

# QMessageBox Class

QMessageBox 类提供了一个模式对话框，用于通知用户或向用户询问问题并接收答案。[更多的...](#)

**Header:** `#include <QMessageBox>`

**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
- [已弃用的成员](#)
- QMessageBox 是[标准对话框](#)的一部分。

## 公共类型

**enum** [ButtonRole](#) { InvalidRole, AcceptRole, RejectRole, DestructiveRole, ActionRole, ..., ResetRole }

**enum** [Icon](#) { NoIcon, Question, Information, Warning, Critical }

**enum** [StandardButton](#) { Ok, Open, Save, Cancel, Close, ..., ButtonMask }

**flags** [StandardButtons](#)

## 特性

**detailedText** : QString**icon** :

**IconPixmap** : QPixmap**informativeText** : QString

**standardButtons** : StandardButtons**text** : QString**textFormat** : Qt::TextFormat**textInteractionFlags** : Qt::TextInteractionFlags

## 公共方法

**QMessageBox**(QWidget *\*parent* = nullptr)

**QMessageBox**(QMessageBox::Icon *icon*, const QString &*title*, const QString &*text*, QMessageBox::StandardButtons *buttons* = NoButton, QWidget *\*parent* =



QMessageBox(QWidget \*parent, Qt::Dialog | Qt::MSWindowsFixedSizeDialogHint)

virtual	<a href="#">~QMessageBox()</a>
void	<a href="#">addButton</a> (QAbstractButton *button, QMessageBox::ButtonRole role)
QPushButton *	<a href="#">addButton</a> (const QString &text, QMessageBox::ButtonRole role)
QPushButton *	<a href="#">addButton</a> (QMessageBox::StandardButton button)
QAbstractButton *	<a href="#">button</a> (QMessageBox::StandardButton which) const
QMessageBox::ButtonRole	<a href="#">buttonRole</a> (QAbstractButton *button) const
QList<QAbstractButton *>	<a href="#">buttons</a> () const
QCheckBox *	<a href="#">checkBox</a> () const
QAbstractButton *	<a href="#">clickedButton</a> () const
QPushButton *	<a href="#">defaultButton</a> () const
QString	<a href="#">detailedText</a> () const
QAbstractButton *	<a href="#">escapeButton</a> () const
QMessageBox::Icon	<a href="#">icon</a> () const
QPixmap	<a href="#">iconPixmap</a> () const
QString	<a href="#">informativeText</a> () const
void	<a href="#">open</a> (QObject receiver*, const char member*)
void	<a href="#">removeButton</a> (QAbstractButton *button)
void	<a href="#">setCheckBox</a> (QCheckBox *cb)
void	<a href="#">setDefaultButton</a> (QPushButton *button)
void	<a href="#">setDefaultButton</a> (QMessageBox::StandardButton button)
void	<a href="#">setDetailedText</a> (const QString &text)
void	<a href="#">setEscapeButton</a> (QAbstractButton *button)
void	<a href="#">setEscapeButton</a> (QMessageBox::StandardButton button)
void	<a href="#">setIcon</a> (QMessageBox::Icon)
void	<a href="#">setIconPixmap</a> (const QPixmap &pixmap)
void	<a href="#">setInformativeText</a> (const QString &text)
void	<a href="#">setStandardButtons</a> (QMessageBox::StandardButtons buttons)
void	<a href="#">setText</a> (const QString &text)
void	<a href="#">setTextFormat</a> (Qt::TextFormat format)

void	<b>QMessageBox</b> (QWidget <i>*parent</i> = nullptr) <b>setTextInteractionFlags</b> (Qt::TextInteractionFlags <i>flags</i> )
void	<b>setWindowModality</b> (Qt::WindowModality <i>windowModality</i> )
void	<b>setWindowTitle</b> (const QString & <i>title</i> )
QMessageBox::StandardButton	<b>standardButton</b> (QAbstractButton <i>*button</i> ) const
QMessageBox::StandardButtons	<b>standardButtons</b> () const
QString	<b>text</b> () const
Qt::TextFormat	<b>textFormat</b> () const
Qt::TextInteractionFlags	<b>textInteractionFlags</b> () const

## 公共槽

virtual int	<b>exec</b> () override
-------------	-------------------------

## 信号

void	<b>buttonClicked</b> (QAbstractButton <i>*button</i> )
------	--

## 静态公共成员

void	<b>about</b> (QWidget <i>*parent</i> , const QString & <i>title</i> , const QString & <i>text</i> )
void	<b>aboutQt</b> (QWidget <i>*parent</i> , const QString & <i>title</i> = QString())
QMessageBox::StandardButton	<b>critical</b> (QWidget <i>*parent</i> , const QString & <i>title</i> , const QString & <i>text</i> , QMessageBox::StandardButtons <i>buttons</i> = Ok, QMessageBox::StandardButton <i>defaultButton</i> = NoButton)
QMessageBox::StandardButton	<b>information</b> (QWidget <i>*parent</i> , const QString & <i>title</i> , const QString & <i>text</i> , QMessageBox::StandardButtons <i>buttons</i> = Ok, QMessageBox::StandardButton <i>defaultButton</i> = NoButton)

void	<b>about</b> (QWidget <i>*parent</i> , const QString & <i>title</i> , const QString & <i>text</i> )
QMessageBox::StandardButton	<b>question</b> (QWidget <i>*parent</i> , const QString & <i>title</i> , const QString & <i>text</i> , QMessageBox::StandardButtons <i>buttons</i> = StandardButtons(Yes   No), QMessageBox::StandardButton <i>defaultButton</i> = NoButton)
QMessageBox::StandardButton	<b>warning</b> (QWidget <i>*parent</i> , const QString & <i>title</i> , const QString & <i>text</i> , QMessageBox::StandardButtons <i>buttons</i> = Ok, QMessageBox::StandardButton <i>defaultButton</i> = NoButton)

## 重载的保护函数

virtual void	<b>changeEvent</b> (QEvent <i>*ev</i> ) override
virtual void	<b>closeEvent</b> (QCloseEvent <i>*e</i> ) override
virtual bool	<b>event</b> (QEvent <i>*e</i> ) override
virtual void	<b>keyPressEvent</b> (QKeyEvent <i>*e</i> ) override
virtual void	<b>resizeEvent</b> (QResizeEvent <i>*event</i> ) override
virtual void	<b>showEvent</b> (QShowEvent <i>*e</i> ) override

## 详细说明

消息框显示主要 **text** 提醒用户某种情况， **informative text** 进一步解释情况，以及可选的 **detailed text** 如果用户请求提供更多数据。

消息框还可以显示 **icon** 和 **standard buttons** 用于接受用户响应。

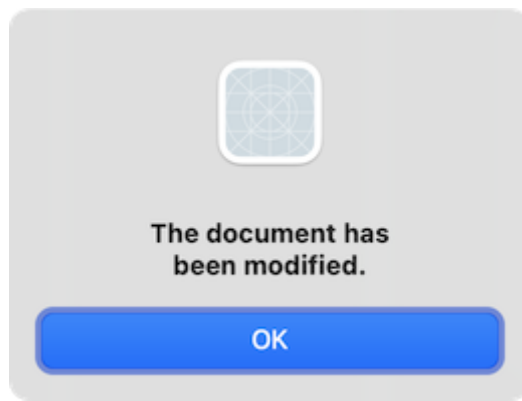
提供了两个使用 QMessageBox 的 API：基于属性的 API 和静态函数。调用静态函数之一是一种更简单的方法，但它不如使用基于属性的 API 灵活，而且结果的信息量也较少。建议使用基于属性的 API。

## 基于属性的 API

要使用基于属性的 API，请构造 QMessageBox 的实例，设置所需的属性，然后调用 **exec()** 显示消息。最简单的配置是只设置 **message text** 财产。

```
QMessageBox msgBox;
msgBox.setText("The document has been modified.");
msgBox.exec();
```

用户必须单击 **OK** 按钮以关闭消息框。GUI 的其余部分将被阻止，直到消息框消失。

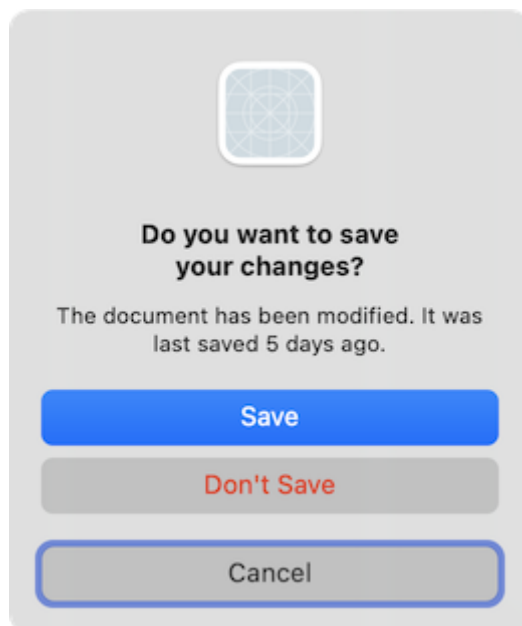


比仅仅提醒用户某个事件更好的方法是询问用户如何处理该事件。

设置 `standard buttons` 属性到您想要作为用户响应集的按钮集。这些按钮是通过组合以下值来指定的 `StandardButtons` 使用按位或运算符。按钮的显示顺序取决于平台。例如，在 Windows 上，**Save** 显示在左侧 **Cancel**，而在 macOS 上，顺序相反。将您的标准按钮之一标记为您的 `default button`。

这 `informative text` 属性可用于添加附加上下文以帮助用户选择适当的操作。

```
QMessageBox msgBox;
msgBox.setText("The document has been modified.");
msgBox.setInformativeText("Do you want to save your changes?");
msgBox.setStandardButtons(QMessageBox::Save | QMessageBox::Discard | QMessageBox::Cancel);
msgBox.setDefaultButton(QMessageBox::Save);
int ret = msgBox.exec();
```

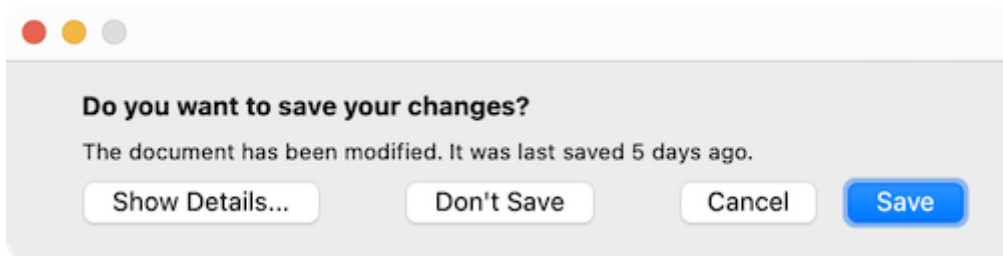


这 `exec()` 槽返回 `StandardButtons` 单击的按钮的值。

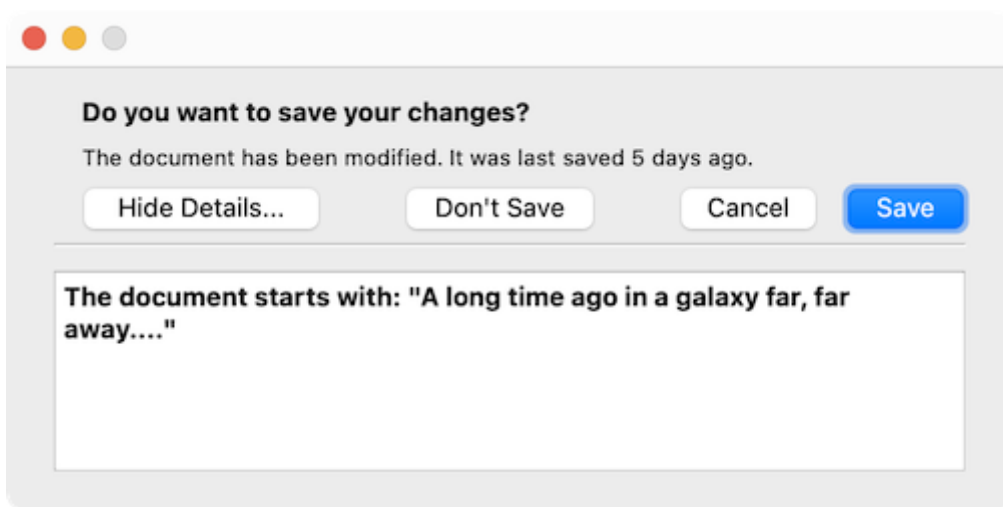
```
switch (ret) {
    case QMessageBox::Save:
        // Save was clicked
        break;
    case QMessageBox::Discard:
        // Don't Save was clicked
        break;
    case QMessageBox::Cancel:
        // Cancel was clicked
        break;
```

```
default:
    // should never be reached
    break;
}
```

为了给用户更多的信息来帮助他们选择合适的操作，设置 `detailed text` 财产。根据平台的不同 `detailed text`，可能需要用户单击 **Show Details...** 要显示的按钮。



单击 **Show Details...** 按钮显示详细文本。






## 富文本和文本格式属性

这 `detailed text` 属性始终被解释为纯文本。这 `main text` 和 `informative text` 属性可以是纯文本或富文本。这些字符串根据设置进行解释 `text format` 财产。默认设置是 `auto-text`。

请注意，对于某些包含 XML 元字符的纯文本字符串，自动文本 `rich text detection test` 可能会失败，导致您的纯文本字符串被错误地解释为富文本。在这些罕见的情况下，使用 `Qt::convertFromPlainText()` 将纯文本字符串转换为视觉上等效的富文本字符串，或设置 `text format` 属性明确地与 `setTextFormat()`。

## 严重级别以及图标和像素图属性

`QMessageBox` 支持四种预定义的消息严重性级别或消息类型，它们实际上仅在各自显示的预定义图标上有所不同。通过设置指定四种预定义消息类型之一 `icon` 财产归其中之一 `predefined icons`。以下规则为指导原则：

	Question	用于在正常操作期间提出问题。
	Information	用于报告有关正常操作的信息。
	Warning	用于报告非严重错误。



Question

用于在正常操作期间提出问题。



Critical

用于报告严重错误。

[Predefined icons](#)不是由 `QMessageBox` 定义的，而是由样式提供的。默认值为 [No Icon](#)。对于所有情况，消息框在其他方面都是相同的。使用标准图标时，请使用表中推荐的图标，或使用适合您平台的样式指南推荐的图标。如果没有一个标准图标适合您的消息框，您可以通过设置来使用自定义图标 [icon pixmap](#) 属性而不是设置 [icon](#) 财产。

总之，要设置图标，请使用 `setIcon()` 代表标准图标之一，或 `setIconPixmap()` 用于自定义图标。

## 静态函数 API

使用静态函数 API 构建消息框虽然方便，但不如使用基于属性的 API 灵活，因为静态函数签名缺少用于设置消息框的参数。 [informative text](#) 和 [detailed text](#) 特性。解决此问题的一种方法是使用参数 `title` 作为消息框主要文本，并将 `text` 参数作为消息框信息文本。因为这有一个明显的缺点，即消息框的可读性较差，所以平台指南不推荐它。 *Microsoft Windows 用户界面指南* 建议使用 [application name](#) 作为 [window's title](#)，这意味着如果除了主文本之外还有信息性文本，则必须将其连接到参数 `text`。

请注意，静态函数签名已更改其按钮参数，这些参数现在用于设置 [standard buttons](#) 和 [default button](#)。

静态函数可用于创建 [information\(\)](#), [question\(\)](#), [warning \(\)](#)，和 [critical\(\)](#) 消息框。

```
int ret = QMessageBox::warning(this, tr("My Application"),
                               tr("The document has been modified.\n"
                                   "Do you want to save your changes?"),
                               QMessageBox::Save | QMessageBox::Discard
                               | QMessageBox::Cancel,
                               QMessageBox::Save);
```

这 [Standard Dialogs](#) 示例展示了如何使用 `QMessageBox` 和其他内置 Qt 对话框。

## 高级用法

如果 [standard buttons](#) 对于您的消息框来说不够灵活，您可以使用 `addButton()` 重载需要一个文本和一个 [ButtonRole](#) 添加自定义按钮。这 [ButtonRole](#) `QMessageBox` 使用它来确定屏幕上按钮的顺序（根据平台的不同而有所不同）。您可以测试以下值 `clickedButton()` 调用后 `exec()`。例如，

```
QMessageBox msgBox;
QPushButton *connectButton = msgBox.addButton(tr("Connect"), QMessageBox::ActionRole);
QPushButton *abortButton = msgBox.addButton(QMessageBox::Abort);

msgBox.exec();

if (msgBox.clickedButton() == connectButton) {
    // connect
} else if (msgBox.clickedButton() == abortButton) {
    // abort
}
```

## 默认键和退出键

默认按钮（即，当Enter被按下）可以使用指定[setDefaultButton\(\)](#)。如果未指定默认按钮，QMessageBox 会尝试根据 [button roles](#)消息框中使用的按钮。

退出按钮（该按钮在以下情况下激活）Esc被按下）可以使用指定[setEscapeButton\(\)](#)。如果未指定转义按钮，QMessageBox 会尝试使用以下规则查找转义按钮：

- 1. 如果只有一个按钮，则该按钮是在以下情况下激活的按钮：Esc被按下。
- 2. 如果有一个Cancel按钮，它是激活时的按钮Esc被按下。
- 3. 如果恰好有一个按钮具有其中之一[the Reject role](#)或者[the No role](#)，它是激活时的按钮Esc被按下。

当使用这些规则无法确定退出按钮时，请按Esc没有影响。

也可以看看[QDialogButtonBox](#),[Standard Dialogs Example](#)和 Qt Widgets - 应用程序示例。

## 会员类型文档

### *enum QMessageBox::ButtonRole*

该枚举描述了可用于描述按钮框中按钮的角色。这些角色的组合就像用于描述其行为的不同方面的标志。

持续的	价值	描述
QMessageBox::InvalidRole	-1	该按钮无效。
QMessageBox::AcceptRole	0	单击该按钮会使对话框被接受（例如“确定”）。
QMessageBox::RejectRole	1	单击该按钮会导致对话框被拒绝（例如取消）。
QMessageBox::DestructiveRole	2	单击该按钮会导致破坏性更改（例如放弃更改）并关闭对话框。
QMessageBox::ActionRole	3	单击该按钮会导致对话框中的元素发生更改。
QMessageBox::HelpRole	4	可以单击该按钮来请求帮助。
QMessageBox::YesRole	5	该按钮是类似“是”的按钮。
QMessageBox::NoRole	6	该按钮是类似“否”的按钮。



持续的	价值	描述
<code>QMessageBox::ApplyRole</code>	8	该按钮应用当前更改。
<code>QMessageBox::ResetRole</code>	7	该按钮将对话框的字段重置为默认值。

也可以看看[StandardButton](#)。

## *enum QMessageBox::Icon*

该枚举具有以下值：

持续的	价值	描述
<code>QMessageBox::NoIcon</code>	0	消息框没有任何图标。
<code>QMessageBox::Question</code>	4	指示该消息正在询问问题的图标。
<code>QMessageBox::Information</code>	1	一个图标，表明该消息没有什么异常。
<code>QMessageBox::Warning</code>	2	一个图标，表明该消息是警告，但可以处理。
<code>QMessageBox::Critical</code>	3	指示该消息代表严重问题的图标。

## 枚举 *QMessageBox::StandardButton* 标志 *QMessageBox::StandardButtons*

这些枚举描述了标准按钮的标志。每个按钮都有一个定义的[ButtonRole](#)。

持续的	价值	描述
<code>QMessageBox::Ok</code>	0x00000400	定义的“确定”按钮 <a href="#">AcceptRole</a> 。
<code>QMessageBox::Open</code>	0x00002000	定义的“打开”按钮 <a href="#">AcceptRole</a> 。
<code>QMessageBox::Save</code>	0x00000800	定义的“保存”按钮 <a href="#">AcceptRole</a> 。
<code>QMessageBox::Cancel</code>	0x00400000	定义的“取消”按钮 <a href="#">RejectRole</a> 。
<code>QMessageBox::Close</code>	0x00200000	定义的“关闭”按钮 <a href="#">RejectRole</a> 。
<code>QMessageBox::Discard</code>	0x00800000	“放弃”或“不保存”按钮，具体取决于平台，由 <a href="#">DestructiveRole</a> 。
<code>QMessageBox::Apply</code>	0x02000000	定义的“应用”按钮 <a href="#">ApplyRole</a> 。
<code>QMessageBox::Reset</code>	0x04000000	一个“重置”按钮定义为 <a href="#">ResetRole</a> 。
<code>QMessageBox::RestoreDefaults</code>	0x08000000	定义的“恢复默认值”按钮 <a href="#">ResetRole</a> 。
<code>QMessageBox::Help</code>	0x01000000	定义的“帮助”按钮 <a href="#">HelpRole</a> 。

持续的	价值	描述
<code>QMessageBox::SaveAll</code>	<code>0x00001000</code>	定义的“全部保存”按钮 <a href="#">AcceptRole</a> 。
<code>QMessageBox::Yes</code>	<code>0x00004000</code>	“是”按钮定义为 <a href="#">YesRole</a> 。
<code>QMessageBox::YesToAll</code>	<code>0x00008000</code>	“全部同意”按钮定义为 <a href="#">YesRole</a> 。
<code>QMessageBox::No</code>	<code>0x00010000</code>	“否”按钮定义为 <a href="#">NoRole</a> 。
<code>QMessageBox::NoToAll</code>	<code>0x00020000</code>	“全部拒绝”按钮定义为 <a href="#">NoRole</a> 。
<code>QMessageBox::Abort</code>	<code>0x00040000</code>	“中止”按钮定义为 <a href="#">RejectRole</a> 。
<code>QMessageBox::Retry</code>	<code>0x00080000</code>	定义的“重试”按钮 <a href="#">AcceptRole</a> 。
<code>QMessageBox::Ignore</code>	<code>0x00100000</code>	定义的“忽略”按钮 <a href="#">AcceptRole</a> 。
<code>QMessageBox::NoButton</code>	<code>0x00000000</code>	无效按钮。

以下值已过时：

持续的	价值	描述
<code>QMessageBox::YesAll</code>	<code>YesToAll</code>	请改用 <code>YesToAll</code> 。
<code>QMessageBox::NoAll</code>	<code>NoToAll</code>	请改用 <code>NoToAll</code> 。
<code>QMessageBox::Default</code>	<code>0x00000100</code>	使用 <code>defaultButton</code> 以下参数 <a href="#">information()</a> , <a href="#">warning()</a> 等代替，或调用 <a href="#">setDefaultButton()</a> 。
<code>QMessageBox::Escape</code>	<code>0x00000200</code>	称呼 <a href="#">setEscapeButton</a> () 反而。
<code>QMessageBox::FlagMask</code>	<code>0x00000300</code>	
<code>QMessageBox::ButtonMask</code>	<code>~FlagMask</code>	

`StandardButtons` 类型是[QFlags](#) 的类型定义。它存储 `StandardButton` 值的 OR 组合。

也可以看看[ButtonRole](#)和[standardButtons](#)。

# 财产文件

*detailedText* : [QString](#)

该属性保存要在详细信息区域中显示的文本。

该文本将被解释为纯文本。

默认情况下，该属性包含一个空字符串。

访问功能：

<code>QString</code>	<code>detailedText() const</code>
----------------------	-----------------------------------

QString	detailedText() const
void	setDetailedText(const QString &text)

也可以看看[QMessageBox::text](#)和[QMessageBox::informativeText](#)。

## icon : Icon

该属性保存消息框的图标

消息框的图标可以用以下值之一指定：

- [QMessageBox::NoIcon](#)
- [QMessageBox::Question](#)
- [QMessageBox::Information](#)
- [QMessageBox::Warning](#)
- [QMessageBox::Critical](#)

默认为[QMessageBox::NoIcon](#)。

用于显示实际图标的像素图取决于当前[GUI style](#)。您还可以通过设置图标来设置自定义像素图[icon pixmap](#)财产。

访问功能：

QMessageBox::Icon	icon() const
void	setIcon(QMessageBox::Icon)

也可以看看[iconPixmap](#)。

## iconPixmap : QPixmap

该属性保存当前图标

消息框当前使用的图标。请注意，通常很难绘制一张看起来适合所有 GUI 风格的像素图；您可能想为每个平台提供不同的像素图。

默认情况下，该属性是未定义的。

访问功能：

QPixmap	iconPixmap() const
void	setIconPixmap(const QPixmap &pixmap)

也可以看看[icon](#)。

## *informativeText : QString*

该属性包含信息文本，为消息提供更完整的描述

信息性文本可用于扩展[text\(\)](#) 向用户提供更多信息，例如描述情况的后果，或建议替代解决方案。

默认情况下，该属性包含一个空字符串。

### 访问功能：

QString	informativeText() const
void	setInformativeText(const QString &text)

也可以看看[QMessageBox::text](#)和[QMessageBox::detailedText](#)。

## *standardButtons : StandardButtons*

消息框中标准按钮的集合

该属性控制消息框使用哪些标准按钮。

默认情况下，此属性不包含标准按钮。

### 访问功能：

QMessageBox::StandardButtons	standardButtons() const
void	setStandardButtons(QMessageBox::StandardButtons buttons)

也可以看看[addButton\(\)](#)。

## *text : QString*

该属性保存要显示的消息框文本。

文本应该是描述情况的简短句子或短语，最好是中立的陈述或号召性用语问题。

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（[QMessageBox::textFormat](#)）。默认设置是[Qt::AutoText](#)，即消息框将尝试自动检测文本的格式。

该属性的默认值为空字符串。

### 访问功能：

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

也可以看看[textFormat](#)、[QMessageBox::informativeText](#)，和[QMessageBox::detailedText](#)。

## *textFormat : Qt::TextFormat*

该属性保存消息框显示的文本格式

消息框当前使用的文本格式。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 format)

也可以看看[setText\(\)](#)。

## *textInteractionFlags : Qt::TextInteractionFlags*

指定消息框的标签应如何与用户输入交互。

默认值取决于样式。

**访问功能：**

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

也可以看看[QStyle::SH\\_MessageBox\\_TextInteractionFlags](#)。

## 成员函数文档

*[explicit] QMessageBox::QMessageBox(QWidget \*parent = nullptr)*

构造一个[application modal](#)没有文本和按钮的消息框。*parent*被传递到[QDialog](#)构造函数。

窗口模式可以通过重写[setWindowModality\(\)](#) 调用之前[show\(\)](#)。

**注意：**使用[open \(\)](#) 或者[exec\(\)](#) 显示的消息框影响窗口模态。请参阅每个功能的详细文档以获取更多信息。

在 macOS 上，如果您希望消息框显示为[Qt::Sheet](#)其*parent*，设置消息框的[window modality](#)到[Qt::WindowModal](#)或使用[open\(\)](#)。否则，消息框将是一个标准对话框。

也可以看看[setWindowTitle\(\)](#),[setText\(\)](#),[setIcon\(\)](#),[setStandardButtons \(\)](#) , 和[setWindowModality\(\)](#)。

```
QMessageBox::QMessageBox(QMessageBox::Icon icon, const QString
&title, const QString &text, QMessageBox::StandardButtons buttons =
NoButton, QWidget *parent = nullptr, Qt::WindowFlags f = Qt::Dialog |
Qt::MSWindowsFixedSizeDialogHint)
```

构造一个 `application modal` 带有给定信息的信息框 `icon, title, text`，和标准 `buttons`。可以随时使用添加标准或自定义按钮 `addButton()`。这 `parent` 和 `f` 参数被传递给 `QDialog` 构造函数。

窗口模式可以通过重写 `setWindowModality()` 调用之前 `show()`。

**注意：**使用 `open ()` 或者 `exec()` 显示的消息框影响窗口模态。请参阅每个功能的详细文档以获取更多信息。

在 macOS 上，如果 `parent` 不是 `nullptr`，并且您希望消息框显示为 `Qt::Sheet` 该父级的，设置消息框的 `window modality` 到 `Qt::WindowModal`（默认）。否则，消息框将是一个标准对话框。

也可以看看 `setWindowTitle()`, `setText()`, `setIcon()`, `setStandardButtons ()`，和 `setWindowModality()`。

```
[virtual] QMessageBox::~~QMessageBox()
```

销毁消息框。

```
[static]void QMessageBox::about(QWidget *parent, const QString &title,
const QString &text)
```

显示一个带有标题的简单“关于”框 `title` 和文字 `text`。关于框的父级是 `parent`。

`about()` 在四个位置寻找合适的图标：

1. 它更喜欢 `parent->icon()` 如果存在的话。
2. 如果没有，它会尝试包含的顶级小部件 `parent`。
3. 如果失败，它会尝试 `active window`。
4. 作为最后的手段，它使用信息图标。

关于框有一个标记为“确定”的按钮。在 macOS 上，“关于”框以无模式窗口的形式弹出；在其他平台上，目前是应用程序模式。

也可以看看 `QWidget::windowIcon ()` 和 `QApplication::activeWindow()`。

*[static]void QMessageBox::aboutQt(QWidget \*parent, const QString &title = QString())*

显示一个关于 Qt 的简单消息框，其中包含给定的内容`title`并以`parent`（如果`parent`不是`nullptr`）。该消息包括应用程序正在使用的 Qt 版本号。

这对于包含在Help应用程序的菜单，如图所示Menus例子。

QApplication作为插槽提供此功能。

在 macOS 上，“关于”框以无模式窗口的形式弹出；在其他平台上，目前是应用程序模式。

也可以看看QApplication::aboutQt()。

*void QMessageBox::addButton(QAbstractButton \*button, QMessageBox::ButtonRole role)*

添加给定的`button`到消息框指定`role`。

也可以看看removeButton(),button () , 和setStandardButtons()。

*QPushButton QMessageBox::addButton(const QString &text\*, QMessageBox::ButtonRole role)*

这是一个过载功能。

使用给定的值创建一个按钮`text`，将其添加到指定的消息框中`role`，并返回它。

*QPushButton \*QMessageBox::addButton(QMessageBox::StandardButton button)*

这是一个过载功能。

添加一个标准`button`如果这样做有效，则发送到消息框，并返回按钮。

也可以看看setStandardButtons()。

*[QAbstractButton](#) \*QMessageBox::button([QMessageBox::StandardButton](#) which) const*

返回对应于标准按钮的指针which, 或者nullptr如果此消息框中不存在标准按钮。

也可以看看[standardButtons](#)和[standardButton\(\)](#)。

*[signal]void QMessageBox::buttonClicked([QAbstractButton](#) \*button)*

每当单击内部按钮时就会发出此信号QMessageBox。被点击的按钮返回了button。

*[QMessageBox::ButtonRole](#) QMessageBox::buttonRole([QAbstractButton](#) \*button) const*

返回指定按钮的角色button。该函数返回InvalidRole如果button已nullptr添加或尚未添加到消息框。

也可以看看[buttons](#) () 和[addButton\(\)](#)。

*[QList](#)<[QAbstractButton](#) \*> QMessageBox::buttons() const*

返回已添加到消息框的所有按钮的列表。

也可以看看[buttonRole\(\)](#),[addButton](#) () , 和[removeButton\(\)](#)。

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

重新实现: [QWidget::changeEvent](#) (QEvent \*事件) 。

*[QCheckBox](#) \*QMessageBox::checkBox() const*

返回对话框中显示的复选框。这是nullptr如果未设置复选框的情况。

也可以看看[setCheckBox\(\)](#)。



## *[QAbstractButton](#) \*[QMessageBox::clickedButton\(\)](#) const*

返回用户单击的按钮，或者nullptr如果用户单击Esc钥匙和没有escape button已设置。

如果exec() 尚未被调用，返回 nullptr。

例子：

```
QMessageBox messageBox(this);
QAbstractButton *disconnectButton =
    messageBox.addButton(tr("Disconnect"), QMessageBox::ActionRole);
...
messageBox.exec();
if (messageBox.clickedButton() == disconnectButton) {
    ...
}
```

也可以看看[standardButton \(\)](#) 和[button\(\)](#)。

## *[override virtual protected]void [QMessageBox::closeEvent\(QCloseEvent \\*e\)](#)*

重新实现：[QDialog::closeEvent\(QCloseEvent \\*e\)](#)。

## *[static][QMessageBox::StandardButton](#) [QMessageBox::critical\(QWidget](#) *\*parent, const [QString](#) &title, const [QString](#) &text,* *[QMessageBox::StandardButtons](#) buttons = Ok,* *[QMessageBox::StandardButton](#) defaultButton = NoButton)**

使用给定的信息打开关键消息框title和text在指定的前面parent小部件。

标准buttons被添加到消息框中。defaultButton指定时使用的按钮Enter被按下。defaultButton必须引用在中给出的按钮buttons。如果defaultButton是QMessageBox::NoButton,QMessageBox自动选择合适的默认值。

返回被单击的标准按钮的标识。如果Esc相反，被按下escape button被返回。

消息框是一个application modal对话框。

**警告：**请勿删除parent在对话框执行期间。如果您想这样做，您应该使用其中之一自己创建对话框[QMessageBox](#)构造函数。

也可以看看[question\(\)](#),[warning \(\)](#) , 和[information\(\)](#)。

## *QPushButton \*QMessageBox::defaultButton() const*

返回应该是消息框的按钮 `default button`。如果未设置默认按钮，则返回 `nullptr`。

也可以看看 `setDefaultButton()`, `addButton ()` , 和 `QPushButton::setDefault()`。

## *QAbstractButton \*QMessageBox::escapeButton() const*

返回按下 `escape` 时激活的按钮。

默认情况下， `QMessageBox` 尝试自动检测退出按钮，如下所示：

1. 如果只有一个按钮，则将其设为退出按钮。
2. 如果有一个 `Cancel` 按钮，它被做成了退出按钮。
3. 仅在 macOS 上，如果只有一个具有该角色的按钮 `QMessageBox::RejectRole`，它被做成了退出按钮。

当无法自动检测到退出按钮时，按 `Esc` 没有影响。

也可以看看 `setEscapeButton ()` 和 `addButton()`。

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

重新实现： `QWidget::event (QEvent *事件)` 。

## *[override virtual slot] int QMessageBox::exec()*

重新实现： `QDialog::exec()`。

将消息框显示为 `modal dialog`，阻塞直到用户关闭它。

当使用 `QMessageBox` 对于标准按钮，此函数返回一个 `StandardButton` 指示单击的标准按钮的值。使用时 `QMessageBox` 对于自定义按钮，此函数返回一个不透明的值；使用 `clickedButton()` 来确定单击了哪个按钮。

注： `result()` 函数也返回 `StandardButton` 值而不是 `QDialog::DialogCode`。

在通过单击按钮或使用窗口系统提供的机制关闭对话框之前，用户无法与同一应用程序中的任何其他窗口进行交互。

也可以看看 `show ()` 和 `result()`。

```
[static] QMessageBox::StandardButton
QMessageBox::information(QWidget *parent, const QString &title, const
    QString &text, QMessageBox::StandardButtons buttons = Ok,
    QMessageBox::StandardButton defaultButton = NoButton)
```

打开带有给定信息的信息消息框`title`和`text`在指定的前面`parent`小部件。

标准`buttons`被添加到消息框中。`defaultButton`指定时使用的按钮`Enter`被按下。`defaultButton`必须引用在中给出的按钮`buttons`。如果`defaultButton`是`QMessageBox::NoButton`,`QMessageBox`自动选择合适的默认值。

返回被单击的标准按钮的标识。如果`Esc`相反，被按下`escape button`被返回。

消息框是一个`application modal`对话框。

**警告：**请勿删除`parent`在对话框执行期间。如果您想这样做，您应该使用其中之一自己创建对话框`QMessageBox`构造函数。

也可以看看`question()`,`warning ()` , 和`critical()`。

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

重新实现：`QDialog::keyPressEvent (QKeyEvent *e)` 。

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

打开对话框并连接其`finished ()` 或者`buttonClicked()` 向由指定的槽发出信号`receiver`和`member`。如果槽位在`member`有一个指向连接的第一个参数的指针`buttonClicked()`，否则连接为`finished()`。

当对话框关闭时，信号将从插槽断开。

```
[static] QMessageBox::StandardButton QMessageBox::question(QWidget
    *parent, const QString &title, const QString &text,
    QMessageBox::StandardButtons buttons = StandardButtons(Yes | No),
    QMessageBox::StandardButton defaultButton = NoButton)
```

打开一个问题消息框，其中包含给定的内容`title`和`text`在指定的前面`parent`小部件。

标准`buttons`被添加到消息框中。`defaultButton`指定时使用的按钮`Enter`被按下。`defaultButton`必须引用在中给出的按钮`buttons`。如果`defaultButton`是`QMessageBox::NoButton`,`QMessageBox`自动选择合适的默认值。

返回被单击的标准按钮的标识。如果`Esc`相反，被按下`escape button`被返回。

消息框是一个`application modal`对话框。

**警告：**请勿删除`parent`在对话框执行期间。如果您想这样做，您应该使用其中之一自己创建对话框[QMessageBox](#)构造函数。

也可以看看[information\(\)](#),[warning \(\)](#) , 和[critical\(\)](#)。

```
void QMessageBox::removeButton(QAbstractButton *button)
```

删除`button`从按钮框中删除它。

也可以看看[addButton \(\)](#) 和[setStandardButtons\(\)](#)。

```
[override virtual protected]void  
QMessageBox::resizeEvent(QResizeEvent *event)
```

重新实现: [QDialog::resizeEvent](#) ([QResizeEvent](#) \*) 。

```
void QMessageBox::setCheckBox(QCheckBox *cb)
```

设置复选框`cb`在消息对话框上。消息框拥有复选框的所有权。论据`cb`可以是`nullptr`从消息框中删除现有的复选框。

也可以看看[checkBox\(\)](#)。

```
void QMessageBox::setDefaultButton(QPushButton *button)
```

设置消息框的[default button](#)到`button`。

也可以看看[defaultButton\(\)](#),[addButton \(\)](#) , 和[QPushButton::setDefault\(\)](#)。

```
void QMessageBox::setDefaultButton(QMessageBox::StandardButton  
button)
```

设置消息框的[default button](#)到`button`。

也可以看看[addButton \(\)](#) 和[QPushButton::setDefault\(\)](#)。

*void QMessageBox::setEscapeButton([QAbstractButton](#) \*button)*

设置当按下按钮时激活的按钮**Escape**键被按下`button`。

也可以看看[escapeButton\(\)](#),[addButton \(\)](#) , 和[clickedButton\(\)](#)。

*void QMessageBox::setEscapeButton([QMessageBox::StandardButton](#) button)*

设置当按下按钮时激活的按钮**Escape**键被按下`button`。

也可以看看[addButton \(\)](#) 和[clickedButton\(\)](#)。

*void QMessageBox::setWindowModality([Qt::WindowModality](#)  
windowModality)*

这个函数有阴影[QWidget::setWindowModality\(\)](#)。

将消息框的模式设置为`windowModality`。

在 macOS 上，如果模式设置为[Qt::WindowModal](#)并且消息框有一个父级，那么消息框将是[Qt::Sheet](#)，否则消息框将是一个标准对话框。

*void QMessageBox::setWindowTitle(const [QString](#) &title)*

这个函数有阴影[QWidget::setWindowTitle\(\)](#)。

将消息框的标题设置为`title`。在 macOS 上，窗口标题将被忽略（根据 macOS 指南的要求）。

*[override virtual protected]void  
QMessageBox::showEvent([QShowEvent](#) \*e)*

重新实现：[QDialog::showEvent](#) ([QShowEvent](#) \*事件) 。

## *QMessageBox::StandardButton*

*QMessageBox::standardButton(QAbstractButton \*button) const*

返回与给定对应的标准按钮枚举值`button`，或者`NoButton`如果给定`button`不是标准按钮。

也可以看看[button \(\)](#) 和[standardButtons\(\)](#)。

```
[static]QMessageBox::StandardButton QMessageBox::warning(QWidget
    *parent, const QString &title, const QString &text,
    QMessageBox::StandardButtons buttons = Ok,
    QMessageBox::StandardButton defaultButton = NoButton)
```

打开一个警告消息框，其中包含给定的内容`title`和`text`在指定的前面`parent`小部件。

标准`buttons`被添加到消息框中。`defaultButton`指定时使用的按钮`Enter`被按下。`defaultButton`必须引用在中给出的按钮`buttons`。如果`defaultButton`是`QMessageBox::NoButton`，`QMessageBox`自动选择合适的默认值。

返回被单击的标准按钮的标识。如果`Esc`相反，被按下`escape button`被返回。

消息框是一个[application modal](#)对话框。

**警告：**请勿删除`parent`在对话框执行期间。如果您想这样做，您应该使用其中之一自己创建对话框[QMessageBox](#)构造函数。

也可以看看[question\(\)](#)，[information \(\)](#)，和[critical\(\)](#)。