QFileDialog Class

QFileDialog 类提供了一个允许用户选择文件或目录的对话框。更多的...

Header:	**#include < QFileDialog >**
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
- QFileDialog 是标准对话框的一部分。

公共类型

enum	**AcceptMode { AcceptOpen, AcceptSave }**
enum	**DialogLabel** { LookIn, FileName, FileType, Accept, Reject }
enum	**FileMode** { AnyFile, ExistingFile, Directory, ExistingFiles }
enum	**Option** { ShowDirsOnly, DontResolveSymlinks, DontConfirmOverwrite, DontUseNativeDialog, ReadOnly, …, DontUseCustomDirectoryIcons }
flags	**Options**
enum	**ViewMode** { Detail, List }

特性

acceptMode : AcceptModedefaultSuffix : QStringfileMode : FileMode	**options : Optionssupported Schemes : QStringListviewMode : ViewMode**

公共职能

QFileDialog(QWidget **parent*, Qt::WindowFlags *flags*)

	QFileDialog(**QWidget** ***parent***, **Qt::WindowFlags** ***flags***)
	QFileDialog(QWidget **parent* = nullptr, const QString &*caption* = QString(), const QString &*directory* = QString(), const QString &*filter* = QString())
virtual	**~QFileDialog**()
QFileDialog::AcceptMode	**acceptMode**() const
QString	**defaultSuffix**() const
QDir	**directory**() const
QUrl	**directoryUrl**() const
QFileDialog::FileMode	**fileMode**() const
QDir::Filters	**filter**() const
QStringList	**history**() const
QAbstractFileIconProvider *	**iconProvider**() const
QAbstractItemDelegate *	**itemDelegate**() const
QString	**labelText**(QFileDialog::DialogLabel *label*) const
QStringList	**mimeTypeFilters**() const
QStringList	**nameFilters**() const
void	**open**(QObject **receiver*, const char** member*)
QFileDialog::Options	**options**() const
QAbstractProxyModel *	**proxyModel**() const
bool	**restoreState**(const QByteArray &*state*)
QByteArray	**saveState**() const
void	**selectFile**(const QString &*filename*)
void	**selectMimeTypeFilter**(const QString &*filter*)
void	**selectNameFilter**(const QString &*filter*)
void	**selectUrl**(const QUrl &*url*)
QStringList	**selectedFiles**() const
QString	**selectedMimeTypeFilter**() const
QString	**selectedNameFilter**() const
QList< QUrl >	**selectedUrls**() const
void	**setAcceptMode**(QFileDialog::AcceptMode *mode*)

	QFileDialog(QWidget **parent*, Qt::WindowFlags *flags*)
void	**setDefaultSuffix**(const QString &*suffix*)
void	**setDirectory**(const QString &*directory*)
void	**setDirectory**(const QDir &*directory*)
void	**setDirectoryUrl**(const QUrl &*directory*)
void	**setFileMode**(QFileDialog::FileMode *mode*)
void	**setFilter**(QDir::Filters *filters*)
void	**setHistory**(const QStringList &*paths*)
void	**setIconProvider**(QAbstractFileIconProvider **provider*)
void	**setItemDelegate**(QAbstractItemDelegate **delegate*)
void	**setLabelText**(QFileDialog::DialogLabel *label*, const QString &*text*)
void	**setMimeTypeFilters**(const QStringList &*filters*)
void	**setNameFilter**(const QString &*filter*)
void	**setNameFilters**(const QStringList &*filters*)
void	**setOption**(QFileDialog::Option *option*, bool *on* = true)
void	**setOptions**(QFileDialog::Options *options*)
void	**setProxyModel**(QAbstractProxyModel **proxyModel*)
void	**setSidebarUrls**(const QList< QUrl > &*urls*)
void	**setSupportedSchemes**(const QStringList &*schemes*)
void	**setViewMode**(QFileDialog::ViewMode *mode*)
QList< QUrl >	**sidebarUrls**() const
QStringList	**supportedSchemes**() const
bool	**testOption**(QFileDialog::Option *option*) const
QFileDialog::ViewMode	**viewMode**() const

重载的公共方法

virtual void	**setVisible**(**bool** ***visible***) **override**

信号

void	**currentChanged(const QString &*path*)**
void	currentUrlChanged(const QUrl &*url*)
void	directoryEntered(const QString &*directory*)
void	directoryUrlEntered(const QUrl &*directory*)
void	fileSelected(const QString &*file*)
void	filesSelected(const QStringList &*selected*)
void	filterSelected(const QString &*filter*)
void	urlSelected(const QUrl &*url*)
void	urlsSelected(const QList< QUrl > &*urls*)

静态公共成员

QString	**getExistingDirectory(QWidget **parent* = nullptr, const QString &*caption* = QString(), const QString &*dir* = QString(), QFileDialog::Options *options* = ShowDirsOnly)**
QUrl	getExistingDirectoryUrl(QWidget **parent* = nullptr, const QString &*caption* = QString(), const QUrl &*dir* = QUrl(), QFileDialog::Options *options* = ShowDirsOnly, const QStringList &*supportedSchemes* = QStringList())
void	getOpenFileContent(const QString &*nameFilter*, const std::function<void (const QString &, const QByteArray &)> &*fileOpenCompleted*)
QString	getOpenFileName(QWidget **parent* = nullptr, const QString &*caption* = QString(), const QString &*dir* = QString(), const QString &*filter* = QString(), QString** selectedFilter* = nullptr, QFileDialog::Options *options* = Options())
QStringList	getOpenFileNames(QWidget **parent* = nullptr, const QString &*caption* = QString(), const QString &*dir* = QString(), const QString &*filter* = QString(), QString** selectedFilter* = nullptr, QFileDialog::Options *options* = Options())
QUrl	getOpenFileUrl(QWidget **parent* = nullptr, const QString &*caption* = QString(), const QUrl &*dir* = QUrl(), const QString &*filter* = QString(), QString** selectedFilter* = nullptr, QFileDialog::Options *options* = Options(), const QStringList &*supportedSchemes* = QStringList())
QList< QUrl >	getOpenFileUrls(QWidget **parent* = nullptr, const QString &*caption* = QString(), const QUrl &*dir* = QUrl(), const QString &*filter* = QString(), QString** selectedFilter* = nullptr, QFileDialog::Options *options* = Options(), const QStringList &*supportedSchemes* = QStringList())
QString	getSaveFileName(QWidget **parent* = nullptr, const QString &*caption* = QString(), const QString &*dir* = QString(), const QString &*filter* = QString(), QString** selectedFilter* = nullptr, QFileDialog::Options *options* = Options())
QUrl	getSaveFileUrl(QWidget **parent* = nullptr, const QString &*caption* = QString(), const QUrl &*dir* = QUrl(), const QString &*filter* = QString(), QString** selectedFilter* = nullptr, QFileDialog::Options *options* = Options(), const QStringList &*supportedSchemes* = QStringList())

QString	**getExistingDirectory(QWidget **parent* = nullptr, const QString &*caption* = QString(), const QString &*dir* = QString(), QFileDialog::Options *options* = ShowDirsOnly)**
void	**saveFileContent**(const QByteArray &*fileContent*, const QString &*fileNameHint* = QString())

重新实现受保护的功能

virtual void	**accept() override**
virtual void	**changeEvent**(QEvent **e*) override
virtual void	**done**(int *result*) override

详细说明

QFileDialog 类使用户能够遍历文件系统以选择一个或多个文件或目录。

创建 QFileDialog 最简单的方法是使用静态函数。

```
fileName = QFileDialog::getOpenFileName(this,
    tr("Open Image"), "/home/jana", tr("Image Files (*.png *.jpg *.bmp)"));
```

在上面的示例中，使用静态函数创建了一个模态 QFileDialog。该对话框最初显示“/home/jana”目录的内容，并显示与字符串“Image Files (*.png *.jpg *.bmp)”*中给定模式匹配的文件。文件对话框的父级设置为*this*，窗口标题设置为“打开图像”。

*如果您想使用多个过滤器，请用两个*分号分隔每个过滤器。例如：

```
"Images (*.png *.xpm *.jpg);;Text files (*.txt);;XML files (*.xml)"
```

您可以在不使用静态函数的情况下创建自己的 QFileDialog。通过调用setFileMode()，您可以指定用户必须在对话框中选择的内容：

```
QFileDialog dialog(this);
dialog.setFileMode(QFileDialog::AnyFile);
```

在上面的示例中，文件对话框的模式设置为AnyFile，这意味着用户可以选择任何文件，甚至可以指定不存在的文件。此模式对于创建“另存为”文件对话框非常有用。使用ExistingFile如果用户必须选择现有文件，或者Directory如果只能选择一个目录。请参阅QFileDialog::FileMode枚举以获得模式的完整列表。

这fileMode属性包含对话框的操作模式；这表明用户期望选择什么类型的对象。使用setNameFilter() 设置对话框的文件过滤器。例如：

```
dialog.setNameFilter(tr("Images (*.png *.xpm *.jpg)"));
```

在上面的示例中，过滤器设置为`"Images (*.png *.xpm *.jpg)"`，这意味着只有扩展名为`png`、`xpm`、 或`jpg`才会显示在 QFileDialog 中。您可以使用以下方法应用多个过滤器setNameFilters()。使用selectNameFilter() 选择您指定的过滤器之一作为文件对话框的默认过滤器。

文件对话框有两种查看模式：List和Detail。List将当前目录的内容显示为文件和目录名称的列表。Detail还显示文件和目录名称的列表，但在每个名称旁边提供附加信息，例如文件大小和修改日期。设置模式setViewMode():

```
dialog.setViewMode(QFileDialog::Detail);
```

创建自己的文件对话框时需要使用的最后一个重要功能是selectedFiles()。

```
QStringList fileNames;
if (dialog.exec())
    fileNames = dialog.selectedFiles();
```

在上面的示例中，创建并显示了一个模式文件对话框。如果用户单击“确定”，他们选择的文件将被放入`fileName`.

对话框的工作目录可以设置为setDirectory()。可以使用以下命令选择当前目录中的每个文件selectFile （） 功能。

这Standard Dialogs示例展示了如何使用 QFileDialog 以及其他内置 Qt 对话框。

默认情况下，如果平台有的话，将使用平台本机文件对话框。在这种情况下，否则将用于构造对话框的小部件将不会被实例化，因此相关的访问器，例如layout （） 和itemDelegate() 将返回 null。此外，并非所有平台都显示带有标题栏的文件对话框，因此请注意，用户可能看不到标题文本。您可以设置DontUseNativeDialog选项以确保使用基于小部件的实现而不是本机对话框。

也可以看看QDir,QFileInfo,QFile,QColorDialog,QFontDialog,Standard Dialogs Example和 Qt Widgets - 应用程序示例。

会员类型文档

enum QFileDialog::AcceptMode

持续的	价值
QFileDialog::AcceptOpen	0
QFileDialog::AcceptSave	1

enum QFileDialog::DialogLabel

持续的	价值
QFileDialog::LookIn	0
QFileDialog::FileName	1
QFileDialog::FileType	2
QFileDialog::Accept	3
QFileDialog::Reject	4

enum QFileDialog::FileMode

该枚举用于指示用户可以在文件对话框中选择什么；即如果用户单击“确定”，对话框将返回什么。

持续的	价值	描述
`QFileDialog::AnyFile`	`0`	文件的名称，无论是否存在。
`QFileDialog::ExistingFile`	`1`	单个现有文件的名称。
`QFileDialog::Directory`	`2`	目录的名称。文件和目录都会显示。但是，本机 Windows 文件对话框不支持在目录选择器中显示文件。
`QFileDialog::ExistingFiles`	`3`	零个或多个现有文件的名称。

也可以看看setFileMode()。

枚举 QFileDialog::选项 标志 QFileDialog::选项

持续的	价值	描述
`QFileDialog::ShowDirsOnly`	`0x00000001`	仅在文件对话框中显示目录。默认情况下，文件和目录都会显示。（仅在Directory文件模式。）
`QFileDialog::DontResolveSymlinks`	`0x00000002`	不要解析文件对话框中的符号链接。默认情况下，符号链接已解析。
`QFileDialog::DontConfirmOverwrite`	`0x00000004`	如果选择现有文件，则不要求确认。默认情况下要求确认。

注意：使用本机文件对话框时，macOS 不支持此选项。

持续的	价值	描述
`QFileDialog::DontUseNativeDialog`	`0x00000008`	不要使用本机文件对话框。默认情况下，使用本机文件对话框，除非您使用QFileDialog其中包含Q_OBJECT宏，或者平台没有您需要的类型的本机对话框。

注意：必须在更改对话框属性或显示对话框之前设置此选项。

持续的	价值	描述
`QFileDialog::ReadOnly`	`0x00000010`	指示模型是只读的。
`QFileDialog::HideNameFilterDetails`	`0x00000020`	指示是否隐藏文件名过滤器详细信息。
`QFileDialog::DontUseCustomDirectoryIcons`	`0x00000040`	始终使用默认目录图标。某些平台允许用户设置不同的图标。自定义图标查找会对网络或可移动驱动器的性能产生重大影响。设置此项将启用图标提供程序中的 QFileIconProvider::DontUseCustomDirectoryIcons 选项。该枚举值是在 Qt 5.2 中添加的。

选项类型是QFlags < Option > 的类型定义。它存储选项值的 OR 组合。

enum QFileDialog::ViewMode

该枚举描述了文件对话框的查看模式；即将显示有关每个文件的哪些信息。

持续的	价值	描述
`QFileDialog::Detail`	`0`	显示目录中每个项目的图标、名称和详细信息。
`QFileDialog::List`	`1`	仅显示目录中每个项目的图标和名称。

也可以看看setViewMode()。

财产文件

acceptMode : AcceptMode

该属性保存对话框的接受模式

操作模式定义对话框是用于打开还是保存文件。

默认情况下，该属性设置为AcceptOpen。

访问功能：

QFileDialog::AcceptMode	acceptMode() const
void	**setAcceptMode**(QFileDialog::AcceptMode *mode*)

也可以看看AcceptMode。

defaultSuffix : QString

如果未指定其他后缀，则添加到文件名的后缀

该属性指定一个字符串，如果文件名还没有后缀，则该字符串将被添加到文件名中。后缀通常用于指示文件类型（例如“txt”指示文本文件）。

如果第一个字符是点 ('.')，则将其删除。

访问功能：

QString	defaultSuffix() const
void	**setDefaultSuffix**(const QString &*suffix*)

fileMode : FileMode

该属性保存对话框的文件模式

文件模式定义用户期望在对话框中选择的项目的数量和类型。

默认情况下，该属性设置为AnyFile。

该函数将设置标签FileName和Accept DialogLabels。调用 setFileMode() 后可以设置自定义文本。

访问功能：

QFileDialog::FileMode	fileMode() const
void	**setFileMode**(QFileDialog::FileMode *mode*)

也可以看看FileMode。

options : Options

该属性包含影响对话框外观的各种选项

默认情况下，所有选项均被禁用。

应在更改对话框属性或显示对话框之前设置选项（特别是 DontUseNativeDialogs 选项）。

在对话框可见时设置选项并不能保证立即对对话框产生影响（取决于选项和平台）。

更改其他属性后设置选项可能会导致这些值不起作用。

访问功能：

QFileDialog::Options	options() const
void	**setOptions**(QFileDialog::Options *options*)

也可以看看setOption（） 和testOption()。

supportedSchemes : QStringList

此属性保存文件对话框应允许导航到的 URL 方案。

设置此属性可以限制用户可以选择的 URL 类型。这是应用程序声明它将支持获取文件内容的协议的一种方式。空列表意味着不应用任何限制（默认）。对本地文件（“文件”方案）的支持是隐式的并且始终启用；没有必要将其包含在限制中。

访问功能：

QStringList	supportedSchemes() const
void	**setSupportedSchemes**(const QStringList &*schemes*)

viewMode : ViewMode

该属性保存文件和目录在对话框中显示的方式

默认情况下，该Detail模式用于显示有关文件和目录的信息。

访问功能：

QFileDialog::ViewMode	viewMode() const
void	**setViewMode**(QFileDialog::ViewMode *mode*)

也可以看看ViewMode。

成员函数文档

*QFileDialog::QFileDialog(QWidget *parent, Qt::WindowFlags flags)*

使用给定的构造一个文件对话框*parent*和小部件*flags*。

*[explicit]QFileDialog::QFileDialog(QWidget *parent = nullptr, const QString &caption = QString(), const QString &directory = QString(), const QString &filter = QString())*

使用给定的构造一个文件对话框*parent*和*caption*最初显示指定的内容*directory*。目录的内容在显示在对话框中之前会被过滤，使用由分号分隔的过滤器列表指定*filter*。

[virtual]QFileDialog::~QFileDialog()

销毁文件对话框。

[override virtual protected]void QFileDialog::accept()

重新实现：QDialog::accept()。

[override virtual protected]void QFileDialog::changeEvent(QEvent *e)

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

[signal]void QFileDialog::currentChanged(const QString &path)

当本地操作的当前文件发生更改时，会发出此信号，并以新文件名作为*path*范围。

也可以看看filesSelected()。

[signal]void QFileDialog::currentUrlChanged(const QUrl &url)

当当前文件更改时，会发出此信号，并以新文件 URL 作为*url*范围。

也可以看看urlsSelected()。

QDir QFileDialog::directory() const

返回当前在对话框中显示的目录。

也可以看看setDirectory()。

[signal]void QFileDialog::directoryEntered(const QString &directory)

当用户输入一个本地操作时，会发出此信号*directory*。

QUrl QFileDialog::directoryUrl() const

返回对话框中当前显示的目录的 url。

也可以看看setDirectoryUrl()。

[signal]void QFileDialog::directoryUrlEntered(const QUrl &directory)

当用户输入*directory*。

[override virtual protected]void QFileDialog::done(int result)

重新实现：QDialog::done（int r）。

[signal]void QFileDialog::fileSelected(const QString &file)

当本地操作的选择发生变化并且对话框被接受时，会发出此信号，并选择（可能为空）*file*。

也可以看看currentChanged（） 和QDialog::Accepted。

[signal]void QFileDialog::filesSelected(const QStringList &selected)

当本地操作的选择发生变化并且对话框被接受时，会发出此信号，并带有（可能为空）列表*selected*文件。

也可以看看currentChanged（） 和QDialog::Accepted。

QDir::Filters QFileDialog::filter() const

返回显示文件时使用的过滤器。

也可以看看setFilter()。

[signal]void QFileDialog::filterSelected(const QString &filter)

当用户选择一个时会发出此信号*filter*。

*[static]QString QFileDialog::getExistingDirectory(QWidget *parent = nullptr, const QString &caption = QString(), const QString &dir = QString(), QFileDialog::Options options = ShowDirsOnly)*

这是一个方便的静态函数，它将返回用户选择的现有目录。

```
QString dir = QFileDialog::getExistingDirectory(this, tr("Open Directory"),
                                                "/home",
                                                QFileDialog::ShowDirsOnly
                                                | QFileDialog::DontResolveSymlinks);
```

该函数使用给定的参数创建一个模式文件对话框*parent*小部件。如果*parent*不是`nullptr`，对话框将显示在父窗口部件的中心。

对话框的工作目录设置为*dir*，并且标题设置为*caption*。其中任何一个都可以是空字符串，在这种情况下，将分别使用当前目录和默认标题。

这*options*参数包含有关如何运行对话框的各种选项，请参阅QFileDialog::Option枚举以获取有关可以传递的标志的更多信息。为了确保本机文件对话框，ShowDirsOnly必须设置。

在 Windows 和 macOS 上，此静态函数将使用本机文件对话框，而不是QFileDialog。但是，本机 Windows 文件对话框不支持在目录选择器中显示文件。你需要通过DontUseNativeDialog使用 a 显示文件QFileDialog。

请注意，macOS 本机文件对话框不显示标题栏。

在 Unix/X11 上，文件对话框的正常行为是解析并遵循符号链接。例如，如果`/usr/tmp`是 的符号链接，则输入后`/var/tmp`文件对话框将变为。如果`/var/tmp``/usr/tmp`*options*包括DontResolveSymlinks，文件对话框会将符号链接视为常规目录。

在 Windows 上，该对话框将旋转一个阻塞模式事件循环，该循环不会调度任何 QTimers，并且如果*parent*不是的`nullptr`话，它将把对话框定位在父级标题栏的正下方。

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QFileDialog构造函数。

也可以看看getOpenFileName(),getOpenFileNames （） ， 和getSaveFileName()。

*[static]QUrl QFileDialog::getExistingDirectoryUrl(QWidget *parent = nullptr, const QString &caption = QString(), const QUrl &dir = QUrl(), QFileDialog::Options options = ShowDirsOnly, const QStringList &supportedSchemes = QStringList())*

这是一个方便的静态函数，它将返回用户选择的现有目录。如果用户按下“取消”，它将返回一个空 URL。

该函数的使用方式类似于QFileDialog::getExistingDirectory()。尤其*parent*,*caption*,*dir*和*options*以完全相同的方式使用。

主要区别与QFileDialog::getExistingDirectory() 来自为用户提供选择远程目录的能力。这就是为什么返回类型和类型*dir*是QUrl。

这*supportedSchemes*参数允许限制用户可以选择的 URL 类型。这是应用程序声明它将支持获取文件内容的协议的一种方式。空列表意味着不应用任何限制（默认）。对本地文件（“文件”方案）的支持是隐式的并且始终启用；没有必要将其包含在限制中。

如果可能，此静态函数将使用本机文件对话框而不是QFileDialog。在不支持选择远程文件的平台上，Qt 将允许仅选择本地文件。

也可以看看getExistingDirectory(),getOpenFileUrl(),getOpenFileUrls（），和getSaveFileUrl()。

[static]void QFileDialog::getOpenFileContent(const QString &nameFilter, const std::function<void (const QString &, const QByteArray &)> &fileOpenCompleted)

这是一个方便的静态函数，它将返回用户选择的文件的内容。

此函数用于访问 Qt for WebAssembly 上的本地文件，其中 Web 沙箱对此类访问的发生方式进行了限制。它的实现将使浏览器显示一个本机文件对话框，用户在其中根据参数进行文件选择*nameFilter*。

它也可以在其他平台上使用，在那里它将回退到使用QFileDialog。

该函数是异步的并立即返回。这*fileOpenCompleted*当选择文件并将其内容读入内存时，将调用回调。

```
auto fileContentReady = [](const QString &fileName, const QByteArray &fileContent) {
    if (fileName.isEmpty()) {
        // No file was selected
    } else {
        // Use fileName and fileContent
    }
};
QFileDialog::getOpenFileContent("Images (*.png *.xpm *.jpg)",  fileContentReady);
```

*[static]QString QFileDialog::getOpenFileName(QWidget **parent*** = **nullptr, const QString &caption = QString(), const QString &dir = QString(), const QString &filter = QString(), QString** *selectedFilter** = *nullptr, QFileDialog::Options options = Options())*

这是一个方便的静态函数，它返回用户选择的现有文件。如果用户按下“取消”，它将返回一个空字符串。

```
QString fileName = QFileDialog::getOpenFileName(this, tr("Open File"),
                                                "/home",
                                                tr("Images (*.png *.xpm *.jpg)"));
```

该函数使用给定的值创建一个模式文件对话框*parent*小部件。如果*parent*不是`nullptr`，对话框将显示在父窗口部件的中心。

文件对话框的工作目录将设置为*dir*。如果*dir*包括文件名，该文件将被选择。仅匹配给定的文件*filter*显示。所选过滤器设置为*selectedFilter*。参数*dir*,*selectedFilter*， 和*filter*可能是空字符串。如果需要多个过滤器，请用“;;”分隔它们，例如：

```
"Images (*.png *.xpm *.jpg);;Text files (*.txt);;XML files (*.xml)"
```

这*options*参数包含有关如何运行对话框的各种选项，请参阅QFileDialog::Option枚举以获取有关可以传递的标志的更多信息。

对话框的标题设置为*caption*。如果*caption*未指定则将使用默认标题。

在 Windows 和 macOS 上，此静态函数将使用本机文件对话框，而不是QFileDialog。请注意，macOS 本机文件对话框不显示标题栏。

在 Windows 上，该对话框将旋转一个阻塞模式事件循环，该循环不会调度任何 QTimers，并且如果*parent*不是的`nullptr`话，它将把对话框定位在父级标题栏的正下方。

在 Unix/X11 上，文件对话框的正常行为是解析并遵循符号链接。例如，如果`/usr/tmp`是 的符号链接，则输入后`/var/tmp`文件对话框将变为。如果`/var/tmp``/usr/tmp`*options*包括DontResolveSymlinks，文件对话框会将符号链接视为常规目录。

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QFileDialog构造函数。

也可以看看getOpenFileNames(),getSaveFileName （） ， 和getExistingDirectory()。

[static]QStringList QFileDialog::getOpenFileNames(QWidget ***parent*** *= **nullptr, const QString &caption = QString(), const QString &dir = QString(), const QString &filter = QString(),** QString selectedFilter* *= nullptr, QFileDialog::Options options = Options())*

这是一种方便的静态函数，它将返回用户选择的一个或多个现有文件。

```
QStringList files = QFileDialog::getOpenFileNames(
                        this,
                        "Select one or more files to open",
                        "/home",
                        "Images (*.png *.xpm *.jpg)");
```

该函数使用给定的参数创建一个模式文件对话框*parent*小部件。如果*parent*不是`nullptr`，对话框将显示在父窗口部件的中心。

文件对话框的工作目录将设置为*dir*。如果*dir*包括文件名，该文件将被选择。过滤器设置为*filter*这样就只显示那些与过滤器匹配的文件。所选过滤器设置为*selectedFilter*。参数*dir*,*selectedFilter*和*filter*可能是空字符串。如果需要多个过滤器，请用“;;”分隔它们，例如：

```
"Images (*.png *.xpm *.jpg);;Text files (*.txt);;XML files (*.xml)"
```

对话框的标题设置为*caption*。如果*caption*未指定则将使用默认标题。

在 Windows 和 macOS 上，此静态函数将使用本机文件对话框，而不是QFileDialog。请注意，macOS 本机文件对话框不显示标题栏。

在 Windows 上，该对话框将旋转一个阻塞模式事件循环，该循环不会调度任何 QTimers，并且如果*parent*不是的`nullptr`话，它将把对话框定位在父级标题栏的正下方。

在 Unix/X11 上，文件对话框的正常行为是解析并遵循符号链接。例如，如果`/usr/tmp`是 的符号链接，则输入后`/var/tmp`文件对话框将变为。这`/var/tmp``/usr/tmp`*options*参数包含有关如何运行对话框的各种选项，请参阅QFileDialog::Option枚举以获取有关可以传递的标志的更多信息。

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QFileDialog构造函数。

也可以看看getOpenFileName(),getSaveFileName（），和getExistingDirectory()。

[static]QUrl QFileDialog::getOpenFileUrl(QWidget ***parent* = nullptr,** ***const QString &caption* = QString(), *const QUrl &dir* = QUrl(), *const QString &filter* = QString(), *QString selectedFilter* = *nullptr, QFileDialog::Options options* = Options(), *const QStringList &supportedSchemes* = QStringList())**

这是一个方便的静态函数，它返回用户选择的现有文件。如果用户按下“取消”，它将返回一个空 URL。

该函数的使用方式类似于QFileDialog::getOpenFileName()。尤其*parent*,*caption*,*dir*,*filter*,*selectedFilter*和*options*以完全相同的方式使用。

主要区别与QFileDialog::getOpenFileName() 来自为用户提供选择远程文件的能力。这就是为什么返回类型和类型*dir*是QUrl。

这*supportedSchemes*参数允许限制用户可以选择的 URL 类型。这是应用程序声明它将支持获取文件内容的协议的一种方式。空列表意味着不应用任何限制（默认）。对本地文件（“文件”方案）的支持是隐式的并且始终启用；没有必要将其包含在限制中。

如果可能，此静态函数将使用本机文件对话框而不是QFileDialog。在不支持选择远程文件的平台上，Qt 将允许仅选择本地文件。

也可以看看getOpenFileName(),getOpenFileUrls(),getSaveFileUrl（），和getExistingDirectoryUrl()。

[static]QList[QUrl](https://doc-qt-io.translate.goog/qt-6/qurl.html?_x_tr_sl=auto&_x_tr_tl=zh-CN&_x_tr_hl=zh-CN&_x_tr_pto=wapp) QFileDialog::getOpenFileUrls(QWidget **parent* = nullptr, const **QString** &**caption** = **QString**(), const **QUrl** &**dir** = **QUrl**(), const **QString** &**filter** = **QString**(), **QString** *selectedFilter* = nullptr, QFileDialog::Options options = Options(), const QStringList &supportedSchemes = QStringList())*

这是一种方便的静态函数，它将返回用户选择的一个或多个现有文件。如果用户按“取消”，它将返回一个空列表。

该函数的使用方式类似于QFileDialog::getOpenFileNames()。尤其*parent*,*caption*,*dir*,*filter*,*selectedFilter*和*options*以完全相同的方式使用。

主要区别与QFileDialog::getOpenFileNames() 来自为用户提供选择远程文件的能力。这就是为什么返回类型和类型*dir*分别是QList[QUrl](https://doc-qt-io.translate.goog/qt-6/qurl.html?_x_tr_sl=auto&_x_tr_tl=zh-CN&_x_tr_hl=zh-CN&_x_tr_pto=wapp) 和QUrl。

这*supportedSchemes*参数允许限制用户可以选择的 URL 类型。这是应用程序声明它将支持获取文件内容的协议的一种方式。空列表意味着不应用任何限制（默认）。对本地文件（“文件”方案）的支持是隐式的并且始终启用；没有必要将其包含在限制中。

如果可能，此静态函数将使用本机文件对话框而不是QFileDialog。在不支持选择远程文件的平台上，Qt 将允许仅选择本地文件。

也可以看看getOpenFileNames(),getOpenFileUrl(),getSaveFileUrl（），和getExistingDirectoryUrl()。

[static]QString QFileDialog::getSaveFileName(QWidget **parent* = **nullptr, const QString &caption = QString(), const QString &dir = QString(), const QString &filter = QString(),** **QString** *selectedFilter* = nullptr, QFileDialog::Options options = Options())*

这是一个方便的静态函数，它将返回用户选择的文件名。该文件不必存在。

它使用给定的内容创建一个模式文件对话框*parent*小部件。如果*parent*不是nullptr，对话框将显示在父窗口部件的中心。

```
QString fileName = QFileDialog::getSaveFileName(this, tr("Save File"),
                           "/home/jana/untitled.png",
                           tr("Images (*.png *.xpm *.jpg)"));
```

文件对话框的工作目录将设置为*dir*。如果*dir*包括文件名，该文件将被选择。仅匹配匹配的文件*filter*显示。所选过滤器设置为*selectedFilter*。参数*dir*,*selectedFilter*， 和*filter*可能是空字符串。多个过滤器用“;;”分隔。例如：

```
"Images (*.png *.xpm *.jpg);;Text files (*.txt);;XML files (*.xml)"
```

这*options*参数包含有关如何运行对话框的各种选项，请参阅QFileDialog::Option枚举以获取有关可以传递的标志的更多信息。

可以通过设置选择默认过滤器*selectedFilter*到所需的值。

对话框的标题设置为*caption*。如果*caption*未指定，将使用默认标题。

在 Windows 和 macOS 上，此静态函数将使用本机文件对话框，而不是QFileDialog。

在 Windows 上，该对话框将旋转一个阻塞模式事件循环，该循环不会调度任何 QTimers，并且如果*parent*不是的`nullptr`话，它将把对话框定位在父级标题栏的正下方。在 macOS 上，通过其本机文件对话框，过滤器参数将被忽略。

在 Unix/X11 上，文件对话框的正常行为是解析并遵循符号链接。例如，如果`/usr/tmp`是 的符号链接，则输入后`/var/tmp`文件对话框将变为。如果`/var/tmp``/usr/tmp`*options*包括DontResolveSymlinks文件对话框会将符号链接视为常规目录。

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QFileDialog构造函数。

也可以看看getOpenFileName(),getOpenFileNames（），和getExistingDirectory()。

[static]QUrl QFileDialog::getSaveFileUrl(QWidget ***parent* = nullptr, const QString &caption = QString(), const QUrl &dir = QUrl(), const QString &filter = QString(),*** *QString selectedFilter* = nullptr, QFileDialog::Options options = Options(), const QStringList &supportedSchemes = QStringList())*

这是一个方便的静态函数，它返回用户选择的文件。该文件不必存在。如果用户按下“取消”，它将返回一个空 URL。

该函数的使用方式类似于QFileDialog::getSaveFileName()。尤其*parent*,*caption*,*dir*,*filter*,*selectedFilter*和*options*以完全相同的方式使用。

主要区别与QFileDialog::getSaveFileName() 来自为用户提供选择远程文件的能力。这就是为什么返回类型和类型*dir*是QUrl。

这*supportedSchemes*参数允许限制用户可以选择的 URL 类型。这是应用程序声明它将支持的保存文件内容的协议的一种方式。空列表意味着不应用任何限制（默认）。对本地文件（“文件”方案）的支持是隐式的并且始终启用；没有必要将其包含在限制中。

如果可能，此静态函数将使用本机文件对话框而不是QFileDialog。在不支持选择远程文件的平台上，Qt 将允许仅选择本地文件。

也可以看看getSaveFileName(),getOpenFileUrl(),getOpenFileUrls（），和getExistingDirectoryUrl()。

QStringList QFileDialog::history() const

以路径列表的形式返回文件对话框的浏览历史记录。

也可以看看setHistory()。

QAbstractFileIconProvider *QFileDialog::iconProvider() const

返回文件对话框使用的图标提供程序。

也可以看看setIconProvider()。

QAbstractItemDelegate *QFileDialog::itemDelegate() const

返回用于在文件对话框的视图中呈现项目的项目委托。

也可以看看setItemDelegate()。

QString QFileDialog::labelText(*QFileDialog::DialogLabel* label) const

返回指定文件对话框中显示的文本*label*。

也可以看看setLabelText()。

QStringList QFileDialog::mimeTypeFilters() const

返回在此文件对话框上运行的 MIME 类型过滤器。

也可以看看setMimeTypeFilters()。

QStringList QFileDialog::nameFilters() const

返回在此文件对话框上运行的文件类型过滤器。

也可以看看setNameFilters()。

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

该函数将其信号之一连接到由*receiver*和*member*。具体信号取决于filesSelected（） 如果fileMode是ExistingFiles和fileSelected（） 如果fileMode是别的什么。

当对话框关闭时，信号将从插槽断开。

*QAbstractProxyModel *QFileDialog::proxyModel() const*

返回文件对话框使用的代理模型。默认情况下没有设置代理。

也可以看看setProxyModel()。

bool QFileDialog::restoreState(const QByteArray &state)

将对话框的布局、历史记录和当前目录恢复到*state*指定的。

通常这与QSettings恢复过去会话的大小。

`false`如果有错误则返回

[static]void QFileDialog::saveFileContent(const QByteArray &fileContent, const QString &fileNameHint = QString())

这是一个方便的静态函数，可以节省*fileContent*使用用户选择的文件名和位置保存到文件。*fileNameHint*可以向用户提供建议文件名。

此函数用于将文件保存到 Qt for WebAssembly 上的本地文件系统，其中 Web 沙箱对此类访问的发生方式进行了限制。它的实现将使浏览器显示一个本机文件对话框，用户可以在其中选择文件。

它也可以在其他平台上使用，在那里它将回退到使用QFileDialog。

该函数是异步的并立即返回。

```
QByteArray imageData; // obtained from e.g. QImage::save()
QFileDialog::saveFileContent(imageData, "myimage.png"); // with filename hint
// OR
QFileDialog::saveFileContent(imageData); // no filename hint
```

QByteArray QFileDialog::saveState() const

保存对话框的布局、历史记录和当前目录的状态。

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。

void QFileDialog::selectFile(const QString &filename)

选择给定的*filename*在文件对话框中。

也可以看看selectedFiles()。

void QFileDialog::selectMimeTypeFilter(const QString &filter)

设置当前的 MIME 类型*filter*。

void QFileDialog::selectNameFilter(const QString &filter)

设置当前文件类型*filter*。可以传入多个过滤器*filter*用分号或空格分隔它们。

也可以看看setNameFilter(),setNameFilters（），和selectedNameFilter()。

void QFileDialog::selectUrl(const QUrl &url)

选择给定的*url*在文件对话框中。

注：非本地人QFileDialog仅支持本地文件。

也可以看看selectedUrls()。

QStringList QFileDialog::selectedFiles() const

返回包含对话框中所选文件的绝对路径的字符串列表。如果没有选择文件，或者模式不是ExistingFiles或者ExistingFile, selectedFiles() 包含视口中的当前路径。

也可以看看selectedNameFilter（） 和selectFile()。

QString QFileDialog::selectedMimeTypeFilter() const

返回用户在文件对话框中选择的文件的 mimetype。

QString QFileDialog::selectedNameFilter() const

返回用户在文件对话框中选择的过滤器。

也可以看看selectedFiles()。

QList[QUrl](https://doc-qt-io.translate.goog/qt-6/qurl.html?_x_tr_sl=auto&_x_tr_tl=zh-CN&_x_tr_hl=zh-CN&_x_tr_pto=wapp) QFileDialog::selectedUrls() const

返回包含对话框中所选文件的 url 列表。如果没有选择文件，或者模式不是ExistingFiles或者ExistingFile，selectedUrls() 包含视口中的当前路径。

也可以看看selectedNameFilter（） 和selectUrl()。

void QFileDialog::setDirectory(const QString &directory)

设置文件对话框的当前*directory*。

注意：在 iOS 上，如果您设置*directory*到QStandardPaths::standardLocations(QStandardPaths::PicturesLocation).last()，本机图像选择器对话框将用于访问用户的相册。返回的文件名可以使用加载QFile以及相关的API。要启用此功能，项目文件中分配给 QMAKE_INFO_PLIST 的 Info.plist 必须包含键`NSPhotoLibraryUsageDescription`。有关此密钥的更多信息，请参阅 Apple 的 Info.plist 文档。该功能是在 Qt 5.5 中添加的。

也可以看看directory()。

void QFileDialog::setDirectory(const QDir &directory)

这是一个过载功能。

void QFileDialog::setDirectoryUrl(const QUrl &directory)

设置文件对话框的当前*directory*网址。

注：非本地人QFileDialog仅支持本地文件。

注意：在 Windows 上，可以传递代表*虚拟文件夹*之一的 URL，例如“计算机”或“网络”。这是通过传递一个来完成的QUrl使用该方案`clsid`后跟 CLSID 值并删除花括号。例如，URL`clsid:374DE290-123F-4565-9164-39C4925E467B`表示下载位置。有关可能值的完整列表，请参阅 MSDN 文档KNOWNFOLDERID。该功能是在 Qt 5.5 中添加的。

也可以看看directoryUrl（） 和QUuid。

void QFileDialog::setFilter(QDir::Filters filters)

将模型使用的过滤器设置为*filters*。过滤器用于指定应显示的文件类型。

也可以看看filter()。

void QFileDialog::setHistory(const QStringList &paths)

设置文件对话框的浏览历史记录以包含给定的*paths*。

也可以看看history()。

*void QFileDialog::setIconProvider(QAbstractFileIconProvider *provider)*

将文件对话框使用的图标提供程序设置为指定的*provider*。

也可以看看iconProvider()。

*void QFileDialog::setItemDelegate(QAbstractItemDelegate *delegate)*

将用于在文件对话框的视图中渲染项目的项目委托设置为给定的*delegate*。

任何现有委托都将被删除，但不会被删除。QFileDialog不拥有所有权*delegate*。

警告：您不应在视图之间共享委托的同一实例。这样做可能会导致不正确或不直观的编辑行为，因为连接到给定委托的每个视图都可能会收到closeEditor() 信号，并尝试访问、修改或关闭已关闭的编辑器。

请注意，使用的模型是QFileSystemModel。它具有自定义项目数据角色，由Roles枚举。您可以使用QFileIconProvider如果您只想要自定义图标。

也可以看看itemDelegate(),setIconProvider（），和QFileSystemModel。

void QFileDialog::setLabelText(QFileDialog::DialogLabel label, const QString &text)

设置*text*显示在指定的文件对话框中*label*。

也可以看看labelText()。

void QFileDialog::setMimeTypeFilters(const QStringList &filters)

设置*filters*在文件对话框中使用，来自 MIME 类型列表。

方便的方法setNameFilters()。用途QMimeType根据每个 MIME 类型中定义的全局模式和描述创建名称过滤器。

使用 application/octet-stream 作为“所有文件 (*)”过滤器，因为这是所有文件的基本 MIME 类型。

调用 setMimeTypeFilters 会覆盖任何先前设置的名称过滤器，并更改nameFilters()。

```
QStringList mimeTypeFilters({"image/jpeg", // will show "JPEG image (*.jpeg *.jpg *.jpe)
                             "image/png",  // will show "PNG image (*.png)"
                             "application/octet-stream" // will show "All files (*)"
                            });

QFileDialog dialog(this);
dialog.setMimeTypeFilters(mimeTypeFilters);
dialog.exec();
```

也可以看看mimeTypeFilters()。

void QFileDialog::setNameFilter(const QString &filter)

将文件对话框中使用的过滤器设置为给定的*filter*。

如果*filter*包含一对括号，其中包含一个或多个文件名通配符模式，并用空格分隔，则仅将括号中包含的文本用作过滤器。这意味着这些调用都是等效的：

```
dialog.setNameFilter("All C++ files (*.cpp *.cc *.C *.cxx *.c++)");
dialog.setNameFilter("*.cpp *.cc *.C *.cxx *.c++");
```

注意：对于 Android 的本机文件对话框，将使用与给定名称过滤器匹配的 mime 类型，因为仅支持 mime 类型。

也可以看看setMimeTypeFilters （） 和setNameFilters()。

void QFileDialog::setNameFilters(const QStringList &filters)

设置*filters*在文件对话框中使用。

请注意，过滤器.**不可移植，因为文件扩展名决定文件类型的历史假设在每个操作系统上都不一致。名称中可能没有点的文件（例如，`Makefile`）。在本机 Windows 文件对话框中，**.***将匹配此类文件，而在其他类型的文件对话框中则可能不匹配。因此，如果您想选择任何文件，最好使用* 。**

```
const QStringList filters({"Image files (*.png *.xpm *.jpg)",
                           "Text files (*.txt)",
                           "Any files (*)"
                          });
QFileDialog dialog(this);
dialog.setNameFilters(filters);
dialog.exec();
```

setMimeTypeFilters() 的优点是为每种文件类型提供所有可能的名称过滤器。例如，JPEG 图像具有三种可能的扩展名：如果您的应用程序可以打开此类文件，选择`image/jpeg`mime 类型作为过滤器将允许您打开所有这些文件。

也可以看看nameFilters()。

void QFileDialog::setOption(QFileDialog::Option option, bool on = true)

设置给定的*option*启用如果*on*是真的; 否则，清除给定的*option*。

应在更改对话框属性或显示对话框之前设置选项（特别是 DontUseNativeDialogs 选项）。

在对话框可见时设置选项并不能保证立即对对话框产生影响（取决于选项和平台）。

更改其他属性后设置选项可能会导致这些值不起作用。

也可以看看options和testOption()。

*void QFileDialog::setProxyModel(QAbstractProxyModel *proxyModel)*

将视图的模型设置为给定的*proxyModel*。如果您想修改底层模型，这非常有用；例如，添加列、过滤数据或添加驱动器。

任何现有的代理模型都将被删除，但不会被删除。文件对话框将取得该文件的所有权*proxyModel*。

也可以看看proxyModel()。

void QFileDialog::setSidebarUrls(const QList[QUrl](https://doc-qt-io.translate.goog/qt-6/qurl.html?_x_tr_sl=auto&_x_tr_tl=zh-CN&_x_tr_hl=zh-CN&_x_tr_pto=wapp) &urls)

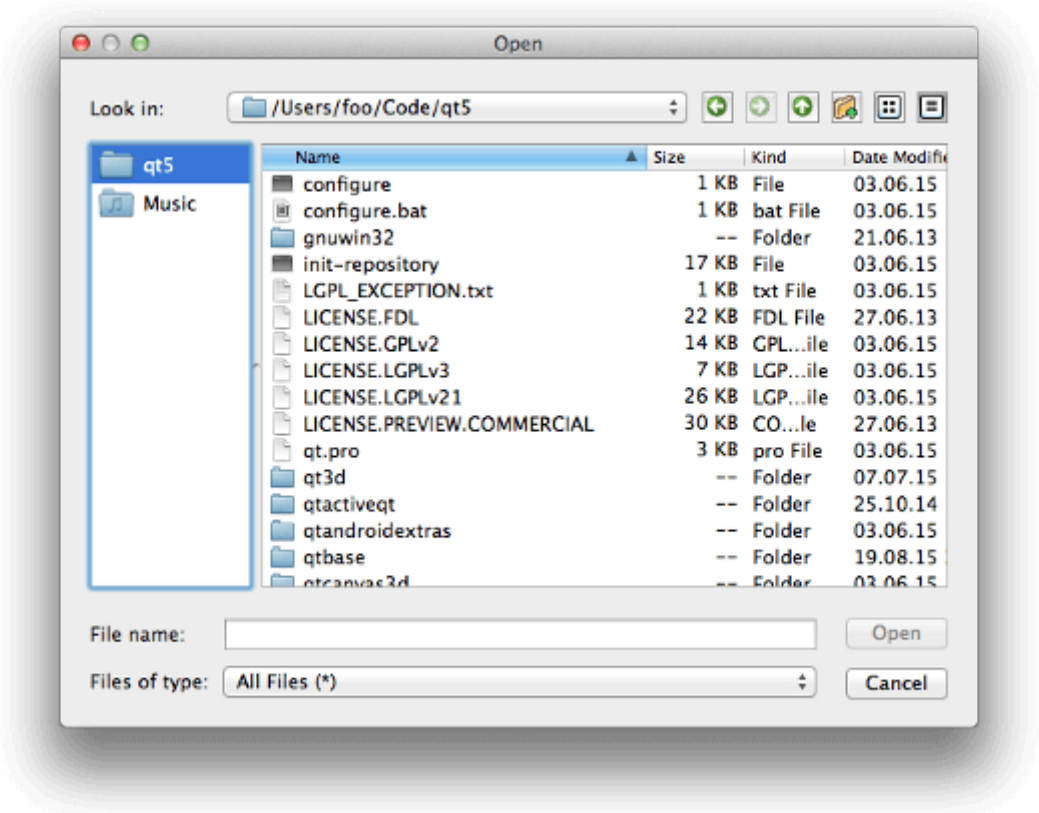
设置*urls*位于侧边栏中。

例如：

```
    QList< QUrl > urls;
    urls << QUrl::fromLocalFile("/Users/foo/Code/qt5")
         <<
QUrl::fromLocalFile(QStandardPaths::standardLocations(QStandardPaths::MusicLocation).first());

    QFileDialog dialog;
    dialog.setSidebarUrls(urls);
    dialog.setFileMode(QFileDialog::AnyFile);
    if (dialog.exec()) {
        // ...
    }
```

文件对话框将如下所示：



也可以看看sidebarUrls()。

[override virtual]void QFileDialog::setVisible(bool visible)

重新实现：QDialog::setVisible（布尔值可见）。

QList[QUrl](https://doc-qt-io.translate.goog/qt-6/qurl.html?_x_tr_sl=auto&_x_tr_tl=zh-CN&_x_tr_hl=zh-CN&_x_tr_pto=wapp) QFileDialog::sidebarUrls() const

返回当前位于侧边栏中的 url 列表

也可以看看setSidebarUrls()。

bool QFileDialog::testOption(QFileDialog::Option option) const

`true`如果给定则返回*option*已启用；否则，返回 false。

也可以看看options和setOption()。

[signal]void QFileDialog::urlSelected(const QUrl &url)

当选择更改并接受对话框时，会发出此信号，并选择（可能为空）*url*。

也可以看看currentUrlChanged（） 和QDialog::Accepted。

[signal]void QFileDialog::urlsSelected(const QList[QUrl](https://doc-qt-io.translate.goog/qt-6/qurl.html?_x_tr_sl=auto&_x_tr_tl=zh-CN&_x_tr_hl=zh-CN&_x_tr_pto=wapp) &urls)

当选择更改并接受对话框时，会发出此信号以及所选的（可能为空）列表*urls*。

也可以看看currentUrlChanged（） 和QDialog::Accepted。