也可以看看**setOrientation()。

### *[explicit]QSplitter::QSplitter(Qt::Orientation orientation, QWidget *parent = nullptr)*

使用给定的构造一个分离器*orientation*和*parent*。

**也可以看看**setOrientation()。

### *[virtual]QSplitter::~QSplitter()*

破坏分离器。所有子项均被删除。

### *void QSplitter::addWidget(QWidget *widget)*

添加给定的*widget*在所有其他项目之后的拆分器布局。

如果*widget*已在分离器中，它将被移动到新位置。

**注意：**拆分器拥有该小部件的所有权。

**也可以看看**insertWidget(),widget（），和indexOf()。

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

重新实现：QFrame::changeEvent（QEvent *ev）。

## *[override virtual protected]void QSplitter::childEvent(QChildEvent *c)*

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

告诉分割器所描述的子部件*c*已被插入或移除。

此方法还用于处理使用拆分器作为父级创建小部件但未显式添加的情况insertWidget（） 或者addWidget()。这是为了兼容性，而不是将小部件放入新代码中的拆分器中的推荐方法。请用insertWidget（） 或者addWidget() 在新代码中。

**也可以看看**addWidget（） 和insertWidget()。

## *[protected]int QSplitter::closestLegalPosition(int pos, int index)*

返回最接近的合法位置*pos*小部件的*index*。

对于从右到左的语言，例如阿拉伯语和希伯来语，水平分隔符的布局是相反的。然后从小部件的右边缘开始测量位置。

**也可以看看**getRange()。

## *int QSplitter::count() const*

返回分割器布局中包含的小部件的数量。

**也可以看看**widget（） 和handle()。

## *[virtual protected]QSplitterHandle *QSplitter::createHandle()*

返回一个新的拆分器句柄作为该拆分器的子小部件。该函数可以在子类中重新实现，以提供对自定义句柄的支持。

**也可以看看**handle（） 和indexOf()。

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

重新实现：QFrame::event（QEvent *e）。

## *void QSplitter::getRange(int index, int **min*, int** max*) const*

返回分割器的有效范围*index*在 **min*和** max*如果*min*和*max*不为 0。

## *QSplitterHandle *QSplitter::handle(int index) const*

返回给定分配器布局中项目左侧（或上方）的句柄*index*，或者`nullptr`如果没有这样的项目。索引 0 处的句柄始终隐藏。

对于从右到左的语言，例如阿拉伯语和希伯来语，水平分隔符的布局是相反的。句柄将位于小部件的右侧*index*。

**也可以看看**count(),widget(),indexOf(),createHandle（），和setHandleWidth()。

## *int QSplitter::indexOf(QWidget *widget) const*

返回指定拆分器布局中的索引*widget*，或 -1 如果*widget*没有找到。这也适用于手柄。

句柄从 0 开始编号。句柄的数量与子控件的数量一样多，但位置 0 处的句柄始终是隐藏的。

**也可以看看**count（）和widget()。

## *void QSplitter::insertWidget(int index, QWidget *widget)*

插入*widget*指定到给定分配器的布局中*index*。

如果*widget*已在分离器中，它将被移动到新位置。

如果*index*是无效索引，则小部件将被插入到末尾。

**注意：**拆分器拥有该小部件的所有权。

**也可以看看**addWidget(),indexOf（），和widget()。

## *bool QSplitter::isCollapsible(int index) const*

`true`如果小部件位于*index*是可折叠的，否则返回`false`.

## *[override virtual]QSize QSplitter::minimumSizeHint() const*

重新实现属性的访问函数：QWidget::minimumSizeHint。

## *[protected]void QSplitter::moveSplitter(int pos, int index)*

将分割器手柄的左边缘或上边缘移动到*index*尽可能靠近位置*pos*，这是距小部件左边缘或上边缘的距离。

对于从右到左的语言，例如阿拉伯语和希伯来语，水平分隔符的布局是相反的。*pos*那么就是距小部件右边缘的距离。

**也可以看看**splitterMoved(),closestLegalPosition（），和getRange()。

## *void QSplitter::refresh()*

更新分离器的状态。您不需要调用此函数。

## *QWidget *QSplitter::replaceWidget(int index, QWidget *widget)*

替换给定分割器布局中的小部件*index*经过*widget*。

返回刚刚被替换的小部件，如果*index*是有效的并且*widget*还不是 splitter 的子级。否则，返回 null 并且不进行替换或添加。

新插入的小部件的几何形状将与其替换的小部件相同。它的可见和折叠状态也是继承的。

**注：**分离器拥有所有权*widget*并将替换的小部件的父级设置为 null。

**注：**因为*widget*得到reparented进入分离器，其geometry可能不会立即设置，但只能在之后设置*widget*将收到适当的事件。

**也可以看看**insertWidget（）和indexOf()。

## *[override virtual protected]void QSplitter::resizeEvent(QResizeEvent *)*

重新实现：QWidget::resizeEvent（QResizeEvent *事件）。

## *bool QSplitter::restoreState(const QByteArray &state)*

将分离器的布局恢复为*state*指定的。`true`如果状态恢复则返回；否则返回`false`.

通常这与QSettings恢复过去会话的大小。这是一个例子：

恢复分离器的状态：

```
    QSettings settings;
    splitter->restoreState(settings.value("splitterSizes").toByteArray());
```

无法恢复拆分器的布局可能是由于所提供的字节数组中的数据无效或过时造成的。

**也可以看看**saveState()。

## *QByteArray QSplitter::saveState() const*

保存分割器布局的状态。

通常这与QSettings记住该大小以供将来的会话使用。版本号作为数据的一部分存储。这是一个例子：

```
    QSettings settings;
    settings.setValue("splitterSizes", splitter->saveState());
```

**也可以看看**restoreState()。

## *void QSplitter::setCollapsible(int index, bool collapse)*

设置子部件是否位于*index*可以折叠到*collapse*。

默认情况下，子项是可折叠的，这意味着用户可以将它们的大小调整为 0，即使它们的大小非零minimumSize（）或者minimumSizeHint()。可以通过调用此函数在每个小部件的基础上更改此行为，或者通过设置拆分器中的所有小部件来全局更改此行为childrenCollapsible财产。

**也可以看看**isCollapsible（） 和childrenCollapsible。

## *[protected]void QSplitter::setRubberBand(int pos)*

在位置显示橡皮筋*pos*。如果*pos*为负值，橡皮筋被移除。

## *void QSplitter::setSizes(const QList &list)*

将子窗口部件各自的大小设置为中给出的值*list*。

如果拆分器是水平的，则这些值设置每个小部件的宽度（以像素为单位），从左到右。如果拆分器是垂直的，则从上到下设置每个小部件的高度。

额外的价值*list*被忽略。如果*list*包含的值太少，结果是未定义的，但程序仍然会表现良好。

拆分器小部件的整体大小不受影响。相反，任何额外/缺失的空间都会根据尺寸的相对权重分布在小部件之间。

如果指定大小为 0，则小部件将不可见。保留小部件的大小策略。也就是说，小于相应小部件的最小尺寸提示的值将被提示的值替换。

**也可以看看**sizes()。

## *void QSplitter::setStretchFactor(int index, int stretch)*

更新位置小部件的大小策略*index*拉伸因子为*stretch*。

*stretch*不是有效拉伸因子；有效拉伸因子是通过获取小部件的初始大小并将其乘以来计算的*stretch*。

提供此功能是为了方便。它相当于

```
QWidget *widget = splitter->widget(index);
QSizePolicy policy = widget->sizePolicy();
policy.setHorizontalStretch(stretch);
policy.setVerticalStretch(stretch);
widget->setSizePolicy(policy);
```

**也可以看看**setSizes（） 和widget()。

## *[override virtual]QSize QSplitter::sizeHint() const*

重新实现：QFrame::sizeHint() const。

## *QList* QSplitter::sizes() const

返回此拆分器中所有小部件的大小参数的列表。

如果拆分器的方向是水平的，则列表从左到右包含以像素为单位的小部件宽度；如果方向是垂直的，则列表包含小部件从上到下的高度（以像素为单位）。

将值赋予另一个分离器setSizes() 函数将生成一个与此布局相同的拆分器。

请注意，不可见部件的大小为 0。

**也可以看看**setSizes()。

## `[signal]`void QSplitter::splitterMoved(int pos, int index)

当分离器手柄处于特定位置时会发出此信号*index*已移至位置*pos*。

对于从右到左的语言，例如阿拉伯语和希伯来语，水平分隔符的布局是相反的。*pos*那么就是距小部件右边缘的距离。

**也可以看看**moveSplitter()。

## *QWidget* *QSplitter::widget(int index) const

返回给定位置的小部件*index*在分割器的布局中，或者`nullptr`如果没有这样的小部件。

**也可以看看**count(),handle(),indexOf（），和insertWidget()。