QTabWidget Class

QTabWidget 类提供了一堆选项卡式小部件。更多的...

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

- 所有成员的列表，包括继承的成员

公共类型

enum	**TabPosition { North, South, West, East }**
enum	**TabShape** { Rounded, Triangular }

特性

count : const intcurrentIndex : intdocumentMode : boolelideMode : Qt::TextElideModeiconSize : QSizemovable : bool	**tabBarAutoHide : booltabPosition : TabPositiontabShape : TabShapetabsClosable : boolusesScrollButtons : bool**

公共职能

	QTabWidget(QWidget **parent* = nullptr)
virtual	**~QTabWidget**()
int	**addTab**(QWidget **page*, const QString &*label*)
int	**addTab**(QWidget **page*, const QIcon &*icon*, const QString &*label*)
void	**clear**()

	QTabWidget(QWidget *parent = nullptr)
QWidget *	**cornerWidget**(Qt::Corner *corner* = Qt::TopRightCorner) const
int	**count**() const
int	**currentIndex**() const
QWidget *	**currentWidget**() const
bool	**documentMode**() const
Qt::TextElideMode	**elideMode**() const
QSize	**iconSize**() const
int	**indexOf**(const QWidget **w*) const
int	**insertTab**(int *index*, QWidget **page*, const QString &*label*)
int	**insertTab**(int *index*, QWidget **page*, const QIcon &*icon*, const QString &*label*)
bool	**isMovable**() const
bool	**isTabEnabled**(int *index*) const
bool	**isTabVisible**(int *index*) const
void	**removeTab**(int *index*)
void	**setCornerWidget**(QWidget **widget*, Qt::Corner *corner* = Qt::TopRightCorner)
void	**setDocumentMode**(bool *set*)
void	**setElideMode**(Qt::TextElideMode *mode*)
void	**setIconSize**(const QSize &*size*)
void	**setMovable**(bool *movable*)
void	**setTabBarAutoHide**(bool *enabled*)
void	**setTabEnabled**(int *index*, bool *enable*)
void	**setTabIcon**(int *index*, const QIcon &*icon*)
void	**setTabPosition**(QTabWidget::TabPosition *position*)
void	**setTabShape**(QTabWidget::TabShape *s*)
void	**setTabText**(int *index*, const QString &*label*)
void	**setTabToolTip**(int *index*, const QString &*tip*)
void	**setTabVisible**(int *index*, bool *visible*)
void	**setTabWhatsThis**(int *index*, const QString &*text*)
void	**setTabsClosable**(bool *closeable*)

	QTabWidget(QWidget *parent = nullptr)
void	**setUsesScrollButtons**(bool *useButtons*)
QTabBar *	**tabBar**() const
bool	**tabBarAutoHide**() const
QIcon	**tabIcon**(int *index*) const
QTabWidget::TabPosition	**tabPosition**() const
QTabWidget::TabShape	**tabShape**() const
QString	**tabText**(int *index*) const
QString	**tabToolTip**(int *index*) const
QString	**tabWhatsThis**(int *index*) const
bool	**tabsClosable**() const
bool	**usesScrollButtons**() const
QWidget *	**widget**(int *index*) const

重载的公共职能

virtual bool	**hasHeightForWidth() const override**
virtual int	**heightForWidth**(int *width*) const override
virtual QSize	**minimumSizeHint**() const override
virtual QSize	**sizeHint**() const override

公共槽

void	**setCurrentIndex(int *index*)**
void	**setCurrentWidget**(QWidget **widget*)

信号

void	**currentChanged(int *index*)**
void	**tabBarClicked**(int *index*)
void	**tabBarDoubleClicked**(int *index*)

void	**currentChanged**(**int** ***index***)
void	**tabCloseRequested**(int *index*)

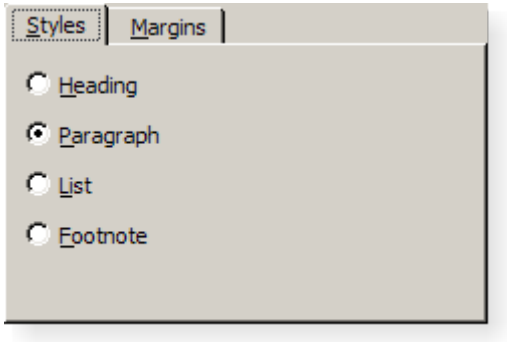
protected function

virtual void	**initStyleOption**(**QStyleOptionTabWidgetFrame** ***option***) **const**
void	**setTabBar**(QTabBar **tb*)
virtual void	**tabInserted**(int *index*)
virtual void	**tabRemoved**(int *index*)

重新实现protected function

virtual void	**changeEvent**(**QEvent** ***ev***) **override**
virtual bool	**event**(QEvent **ev*) override
virtual void	**keyPressEvent**(QKeyEvent **e*) override
virtual void	**paintEvent**(QPaintEvent **event*) override
virtual void	**resizeEvent**(QResizeEvent **e*) override
virtual void	**showEvent**(QShowEvent *) override

详细说明



选项卡小部件提供了一个选项卡栏（请参阅QTabBar）和一个“页面区域”，用于显示与每个选项卡相关的页面。默认情况下，选项卡栏显示在页面区域上方，但可以使用不同的配置（请参阅TabPosition）。每个选项卡都与不同的小部件（称为页面）相关联。页面区域仅显示当前页面；所有其他页面均被隐藏。用户可以通过单击其选项卡或按其 Alt+*字母*快捷方式（如果有）来显示不同的页面。

使用 QTabWidget 的正常方法是执行以下操作：

1. 创建一个 QTabWidget。

2. 创建一个QWidget对于选项卡对话框中的每个页面，但不要为其指定父窗口小部件。
3. 将子窗口小部件插入页面窗口小部件，使用布局将它们正常放置。
4. 称呼addTab（） 或者insertTab() 将页面小部件放入选项卡小部件中，为每个选项卡提供合适的标签和可选的键盘快捷键。

选项卡的位置由下式定义tabPosition，它们的形状由tabShape。

信号currentChanged当用户选择页面时会发出 ()。

当前页面索引可用作currentIndex()，当前页面小部件currentWidget()。您可以使用给定索引检索指向页面小部件的指针widget()，并且可以使用以下命令找到小部件的索引位置indexOf()。使用setCurrentWidget（） 或者setCurrentIndex() 显示特定页面。

您可以使用以下命令更改选项卡的文本和图标setTabText（） 或者setTabIcon()。可以使用以下命令删除选项卡及其关联页面removeTab()。

每个选项卡在任何给定时间都可以启用或禁用（请参阅setTabEnabled()）。如果启用了选项卡，则会正常绘制选项卡文本，并且用户可以选择该选项卡。如果禁用，选项卡将以不同的方式绘制，并且用户无法选择该选项卡。请注意，即使禁用了某个选项卡，该页面仍然可见，例如，如果所有选项卡碰巧都被禁用了。

选项卡小部件是分割复杂对话框的好方法。另一种方法是使用QStackedWidget您可以为此提供一些在页面之间导航的方法，例如QToolBar或一个QListWidget。

QTabWidget 中的大部分功能都是由QTabBar（在顶部，提供选项卡）和QStackedWidget（大部分区域，组织各个页面）。

也可以看看QTabBar,QStackedWidget,QToolBox， 和Tab Dialog Example。

会员类型文档

enum QTabWidget::TabPosition

这个枚举类型定义了哪里QTabWidget绘制选项卡行：

持续的	价值	描述
`QTabWidget::North`	`0`	选项卡绘制在页面上方。
`QTabWidget::South`	`1`	选项卡绘制在页面下方。
`QTabWidget::West`	`2`	选项卡绘制在页面左侧。
`QTabWidget::East`	`3`	选项卡绘制在页面右侧。

enum QTabWidget::TabShape

此枚举类型定义选项卡的形状：

持续的	价值	描述
`QTabWidget::Rounded`	`0`	选项卡采用圆形外观绘制。这是默认形状。

QTabWidget::Triangular	1	选项卡以三角形外观绘制。

属性文档

[read-only]count : const int

该属性保存选项卡栏中的选项卡数量

默认情况下，此属性包含值 0。

访问功能：

int	**count() const**

currentIndex : int

该属性保存当前标签页的索引位置

如果当前没有小部件，则当前索引为 -1。

默认情况下，此属性包含值 -1，因为小部件中最初没有选项卡。

访问功能：

int	**currentIndex() const**
void	**setCurrentIndex**(int *index*)

通知器信号：

void	**currentChanged(int *index*)**

documentMode : bool

该属性保存选项卡小部件是否以适合文档页面的模式呈现。这与 macOS 上的文档模式相同。

设置此属性后，不会呈现选项卡小部件框架。此模式对于显示文档类型页面非常有用，其中页面覆盖了大部分选项卡小部件区域。

访问功能：

bool	**documentMode() const**

bool	documentMode() const
void	**setDocumentMode**(bool *set*)

也可以看看elideMode,QTabBar::documentMode,QTabBar::usesScrollButtons， 和QStyle::SH_TabBar_PreferNoArrows。

elideMode : Qt::TextElideMode

如何删除标签栏中的文本

此属性控制当没有足够的空间来显示给定选项卡栏大小的项目时如何隐藏它们。

默认情况下，该值取决于样式。

访问功能：

Qt::TextElideMode	elideMode() const
void	**setElideMode**(Qt::TextElideMode *mode*)

也可以看看QTabBar::elideMode,usesScrollButtons， 和QStyle::SH_TabBar_ElideMode。

iconSize : QSize

该属性保存选项卡栏中图标的大小

默认值取决于样式。这是图标的最大尺寸。如果图标尺寸较小，则不会放大。

访问功能：

QSize	iconSize() const
void	**setIconSize**(const QSize &*size*)

也可以看看QTabBar::iconSize。

movable : bool

该属性保存用户是否可以在选项卡栏区域内移动选项卡。

默认情况下，该属性是`false`;

访问功能：

bool	isMovable() const
void	**setMovable**(bool *movable*)

tabBarAutoHide : bool

如果为 true，则当选项卡栏包含的选项卡少于 2 个时，选项卡栏将自动隐藏。

默认情况下，该属性为 false。

访问功能：

bool	**tabBarAutoHide() const**
void	**setTabBarAutoHide**(bool *enabled*)

也可以看看QWidget::visible。

tabPosition : TabPosition

此属性保存此选项卡小部件中选项卡的位置

该属性的可能值由TabPosition枚举。

默认情况下，该属性设置为North。

访问功能：

QTabWidget::TabPosition	**tabPosition() const**
void	**setTabPosition**(QTabWidget::TabPosition *position*)

也可以看看TabPosition。

tabShape : TabShape

此属性保存此选项卡小部件中选项卡的形状

该属性的可能值为QTabWidget::Rounded（默认）或QTabWidget::Triangular。

访问功能：

QTabWidget::TabShape	**tabShape() const**
void	**setTabShape**(QTabWidget::TabShape *s*)

也可以看看TabShape。

tabsClosable : bool

该属性保存关闭按钮是否自动添加到每个选项卡。

访问功能：

bool	**tabsClosable() const**
void	**setTabsClosable**(bool *closeable*)

也可以看看QTabBar::tabsClosable()。

usesScrollButtons : bool

此属性保存当选项卡栏有多个选项卡时是否应使用按钮来滚动选项卡。

当选项卡栏中的选项卡数量超出其大小时，选项卡栏可以选择扩大其大小或添加允许您滚动选项卡的按钮。

默认情况下，该值取决于样式。

访问功能：

bool	**usesScrollButtons() const**
void	**setUsesScrollButtons**(bool *useButtons*)

也可以看看elideMode,QTabBar::usesScrollButtons， 和QStyle::SH_TabBar_PreferNoArrows。

成员函数文档

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

构造一个带有父级的选项卡式小部件*parent*。

[virtual]QTabWidget::~QTabWidget()

销毁选项卡式小部件。

*int QTabWidget::addTab(QWidget *page, const QString &label)*

添加一个带有给定选项的选项卡*page*和*label*到选项卡小部件，并返回选项卡在选项卡栏中的索引。所有权*page*被传递到QTabWidget。

如果选项卡的*label*包含一个 & 符号，& 符号后面的字母用作选项卡的快捷方式，例如，如果标签是“Bro&wse”，则 Alt+W 成为将焦点移动到该选项卡的快捷方式。

注意：如果您在之后调用 addTab()show()，布局系统将尝试调整其小部件层次结构的变化，并可能导致闪烁。为了防止这种情况，您可以设置QWidget::updatesEnabled属性在更改之前为 false；请记住在更改完成后将该属性设置为 true，使小部件再次接收绘制事件。

也可以看看insertTab()。

*int QTabWidget::addTab(QWidget *page, const QIcon &icon, const QString &label)*

这是一个过载功能。

添加一个带有给定选项的选项卡*page*,*icon*， 和*label*到选项卡小部件，并返回选项卡在选项卡栏中的索引。所有权*page*被传递到QTabWidget。

该函数与 addTab() 相同，但多了一个*icon*。

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

重新实现：QWidget::changeEvent（QEvent *事件）。

void QTabWidget::clear()

删除所有页面，但不删除它们。调用该函数相当于调用removeTab() 直到选项卡小部件为空。

*QWidget *QTabWidget::cornerWidget(Qt::Corner corner = Qt::TopRightCorner) const*

返回显示在*corner*选项卡小部件的 或nullptr.

也可以看看setCornerWidget()。

[signal]void QTabWidget::currentChanged(int index)

每当当前页面索引发生变化时，就会发出此信号。参数为新的当前页面*index*位置，如果没有新位置，则为 -1（例如，如果QTabWidget）

注意：属性的通知程序信号currentIndex。

也可以看看currentWidget（） 和currentIndex。

*QWidget *QTabWidget::currentWidget() const*

返回指向选项卡对话框当前显示的页面的指针。选项卡对话框会尽力确保该值永远不会为 0（但如果您足够努力，它也可以是 0）。

也可以看看currentIndex（） 和setCurrentWidget()。

*[override virtual protected]bool QTabWidget::event(QEvent *ev)*

重新实现：QWidget::event（QEvent *事件）。

[override virtual]bool QTabWidget::hasHeightForWidth() const

重新实现：QWidget::hasHeightForWidth() const。

[override virtual]int QTabWidget::heightForWidth(int width) const

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

*int QTabWidget::indexOf(const QWidget *w) const*

返回widget占用的页面索引位置*w*，如果找不到小部件，则返回 -1。

*[virtual protected]void QTabWidget::initStyleOption(QStyleOptionTabWidgetFrame *option) const*

初始化*option*与此值QTabWidget。当子类需要时，此方法非常有用QStyleOptionTabWidgetFrame，但不想自己填写所有信息。

也可以看看QStyleOption::initFrom（） 和QTabBar::initStyleOption()。

*int QTabWidget::insertTab(int index, QWidget *page, const QString &label)*

插入一个带有给定选项的选项卡*label*和*page*进入指定的选项卡小部件*index*，并返回选项卡栏中插入的选项卡的索引。所有权*page*被传递到QTabWidget。

标签显示在选项卡中，并且外观可能会有所不同，具体取决于选项卡小部件的配置。

如果选项卡的*label*包含一个 & 符号，& 符号后面的字母用作选项卡的快捷方式，例如，如果标签是“Bro&wse”，则 Alt+W 成为将焦点移动到该选项卡的快捷方式。

如果*index*超出范围，仅附加选项卡。否则插入到指定位置。

如果QTabWidget在调用该函数之前为空，新页面成为当前页面。在小于或等于当前索引的索引处插入新选项卡将增加当前索引，但保留当前页面。

注意：如果您在之后调用 insertTab()show()，布局系统将尝试调整其小部件层次结构的变化，并可能导致闪烁。为了防止这种情况，您可以设置QWidget::updatesEnabled属性在更改之前为 false；请记住在更改完成后将该属性设置为 true，使小部件再次接收绘制事件。

也可以看看addTab()。

*int QTabWidget::insertTab(int index, QWidget *page, const QIcon &icon, const QString &label)*

这是一个过载功能。

插入一个带有给定选项的选项卡*label*,*page*， 和*icon*进入指定的选项卡小部件*index*，并返回选项卡栏中插入的选项卡的索引。所有权*page*被传递到QTabWidget。

该函数与 insertTab() 相同，但多了一个*icon*。

bool QTabWidget::isTabEnabled(int index) const

返回`true`页面是否位于位置*index*已启用；否则返回`false`.

也可以看看setTabEnabled（） 和QWidget::isEnabled()。

bool QTabWidget::isTabVisible(int index) const

如果页面位于位置，则返回 true*index*是可见的；否则返回 false。

也可以看看setTabVisible()。

*[override virtual protected]void QTabWidget::keyPressEvent(QKeyEvent *e)*

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

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

重新实现属性的访问函数：QWidget::minimumSizeHint。

返回选项卡小部件合适的最小尺寸。

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

重新实现：QWidget::paintEvent（QPaintEvent *事件）。

绘制选项卡小部件的选项卡栏以响应绘制*event*。

void QTabWidget::removeTab(int index)

删除位置处的标签*index*从这堆小部件中。页面小部件本身不会被删除。

也可以看看addTab（） 和insertTab()。

*[override virtual protected]void QTabWidget::resizeEvent(QResizeEvent *e)*

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

*void QTabWidget::setCornerWidget(QWidget *widget, Qt::Corner corner = Qt::TopRightCorner)*

设置给定的*widget*显示在指定的*corner*选项卡小部件的。小部件的几何形状是根据小部件的sizeHint() 和style()。

仅水平元素*corner*将会被使用。

通过`nullptr`显示角落里没有小部件。

任何先前设置的角落小部件都会被隐藏。

此处设置的所有小部件将在选项卡小部件被销毁时被删除，除非您在设置其他一些角小部件（或`nullptr`）后单独重新设置该小部件的父级。

注意：角落小部件设计用于North和South选项卡位置；已知其他方向无法正常工作。

也可以看看cornerWidget （） 和setTabPosition()。

*[slot]void QTabWidget::setCurrentWidget(QWidget *widget)*

使*widget*当前小部件。这*widget*使用的必须是此选项卡小部件中的页面。

也可以看看addTab(),setCurrentIndex （） ， 和currentWidget()。

*[protected]void QTabWidget::setTabBar(QTabBar *tb)*

替换对话框的QTabBar带标签栏的标题*tb*。请注意，必须在添加任何选项卡*之前*调用此函数，否则行为未定义。

也可以看看tabBar()。

void QTabWidget::setTabEnabled(int index, bool enable)

如果*enable*为 true，则页面位于位置*index*已启用；否则页面位于位置*index*被禁用。页面的选项卡被适当地重新绘制。

QTabWidget用途QWidget::setEnabled() 内部，而不是保留一个单独的标志。

请注意，即使禁用的选项卡/页面也可能是可见的。如果页面已经可见，QTabWidget不会隐藏它；如果所有页面都被禁用，QTabWidget将显示其中之一。

也可以看看isTabEnabled （） 和QWidget::setEnabled()。

void QTabWidget::setTabIcon(int index, const QIcon &icon)

设置*icon*对于位置处的选项卡*index*。

也可以看看tabIcon()。

void QTabWidget::setTabText(int index, const QString &label)

定义一个新的*label*对于位于位置的页面*index*的选项卡。

如果提供的文本包含与号字符 ('&')，则会自动为其创建快捷方式。“&”后面的字符将用作快捷键。如果文本没有定义快捷方式，则任何先前的快捷方式都将被覆盖或清除。请参阅QShortcut有关详细信息的文档（要显示实际的＆符号，请使用“&&”）。

也可以看看tabText()。

void QTabWidget::setTabToolTip(int index, const QString &tip)

设置页面的选项卡工具提示*index*到*tip*。

也可以看看tabToolTip()。

void QTabWidget::setTabVisible(int index, bool visible)

如果*visible*为 true，则页面位于位置*index*是可见的；否则页面位于位置*index*被隐藏。页面的选项卡被适当地重新绘制。

也可以看看isTabVisible()。

void QTabWidget::setTabWhatsThis(int index, const QString &text)

设置页面的“这是什么”帮助文本*index*到*text*。

也可以看看tabWhatsThis()。

*[override virtual protected]void QTabWidget::showEvent(QShowEvent *)*

重新实现：QWidget::showEvent（QShowEvent *事件）。

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

重新实现属性的访问函数：QWidget::sizeHint。

*QTabBar *QTabWidget::tabBar() const*

返回当前值QTabBar。

也可以看看setTabBar()。

[signal]void QTabWidget::tabBarClicked(int index)

当用户单击某个选项卡时会发出此信号*index*。

*index*指单击的选项卡，如果光标下没有选项卡，则为 -1。

[signal]void QTabWidget::tabBarDoubleClicked(int index)

当用户双击某个选项卡时会发出此信号*index*。

*index*是单击的选项卡的索引，如果光标下没有选项卡，则为 -1。

[signal]void QTabWidget::tabCloseRequested(int index)

单击选项卡上的关闭按钮时会发出此信号。这*index*是应该删除的索引。

也可以看看setTabsClosable()。

QIcon QTabWidget::tabIcon(int index) const

返回页面上位置处的选项卡的图标*index*。

也可以看看setTabIcon()。

[virtual protected]void QTabWidget::tabInserted(int index)

在位置添加或插入新选项卡后调用此虚拟处理程序*index*。

也可以看看tabRemoved()。

[virtual protected]void QTabWidget::tabRemoved(int index)

从位置删除选项卡后调用此虚拟处理程序*index*。

也可以看看tabInserted()。

QString QTabWidget::tabText(int index) const

返回页面上位置处的选项卡的标签文本*index*。

也可以看看setTabText()。

QString QTabWidget::tabToolTip(int index) const

返回位于位置的页面的选项卡工具提示*index*如果未设置工具提示，则为空字符串。

也可以看看setTabToolTip()。

QString QTabWidget::tabWhatsThis(int index) const

返回该位置页面的“这是什么”帮助文本*index*，如果未设置帮助文本，则为空字符串。

也可以看看setTabWhatsThis()。

QWidget *QTabWidget::widget(int index) const

返回索引位置的标签页*index*或者`nullptr`如果*index*超出范围。