# 学习网址：

库介绍

<http://nullege.com/codes/search?cq=PyQt4.QtGui.QTableView>

<http://pyqt.sourceforge.net/Docs/PyQt4/classes.html>

# [Pyqt4学习笔记-简单案例](http://www.cnblogs.com/shadow-ccos/p/5176043.html)

2016-02-03 16:35 by 影澜, 1281 阅读, 0 评论, [收藏](http://www.cnblogs.com/shadow-ccos/p/5176043.html), [编辑](https://i.cnblogs.com/EditPosts.aspx?postid=5176043)

官方文档：http://zetcode.com/gui/pyqt4/

中文文档：http://www.qaulau.com/books/PyQt4\_Tutorial/index.html



QtCore：核心的非GUI功能，对时间、文件、目录、各种数据类型、流、网址、媒体类型、线程或进程进行处理

QtGui：图形化窗口组件和及相关类。包括如按钮、窗体、状态栏、滑块、位图、颜色、字体等等

QtHelp：包含了用于创建和查看可查找的文档的类

QtNetwork：包括网络编程的类。用来编写TCP/IP和UDP的客户端和服务器

QtOpenGL：使用OpenGL库来渲染3D和2D图形

QtScript：包含了使PyQt应用程序使用JavaScript解释器编写脚本的类

QtSql：提供操作数据库的类

QtSvg：提供了显示SVG文件内容的类。可缩放矢量图形(SVG)

QtTest：包含了对PyQt应用程序进行单元测试的功能

QtWebKit：实现了基于开源浏览器引擎WebKit的浏览器引擎

QtXml：处理XML文件的类，该模块提供了SAX和DOM API的接口

QtXmlPatterns：包含的类实现了对XML和自定义数据模型的XQuery和XPath的支持

phonon：包含的类实现了跨平台的多媒体框架，可以在PyQt应用程序中使用音频和视频内容

QtMultimedia：提供了低级的多媒体功能，通常更多使用 phonon 模块

QtAssistant：包含的类允许集成 Qt Assistant 到PyQt应用程序中，提供在线帮助。

QtDesigner：包含的类允许使用PyQt扩展 Qt Designer 。

Qt：综合了上面描述的模块中的类到一个单一的模块中。

这样做的好处是你不用担心哪个模块包含哪个特定的类，坏处是加载进了整个Qt框架，从而增加了应用程序的内存占用。

uic：包含的类用来处理.ui文件，该文件由Qt Designer创建，用于描述整个或者部分用户界面。

它包含的加载.ui文件和直接渲染以及从.ui文件生成Python代码为以后执行的类。

先记录简单的，以下代码可显示一个基本的window：

[复制代码](javascript:void(0);)

#!/usr/bin/python

# -\*- coding: utf-8 -\*-

import sys

from PyQt4 import QtGui

def main():

app = QtGui.QApplication(sys.argv)

w = QtGui.QWidget()

w.resize(250, 150)

w.move(300, 300)

w.setWindowTitle('Simple')

w.show()

sys.exit(app.exec\_())

if \_\_name\_\_ == '\_\_main\_\_':

main()

[复制代码](javascript:void(0);)

from PyQt4 import QtGui，基本的GUI组件都存放在QtGui这个模块  
  
app = QtGui.QApplication(sys.argv)，每个Pyqt的程序都必须创建一个application对象，application在 QtGui 模块中，sys.argv 参数是命令行中的一组参数。  
  
w = QtGui.QWidget()，创建窗口对象，QWidget窗口组件是PyQt4中所有用户界面对象的基类，这里直接使用QWidget默认的构造。  
  
app.exec\_()其实就是QApplication的方法，原来这个exec\_()方法的作用是“进入程序的主循环直到exit()被调用”，如果没有这个方法，运行的时候窗口会闪退，所以show是有发挥作用的，但没有使用exec\_()，所以没有进入程序的主循环就直接结束了。于是修改代码：

[复制代码](javascript:void(0);)

#!/usr/bin/python

# -\*- coding: utf-8 -\*-

import sys

from PyQt4 import QtGui

def main():

app = QtGui.QApplication(sys.argv)

w = QtGui.QWidget()

w.resize(250, 150)

w.move(300, 300)

w.setWindowTitle('Simple')

w.show()

app.exec\_()

if \_\_name\_\_ == '\_\_main\_\_':

main()

[复制代码](javascript:void(0);)

不用sys.exit(app.exec\_())，只使用app.exec\_()，程序一样可以正常运行，但是关闭窗口后进程却不会退出，尝试print输出app.exec\_()的结果，返回0，于是再做修改：

[复制代码](javascript:void(0);)

#!/usr/bin/python

# -\*- coding: utf-8 -\*-

import sys

from PyQt4 import QtGui

def main():

app = QtGui.QApplication(sys.argv)

w = QtGui.QWidget()

w.resize(250, 150)

w.move(300, 300)

w.setWindowTitle('Simple')

w.show()

app.exec\_()

sys.exit(0)

if \_\_name\_\_ == '\_\_main\_\_':

main()

[复制代码](javascript:void(0);)

这样的话，运行时打开窗口也正常，退出时进程也可以结束了。所以才可以这么用sys.exit(app.exec\_())。  
增加图标：

[复制代码](javascript:void(0);)

#!/usr/bin/python

# -\*- coding: utf-8 -\*-

import sys

from PyQt4 import QtGui

class Icon(QtGui.QWidget):

def \_\_init\_\_(self, parent=None):

QtGui.QWidget.\_\_init\_\_(self, parent)

self.setGeometry(300, 300, 250, 150)

# 设置初始位置

self.setWindowTitle('Icon')

# window标题

self.setWindowIcon(QtGui.QIcon('icons/web.png'))

# 图标文件位置

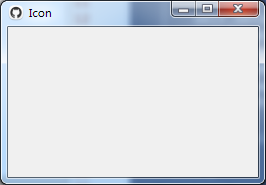
app = QtGui.QApplication(sys.argv)

icon = Icon()

icon.show()

sys.exit(app.exec\_())

[复制代码](javascript:void(0);)

　　图标显示效果  
移动到视窗范围内触发提示信息：

[复制代码](javascript:void(0);)

#!/usr/bin/python

# -\*- coding: utf-8 -\*-

import sys

from PyQt4 import QtGui

class Tooltip(QtGui.QWidget):

def \_\_init\_\_(self, parent=None):

QtGui.QWidget.\_\_init\_\_(self, parent)

self.setGeometry(300, 300, 250, 150)

self.setWindowTitle('Tooltip')

self.setToolTip('This is a <b>QWidget</b> widget')

# 设置提示信息

QtGui.QToolTip.setFont(QtGui.QFont('OldEnglish', 10))

# 显示字体，在MAC下无效

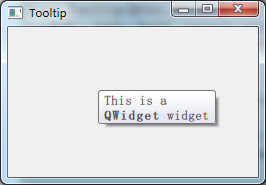
app = QtGui.QApplication(sys.argv)

tooltip = Tooltip()

tooltip.show()

sys.exit(app.exec\_())

[复制代码](javascript:void(0);)

　　提示效果  
退出时弹出确认消息框：

[复制代码](javascript:void(0);)

#!/usr/bin/python

# -\*- coding: utf-8 -\*-

import sys

from PyQt4 import QtGui

class MessageBox(QtGui.QWidget):

def \_\_init\_\_(self, parent=None):

QtGui.QWidget.\_\_init\_\_(self, parent)

self.setGeometry(300, 300, 250, 150)

self.setWindowTitle('message box')

def closeEvent(self, event):

# 关闭窗口的事件触发消息框询问，并设置消息框标题，提示信息，选择按键

reply = QtGui.QMessageBox.question(self, 'Message',

"Are you sure to quit?", QtGui.QMessageBox.Yes, QtGui.QMessageBox.No)

if reply == QtGui.QMessageBox.Yes:

event.accept()

else:

event.ignore()

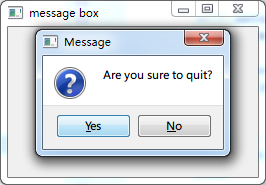
app = QtGui.QApplication(sys.argv)

qb = MessageBox()

qb.show()

sys.exit(app.exec\_())

[复制代码](javascript:void(0);)

　　提示效果

设置窗口居中显示：

[复制代码](javascript:void(0);)

#!/usr/bin/python

# -\*- coding: utf-8 -\*-

import sys

from PyQt4 import QtGui

class Center(QtGui.QWidget):

def \_\_init\_\_(self, parent=None):

QtGui.QWidget.\_\_init\_\_(self, parent)

self.setWindowTitle('center')

self.resize(250, 150)

# 重新设置大小

self.center()

def center(self):

screen = QtGui.QDesktopWidget().screenGeometry()

# 获取屏幕分辨率

size = self.geometry()

# 获取组件大小

self.move((screen.width()-size.width())/2, (screen.height()-size.height())/2)

app = QtGui.QApplication(sys.argv)

qb = Center()

qb.show()

sys.exit(app.exec\_())

[复制代码](javascript:void(0);)

 文档中简单的使用范例，添加注释后做个记录

# PyQt中弹出对话框,获取对话框返回值

http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif (2013-11-12 10:16:36)

[[http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif](javascript:;)转载▼](javascript:;)

|  |  |
| --- | --- |
| 标签： [子窗口](http://search.sina.com.cn/?c=blog&q=%D7%D3%B4%B0%BF%DA&by=tag)  [return](http://search.sina.com.cn/?c=blog&q=return&by=tag)  [value](http://search.sina.com.cn/?c=blog&q=value&by=tag)  [it](http://search.sina.com.cn/?c=blog&q=it&by=tag) | 分类： [pyqt4](http://blog.sina.com.cn/s/articlelist_2608384273_4_1.html) |

转载自<http://jimmykuu.sinaapp.com/blog/9>

经常有初学者搞不清楚如何在PyQt中弹出对话框，以及如何处理返回值。这篇文章会举例说明，界面采用手工编写。

我们一般说的对话框指的是模态对话框（Modal Dialogue Box），一旦弹出，就不能对话框以外的窗口进行操作，必须先关闭对话框。

在PyQt中我们一般从QDialog继承创建一个类来操作，根据exec\_()方法的返回值判断用户是【确定】还是【取消】了，当然也可以其他返回值，具体看文档。

这个例子创建一个主窗口，有一个表格，记录用户姓名和年龄，一个【添加】按钮，点击弹出对话框，用户输入姓名和年龄，点击【确定】返回，在主窗体表格中插入一行数据。

## 创建主窗体

为了方便起见使用QWdiget创建主窗体，当然你可以使用QMainWindow，用QTableView和QStandardItemModel来创建表格。

1. class MainWindow(QtGui.QWidget):
2. def \_\_init\_\_(self, parent=None):
3. QtGui.QMainWindow.\_\_init\_\_(self, parent)
5. # 创建table和model
6. table = QtGui.QTableView(parent=self)
7. self.model = QtGui.QStandardItemModel(parent=self)
8. self.model.setHorizontalHeaderLabels((u'姓名', u'年龄'))
9. table.setModel(self.model)
11. # 创建添加按钮
12. button = QtGui.QPushButton(u'添加', parent=self)
14. # 添加信号槽
15. button.clicked.connect(self.add)
17. # 创建一个垂直布局，用于防止表格和按钮
18. layout = QtGui.QVBoxLayout()
19. layout.addWidget(table)
20. layout.addWidget(button)
22. self.setLayout(layout)
24. def add(self):
25. pass

## 创建对话框

对话框从QDialog继承，按钮这里使用QButtonBox来创建，用QButtonBox的好处是创建方便，定义参数即可，并且会自动根据不同平台显示按钮的位置，和各平台风格保持一致，当然默认是英文的，你可以通过国际化来做成中文的。

这里没有做对话框内容的验证，你可以覆盖QDialog的accept方法来进行验证。

下面是对话框的创建代码，为了方便获取姓名和年龄变量，我写了两个方法供外部调用。

1. class Dialog(QtGui.QDialog):
2. def \_\_init\_\_(self, parent=None):
3. QtGui.QDialog.\_\_init\_\_(self, parent)
4. self.resize(240, 200)
6. # 表格布局，用来布局QLabel和QLineEdit及QSpinBox
7. grid = QtGui.QGridLayout()
9. grid.addWidget(QtGui.QLabel(u'姓名', parent=self), 0, 0, 1, 1)
11. self.leName = QtGui.QLineEdit(parent=self)
12. grid.addWidget(self.leName, 0, 1, 1, 1)
14. grid.addWidget(QtGui.QLabel(u'年龄', parent=self), 1, 0, 1, 1)
16. self.sbAge = QtGui.QSpinBox(parent=self)
17. grid.addWidget(self.sbAge, 1, 1, 1, 1)
19. # 创建ButtonBox，用户确定和取消
20. buttonBox = QtGui.QDialogButtonBox(parent=self)
21. buttonBox.setOrientation(QtCore.Qt.Horizontal) # 设置为水平方向
22. buttonBox.setStandardButtons(QtGui.QDialogButtonBox.Cancel|QtGui.QDialogButtonBox.Ok)# 确定和取消两个按钮
23. # 连接信号和槽
24. buttonBox.accepted.connect(self.accept) # 确定
25. buttonBox.rejected.connect(self.reject) # 取消
27. # 垂直布局，布局表格及按钮
28. layout = QtGui.QVBoxLayout()
30. # 加入前面创建的表格布局
31. layout.addLayout(grid)
33. # 放一个间隔对象美化布局
34. spacerItem = QtGui.QSpacerItem(20, 48, QtGui.QSizePolicy.Minimum,QtGui.QSizePolicy.Expanding)
35. layout.addItem(spacerItem)
37. # ButtonBox
38. layout.addWidget(buttonBox)
40. self.setLayout(layout)
42. def name(self):
43. return self.leName.text()
45. def age(self):
46. return self.sbAge.value()

## 编写对话框调用代码

调用对话框只要使用exec\_方法即可，它会弹出对话框并根据用户操作返回值，根据返回值判断是【确定】还是【取消】。

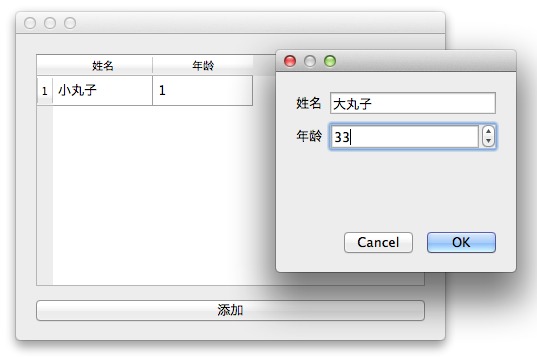
1. dialog = Dialog(parent=self)
2. if dialog.exec\_():
3. self.model.appendRow((
4. QtGui.QStandardItem(dialog.name()),
5. QtGui.QStandardItem(str(dialog.age())),
6. ))
8. dialog.destroy()

## 完整代码和截图

1. # -\*- coding: utf-8 -\*-
2. from PyQt4 import QtGui, QtCore
4. class MainWindow(QtGui.QWidget):
5. def \_\_init\_\_(self, parent=None):
6. QtGui.QMainWindow.\_\_init\_\_(self, parent)
8. # 创建table和model
9. table = QtGui.QTableView(parent=self)
10. self.model = QtGui.QStandardItemModel(parent=self)
11. self.model.setHorizontalHeaderLabels((u'姓名', u'年龄'))
12. table.setModel(self.model)
14. # 创建添加按钮
15. button = QtGui.QPushButton(u'添加', parent=self)
17. # 添加信号槽
18. button.clicked.connect(self.add)
20. # 创建一个垂直布局，用于防止表格和按钮
21. layout = QtGui.QVBoxLayout()
22. layout.addWidget(table)
23. layout.addWidget(button)
25. self.setLayout(layout)
27. def add(self):
28. dialog = Dialog(parent=self) #实例化
29. if dialog.exec\_():
30. self.model.appendRow((
31. QtGui.QStandardItem(dialog.name()),#得到值
32. QtGui.QStandardItem(str(dialog.age())),
33. ))
35. dialog.destroy()

38. class Dialog(QtGui.QDialog):
39. def \_\_init\_\_(self, parent=None):
40. QtGui.QDialog.\_\_init\_\_(self, parent)
41. self.resize(240, 200)
43. # 表格布局，用来布局QLabel和QLineEdit及QSpinBox
44. grid = QtGui.QGridLayout()
46. grid.addWidget(QtGui.QLabel(u'姓名', parent=self), 0, 0, 1, 1)
48. self.leName = QtGui.QLineEdit(parent=self)
49. grid.addWidget(self.leName, 0, 1, 1, 1)
51. grid.addWidget(QtGui.QLabel(u'年龄', parent=self), 1, 0, 1, 1)
53. self.sbAge = QtGui.QSpinBox(parent=self)
54. grid.addWidget(self.sbAge, 1, 1, 1, 1)
56. # 创建ButtonBox，用户确定和取消
57. buttonBox = QtGui.QDialogButtonBox(parent=self)
58. buttonBox.setOrientation(QtCore.Qt.Horizontal) # 设置为水平方向
59. buttonBox.setStandardButtons(QtGui.QDialogButtonBox.Cancel|QtGui.QDialogButtonBox.Ok)# 确定和取消两个按钮
60. # 连接信号和槽
61. buttonBox.accepted.connect(self.accept) # 确定
62. buttonBox.rejected.connect(self.reject) # 取消
64. # 垂直布局，布局表格及按钮
65. layout = QtGui.QVBoxLayout()
67. # 加入前面创建的表格布局
68. layout.addLayout(grid)
70. # 放一个间隔对象美化布局
71. spacerItem = QtGui.QSpacerItem(20, 48, QtGui.QSizePolicy.Minimum,QtGui.QSizePolicy.Expanding)
72. layout.addItem(spacerItem)
74. # ButtonBox
75. layout.addWidget(buttonBox)
77. self.setLayout(layout)
79. def name(self):
80. return self.leName.text()
81. #以函数返回值的形式,这样可以在主窗口中被使用值
83. def age(self):
84. return self.sbAge.value()

87. if \_\_name\_\_ == '\_\_main\_\_':
88. import sys
89. app = QtGui.QApplication(sys.argv)
90. mainWindow = MainWindow()
91. mainWindow.show()
92. sys.exit(app.exec\_())



# [Pyqt5.5 for Python3.4.3 学习笔记]-->QTableView表格视图控件的使用方法

来源:[赵克立](http://www.zhaokeli.com/) 分类:[Python](https://www.zhaokeli.com/cat/python.html) 标签：[Python](https://www.zhaokeli.com/tag/272.html)[PyQt5](https://www.zhaokeli.com/tag/273.html)发布时间:2016-06-06 21:34:57浏览:3852

**版权声明：**

本文为博主原创文章，转载请声明原文链接...谢谢。o\_0。

**原文链接:**

<https://www.zhaokeli.com/article/7986.html>

**更新时间：**

2016-08-07 23:27:03

qtableview这个控件可以绑定一个模型数据用来更新控件上的内容

可用的模式有以下几个

// QStringListModel          存储一组字符串

// QStandardItemModel        存储任意层次结构的数据

// QDirModel              对文件系统进行封装

// QSqlQueryModel           对SQL的查询结果集进行封装

// QSqlTableModel           对SQL中的table进行封装

// QSqlRelationalTableModel    对带有foreign key的SQL table进行封装

// QSortFilterProxyModel       对另一个model执行sort and/or filter

下面使用QStandardItemModel这个模型来给QTableView绑定数据

代码示例如下

[复制代码](javascript:;)



2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

from PyQt5.QtWidgets import \*

from PyQt5.QtGui import \*

from PyQt5.QtCore import \*

import sys

class myDialog(QDialog):

"""docstring for myDialog"""

def \_\_init\_\_(self, arg=None):

super(myDialog, self).\_\_init\_\_(arg)

self.setWindowTitle("first window")

self.setWindowFlags(Qt.WindowMaximizeButtonHint|Qt.WindowMinimizeButtonHint|Qt.WindowCloseButtonHint)

self.resize(500,300);

self.model=QStandardItemModel(4,4);

self.model.setHorizontalHeaderLabels(['标题1','标题2','标题3','标题4'])

for row in range(4):

for column in range(4):

item = QStandardItem("row %s, column %s"%(row,column))

self.model.setItem(row, column, item)

self.tableView=QTableView();

self.tableView.setModel(self.model)

#下面代码让表格100填满窗口

#self.tableView.horizontalHeader().setStretchLastSection(True)

#self.tableView.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)

dlgLayout=QVBoxLayout();

dlgLayout.addWidget(self.tableView)

self.setLayout(dlgLayout)

app = QApplication(sys.argv)

#全局设置QPushButton的背景样式

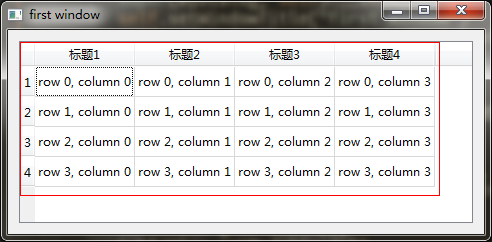
dlg = myDialog()

dlg.show()

dlg.exec\_()

app.exit()

效果如图：



上面的表格并没有填满窗口但是每列可以自由拉动,但是可能会出现滚动条。

有时候我们需要表格100%填满窗口可以添加下面代码

[复制代码](javascript:;)



1

2

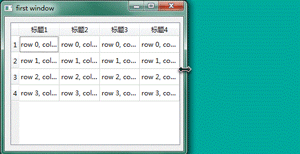
3

#下面代码让表格100填满窗口

self.tableView.horizontalHeader().setStretchLastSection(True)

self.tableView.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)

效果图：



下面对列表进行添加和删除

添加数据

[复制代码](javascript:;)



1

2

3

4

5

6

self.model.appendRow([

QStandardItem("row %s, column %s"%(11,11)),

QStandardItem("row %s, column %s"%(11,11)),

QStandardItem("row %s, column %s"%(11,11)),

QStandardItem("row %s, column %s"%(11,11)),

])

删除当前选中数据第一种方法

[复制代码](javascript:;)



1

2

3

4

5

6

#取当前选中的所有行

indexs=self.tableView.selectionModel().selection().indexes()

if len(indexs)>0:

#取第一行

index = indexs[0]

self.model.removeRows(index.row(),1)

第二种方法

[复制代码](javascript:;)



1

2

3

index=self.tableView.currentIndex()

print(index.row())

self.model.removeRow(index.row())

什么也不选的时候删除第一行也就是索引为0

选中一行的时候就删除选中的那一行

选中多行时就删除最后一行(国为焦点在最后一行)

# 转载：QT QTableView用法小结

时间：2015-09-08 15:17:59      阅读：230      评论：0      收藏：0      [点我收藏+]

标签：[des](http://www.bubuko.com/so/1/des)   [class](http://www.bubuko.com/so/1/class)   [style](http://www.bubuko.com/so/1/style)   [log](http://www.bubuko.com/so/1/log)   [com](http://www.bubuko.com/so/1/com)   [http](http://www.bubuko.com/so/1/http)   [使用](http://www.bubuko.com/so/1/%e4%bd%bf%e7%94%a8)   [si](http://www.bubuko.com/so/1/si)   [it](http://www.bubuko.com/so/1/it)

出自：

　　http://blog.chinaunix.net/uid-20382483-id-3518513.html

QTableView常用于实现数据的表格显示。下面我们如何按步骤实现学生信息表格：

**一 添加表头**

    //准备数据模型  
    QStandardItemModel \*student\_model = new QStandardItemModel();  
    student\_model->setHorizontalHeaderItem(0, new QStandardItem(QObject::tr("Name")));  
    student\_model->setHorizontalHeaderItem(1, new QStandardItem(QObject::tr("NO.")));  
    student\_model->setHorizontalHeaderItem(2, new QStandardItem(QObject::tr("Sex")));  
    student\_model->setHorizontalHeaderItem(3, new QStandardItem(QObject::tr("Age")));  
    student\_model->setHorizontalHeaderItem(4, new QStandardItem(QObject::tr("College")));  
    //利用setModel()方法将数据模型与QTableView绑定  
    ui->student\_tableview->setModel(student\_model);

**二 设置表格属性**

    //设置列宽不可变动，即不能通过鼠标拖动增加列宽          
    ui->student\_tableview->horizontalHeader()->setResizeMode(0, QHeaderView::Fixed);     
    ui->student\_tableview->horizontalHeader()->setResizeMode(1, QHeaderView::Fixed);     
    ui->student\_tableview->horizontalHeader()->setResizeMode(2, QHeaderView::Fixed);     
    ui->student\_tableview->horizontalHeader()->setResizeMode(3, QHeaderView::Fixed);     
    ui->student\_tableview->horizontalHeader()->setResizeMode(4, QHeaderView::Fixed);

    //设置表格的各列的宽度值          
    ui->student\_tableview->setColumnWidth(0,100);      
    ui->student\_tableview->setColumnWidth(1,100);      
    ui->student\_tableview->setColumnWidth(2,100);      
    ui->student\_tableview->setColumnWidth(3,100);      
    ui->student\_tableview->setColumnWidth(4,100);

    //默认显示行头，如果你觉得不美观的话，我们可以将隐藏          
    ui->student\_tableview->verticalHeader()->hide();

    //设置选中时为整行选中          
    ui->student\_tableview->setSelectionBehavior(QAbstractItemView::SelectRows);           
        
    //设置表格的单元为只读属性，即不能编辑          
    ui->student\_tableview->setEditTriggers(QAbstractItemView::NoEditTriggers);

    //如果你用在QTableView中使用右键菜单，需启用该属性          
    ui->tstudent\_tableview->setContextMenuPolicy(Qt::CustomContextMenu);

**三 动态添加行**

    在表格中添加行时，我们只需要在model中插入数据即可，一旦model中的数据发生变化，QTabelView显示就会做相应的变动

    //在第一行添加学生张三的个人信息(setItem函数的第一个参数表示行号，第二个表示列号，第三个为要显示的数据)  
    student\_model->setItem(0, 0, new QStandardItem(“张三"));  
    student\_model->setItem(0, 1, new QStandardItem("20120202"));  
    student\_model->setItem(0, 2, new QStandardItem("男"));  
    student\_model->setItem(0, 3, new QStandardItem("18"));  
    student\_model->setItem(0, 4, new QStandardItem("土木学院"));

**四 设置数据显示的样式**

    //设置单元格文本居中，张三的数据设置为居中显示  
    student\_model->item(0, 0)->setTextAlignment(Qt::AlignCenter);  
    student\_model->item(0, 1)->setTextAlignment(Qt::AlignCenter);  
    student\_model->item(0, 2)->setTextAlignment(Qt::AlignCenter);  
    student\_model->item(0, 3)->setTextAlignment(Qt::AlignCenter);  
    student\_model->item(0, 4)->setTextAlignment(Qt::AlignCenter);

    //设置单元格文本颜色，张三的数据设置为红色  
    student\_model->item(0, 0)->setForeground(QBrush(QColor(255, 0, 0)));   
    student\_model->item(0, 1)->setForeground(QBrush(QColor(255, 0, 0)));   
    student\_model->item(0, 2)->setForeground(QBrush(QColor(255, 0, 0)));   
    student\_model->item(0, 3)->setForeground(QBrush(QColor(255, 0, 0)));   
    student\_model->item(0, 4)->setForeground(QBrush(QColor(255, 0, 0)));

    //将字体加粗  
    student\_model->item(0, 0)->setFont( QFont( "Times", 10, QFont::Black ) );  
    student\_model->item(0, 1)->setFont( QFont( "Times", 10, QFont::Black ) );  
    student\_model->item(0, 2)->setFont( QFont( "Times", 10, QFont::Black ) );  
    student\_model->item(0, 3)->setFont( QFont( "Times", 10, QFont::Black ) );  
    student\_model->item(0, 4)->setFont( QFont( "Times", 10, QFont::Black ) );

    //设置排序方式，按年龄降序显示  
    student\_model->sort(3, Qt::DescendingOrder);

①、

设置选中单个元素、整行、整列。

ui->student\_tableview->setSelectionBehavior(QAbstractItemView::SelectRows);

### [enum QAbstractItemView::SelectionBehavior](http://doc.qt.io/qt-5/qabstractitemview.html#selectionBehavior-prop)

|  |  |  |
| --- | --- | --- |
| **Constant** | **Value** | **Description** |
| QAbstractItemView::SelectItems | 0 | Selecting single items. |
| QAbstractItemView::SelectRows | 1 | Selecting only rows. |
| QAbstractItemView::SelectColumns | 2 | Selecting only columns. |

### 

### 

②、设置是否允许多选：

ui->student\_tableview->setSelectionMode(QAbstractItemView::SingleSelection);

### [enum QAbstractItemView::SelectionMode](http://doc.qt.io/qt-5/qabstractitemview.html" \l "selectionMode-prop" \t "_blank)

This enum indicates how the view responds to user selections:

|  |  |  |
| --- | --- | --- |
| **Constant** | **Value** | **Description** |
| QAbstractItemView::SingleSelection | 1 | When the user selects an item, any already-selected item becomes unselected, and the user cannot unselect the selected item by clicking on it. |
| QAbstractItemView::ContiguousSelection | 4 | When the user selects an item in the usual way, the selection is cleared and the new item selected. However, if the user presses the Shift key while clicking on an item, all items between the current item and the clicked item are selected or unselected, depending on the state of the clicked item. |
| QAbstractItemView::ExtendedSelection | 3 | When the user selects an item in the usual way, the selection is cleared and the new item selected. However, if the user presses the Ctrl key when clicking on an item, the clicked item gets toggled and all other items are left untouched. If the user presses the Shift key while clicking on an item, all items between the current item and the clicked item are selected or unselected, depending on the state of the clicked item. Multiple items can be selected by dragging the mouse over them. |
| QAbstractItemView::MultiSelection | 2 | When the user selects an item in the usual way, the selection status of that item is toggled and the other items are left alone. Multiple items can be toggled by dragging the mouse over them. |
| QAbstractItemView::NoSelection | 0 | Items cannot be selected. |

The most commonly used modes are SingleSelection and ExtendedSelection.

③、设置表的单元格的编辑模式

ui->student\_tableview->setEditTriggers(QAbstractItemView.NoEditTriggers)

### [enum QAbstractItemView::EditTrigger](http://doc.qt.io/qt-5/qabstractitemview.html#EditTrigger-enum) flags QAbstractItemView::EditTriggers

This enum describes actions which will initiate item editing.

|  |  |  |
| --- | --- | --- |
| **Constant** | **Value** | **Description** |
| QAbstractItemView::NoEditTriggers | 0 | No editing possible. |
| QAbstractItemView::CurrentChanged | 1 | Editing start whenever current item changes. |
| QAbstractItemView::DoubleClicked | 2 | Editing starts when an item is double clicked. |
| QAbstractItemView::SelectedClicked | 4 | Editing starts when clicking on an already selected item. |
| QAbstractItemView::EditKeyPressed | 8 | Editing starts when the platform edit key has been pressed over an item. |
| QAbstractItemView::AnyKeyPressed | 16 | Editing starts when any key is pressed over an item. |
| QAbstractItemView::AllEditTriggers | 31 | Editing starts for all above actions. |

The EditTriggers type is a typedef for [QFlags](http://doc.qt.io/qt-5/qflags.html)<EditTrigger>. It stores an OR combination of EditTrigger values

# [QTableView是显示表格状的视图](http://blog.csdn.net/lixingworkccc/article/details/7994755)

标签： [qt](http://www.csdn.net/tag/qt)[path](http://www.csdn.net/tag/path)[file](http://www.csdn.net/tag/file)[c](http://www.csdn.net/tag/c)

2012-09-19 10:08 1874人阅读 [评论](http://blog.csdn.net/lixingworkccc/article/details/7994755#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/lixingworkccc/article/details/7994755#report)

其经常使用的函数有

1.显示格子 setShowGrid(false);

2.由内容调整列 resizeColumnsToContents();

3.获取垂直表头 verticalHeader();

当然还有水平表头  
4.设置非编辑状态   
verticalView->setEditTriggers(QTableView::NoEditTriggers);  
5.设置点击行为  
verticalView->setSelectionBehavior(QTableView::SelectRows);  
6.模型对象  
 QStandardItemModel \*model = new QStandardItemModel;   
 model->setColumnCount(4);  
 model->setHeaderData(0,Qt::Horizontal,tr("File"));  
 model->setHeaderData(1,Qt::Horizontal,tr("Path"));  
 model->setHeaderData(2,Qt::Horizontal,tr("Size"));  
 model->setHeaderData(3,Qt::Horizontal,tr("Duration"));  
设置显示模型为4列  
7.增加项到模型  
  QStandardItem\* item1 = new QStandardItem(tr("Yesterday Once More.mid"));   
 QStandardItem\* item2 = new QStandardItem(tr("C://Mid//"));  
 QStandardItem\* item3 = new QStandardItem(tr("0.001M"));  
 QStandardItem\* item4 = new QStandardItem(tr("3:48"));  
 QList<QStandardItem\*> item;  
 item << item1 << item2 << item3 << item4;  
 model->appendRow(item);  
8.设置模型到View  
setModel(model);

如果要隐藏某一列或行, 必须在setModel之后. 至于resizeColumnToContents是在隐藏之后调用还是在隐藏之前调用, 并不影响显示的效果.很简单的控制QTableView的列大小:初始化QTableView:    ui->tableView->horizontalHeader()->setDefaultSectionSize(110);    ui->tableView->horizontalHeader()->setMinimumSectionSize(110);在setModel(model);后调用ui->tableView->resizeColumnsToContents(); 这样会省很多事, 也简单美观.

# [Qt QTableview的用法详解](http://www.cnblogs.com/mwl523study/p/5478491.html)

## 

|  |
| --- |
| 本文转自http://www.360doc.com/content/14/0210/11/3300331\_351302235.shtml    一. 对QTableWidget本身的效果实现   1. 将表格变为禁止编辑  在默认情况下，表格里的字符是可以更改的，比如双击一个单元格，就可以修改原来的内容，如果想禁止用户的这种操作，让这个表格对用户只读，可以这样：  tableWidget->setEditTriggers(QAbstractItemView::NoEditTriggers);    QAbstractItemView.NoEditTriggers是QAbstractItemView.EditTrigger枚举中的一个，都是触发修改单元格内容的条件：  QAbstractItemView.NoEditTriggers  0  No editing possible. 不能对表格内容进行修改  QAbstractItemView.CurrentChanged  1  Editing start whenever current item changes.任何时候都能对单元格修改  QAbstractItemView.DoubleClicked  2  Editing starts when an item is double clicked.双击单元格  QAbstractItemView.SelectedClicked  4  Editing starts when clicking on an already selected item.单击已选中的内容  QAbstractItemView.EditKeyPressed  8  Editing starts when the platform edit key has been pressed over an item.  QAbstractItemView.AnyKeyPressed  16  Editing starts when any key is pressed over an item.按下任意键就能修改  QAbstractItemView.AllEditTriggers  31  Editing starts for all above actions.以上条件全包括      2. 设置表格为整行选择   tableWidget->setSelectionBehavior(QAbstractItemView::SelectRows); //整行选中的方式  QAbstractItemView.SelectionBehavior枚举还有如下类型  Constant  Value  Description  QAbstractItemView.SelectItems  0  Selecting single items.选中单个单元格  QAbstractItemView.SelectRows  1  Selecting only rows.选中一行  QAbstractItemView.SelectColumns  2  Selecting only columns.选中一列     3.单个选中和多个选中的设置：  tableWidget->setSelectionMode(QAbstractItemView::ExtendedSelection); //设置为可以选中多个目标  该函数的参数还可以是：  QAbstractItemView.NoSelection 不能选择  QAbstractItemView.SingleSelection 选中单个目标  QAbstractItemView.MultiSelection 选中多个目标  QAbstractItemView.ExtendedSelection QAbstractItemView.ContiguousSelection 的区别不明显，主要功能是正常情况下是单选，但按下Ctrl或Shift键后，可以多选    4. 表格表头的显示与隐藏  对于水平或垂直方法的表头，可以用以下方式进行 隐藏/显示 的设置： view plain tableWidget->verticalHeader()->setVisible(false); //隐藏列表头  tableWidget->horizontalHeader()->setVisible(false); //隐藏行表头   注意：需要 #include <QHeaderView>    5. 对表头文字的字体、颜色进行设置  view plain QTableWidgetItem \*columnHeaderItem0 = tableWidget->horizontalHeaderItem(0); //获得水平方向表头的Item对象 columnHeaderItem0->setFont(QFont("Helvetica")); //设置字体  columnHeaderItem0->setBackgroundColor(QColor(0,60,10)); //设置单元格背景颜色  columnHeaderItem0->setTextColor(QColor(200,111,30)); //设置文字颜色   注意：需要 #include <QHeaderView>    6. 在单元格里加入控件：  QTableWidget不仅允许把文字加到单元格，还允许把控件也放到单元格中。比如，把一个下拉框加入单元格，可以这么做： view plain QComboBox \*comBox = new QComboBox();  comBox->addItem("Y");  comBox->addItem("N");  tableWidget->setCellWidget(0,2,comBox);    二. 对单元格的进行设置   1. 单元格设置字体颜色和背景颜色 及字体字符 view plain QTableWidgetItem \*item = new QTableWidgetItem("Apple");  item->setBackgroundColor(QColor(0,60,10));  item->setTextColor(QColor(200,111,100));  item->setFont(QFont("Helvetica"));  tableWidget->setItem(0,3,item);   另：如果需要对所有的单元格都使用这种字体，则可以使用 tableWidget->setFont(QFont("Helvetica"));  2. 设置单元格内文字的对齐方式  这个比较简单，使用newItem.setTextAlignment()函数即可，该函数的参数为单元格内的对齐方式，和字符输入顺序是自左相右还是自右向左。  水平对齐方式有： Constant Value Description Qt.AlignLeft 0x0001 Aligns with the left edge. Qt.AlignRight 0x0002 Aligns with the right edge. Qt.AlignHCenter 0x0004 Centers horizontally in the available space. Qt.AlignJustify 0x0008 Justifies the text in the available space.    垂直对齐方式： Constant Value Description Qt.AlignTop 0x0020 Aligns with the top. Qt.AlignBottom 0x0040 Aligns with the bottom. Qt.AlignVCenter 0x0080 Centers vertically in the available space.   如果两种都要设置，只要用 Qt.AlignHCenter | Qt.AlignVCenter 的方式即可     3. 合并单元格效果的实现：  tableWidget->setSpan(0, 0, 3, 1) # 其参数为： 要改变单元格的 1行数 2列数 要合并的 3行数 4列数    4. 设置单元格的大小  首先，可以指定某个行或者列的大小 view plain tableWidget->setColumnWidth(3,200);  tableWidget->setRowHeight(3,60);   还可以将行和列的大小设为与内容相匹配 view plain tableWidget->resizeColumnsToContents();  tableWidget->resizeRowsToContents();    5. 获得单击单元格的内容   通过实现 itemClicked (QTableWidgetItem \*) 信号的槽函数，就可以获得鼠标单击到的单元格指针，进而获得其中的文字信息  connect(tableWidget,SIGNAL(itemDoubleClicked(QTreeWidgetItem\*,int)),this, SLOT( getItem(QTreeWidgetItem\*,int)) );  //将itemClicked信号与函数getItem绑定        1.QTableWidget不能在mainwindow中随主窗口的大小变化？    解决：在表格外部添加布局。    代码：tableWidget = new QTableWidget;           tableWidget ->setObjectName(QString::fromUtf8("tableWidget"));           QVBoxLayout \*verticalLayout;           verticalLayout->addWidget(tableWidget );  2.将表格变为禁止编辑：    tableWidget->setEditTriggers(QAbstractItemView::NoEditTriggers);    （参数含义：QAbstractItemView.NoEditTriggers--不能对表格内容进行修改                QAbstractItemView.CurrentChanged--任何时候都能对单元格修改                QAbstractItemView.DoubleClicked--双击单元格                QAbstractItemView.SelectedClicked--单击已选中的内容                QAbstractItemView.EditKeyPressed--                QAbstractItemView.AnyKeyPressed--按下任意键就能修改                QAbstractItemView.AllEditTriggers--以上条件全包括）  3.设置表格为整行选择    tableWidget->setSelectionBehavior(QAbstractItemView::SelectRows);  //整行选中的方式    （参数含义：AbstractItemView.SelectItems--选中单个单元格                QAbstractItemView.SelectRows--选中一行                QAbstractItemView.SelectColumns--选中一列）  4.单个选中和多个选中的设置：    tableWidget->setSelectionMode(QAbstractItemView::ExtendedSelection);  //设置为可以选中多个目标   （参数含义：QAbstractItemView.NoSelection--不能选择               QAbstractItemView.SingleSelection--选中单个目标               QAbstractItemView.MultiSelection--选中多个目标    QAbstractItemView.ExtendedSelection/QAbstractItemView.ContiguousSelection 的区别不明显，主要功能是正常情况下是单选，但按下Ctrl或Shift键后，可以多选）  5.表格表头的显示与隐藏    对于水平或垂直方法的表头，可以用以下方式进行 隐藏/显示 的设置：    tableWidget->verticalHeader()->setVisible(false);   //隐藏列表头    tableWidget->horizontalHeader()->setVisible(false); //隐藏行表头    注意：需要 #include <QHeaderView>  6.对表头文字的字体、颜色进行设置    QTableWidgetItem \*columnHeaderItem0 = tableWidget->horizontalHeaderItem(0); //获得水平方向表头的Item对象    columnHeaderItem0->setFont(QFont("Helvetica")); //设置字体    columnHeaderItem0->setBackgroundColor(QColor(0,60,10)); //设置单元格背景颜色    columnHeaderItem0->setTextColor(QColor(200,111,30)); //设置文字颜色    注意：需要 #include <QHeaderView>  7.在单元格里加入控件：     QComboBox \*comBox = new QComboBox();     comBox->addItem("Y");     comBox->addItem("N");     tableWidget->setCellWidget(0,2,comBox);  8.单元格中添加图片：    tableWidget->setItem(row, 0, new QTableWidgetItem(QIcon(":/new/images/kingdemo.ico"),tr("")));  9设置单元格字体颜色、背景颜色和字体字符：    QTableWidgetItem \*item = new QTableWidgetItem("Apple");    item->setBackgroundColor(QColor(0,60,10));    item->setTextColor(QColor(200,111,100));    item->setFont(QFont("Helvetica"));    tableWidget->setItem(0,3,item);    另：如果需要对所有的单元格都使用这种字体，则可以使用  tableWidget->setFont(QFont("Helvetica"));  10.设置单元格内文字的对齐方式   水平对齐方式有：   Constant Value Description   Qt.AlignLeft 0x0001 Aligns with the left edge.   Qt.AlignRight 0x0002 Aligns with the right edge.   Qt.AlignHCenter 0x0004 Centers horizontally in the available space.   Qt.AlignJustify 0x0008 Justifies the text in the available space.   垂直对齐方式：   Constant Value Description   Qt.AlignTop 0x0020 Aligns with the top.   Qt.AlignBottom 0x0040 Aligns with the bottom.   Qt.AlignVCenter 0x0080 Centers vertically in the available space.   如果两种都要设置，只要用 Qt.AlignHCenter |  Qt.AlignVCenter 的方式即可  11.合并单元格：    tableWidget->setSpan(0, 0, 3, 1)  # 其参数为： 要改变单元格的1行数、2列数，要合并的3行数、4列数  12.设置单元格的大小    首先，可以指定某个行或者列的大小     tableWidget->setColumnWidth(3,200);     tableWidget->setRowHeight(3,60);  还可以将行和列的大小设为与内容相匹配     tableWidget->resizeColumnsToContents();     tableWidget->resizeRowsToContents();  13.获得单击单元格的内容    通过实现 itemClicked (QTableWidgetItem \*) 信号的槽函数，就可以获得鼠标单击到的单元格指针，进而获得其中的文字信息  connect(tableWidget,SIGNAL(itemDoubleClicked(QTreeWidgetItem\*,int)),this,SLOT(getItem(QTreeWidgetItem\*,int)));  //将itemClicked信号与函数getItem绑定  14.QTableWidget要调整表格行宽主要涉及以下函数    tableWidget->horizontalHeader()->setResizeMode(QHeaderView::Stretch);//使列完全填充并平分    tableWidget->verticalHeader()->setResizeMode(QHeaderView::Stretch);//行自适应宽度    tableWidget->resizeColumnsToContents(); //根据内容调整列宽    tableWidget->resizeColumnToContents(int col);//根据内容自动调整给定列宽    tableWidget->horizontalHeader()->setResizeMode//把给定列设置为给定模式    主要模式有Stretch和Fixed  15.添加表头内容：    方法一：    QStringList header;    header<<""<<tr("1")<<tr("2")<<tr("3")<<tr("4)<<tr("5");    方法二：    tableWidget->setHorizontalHeaderLabels(QStringList() << tr("1")<<tr("2")<<tr("3")<<tr("4)<<tr("5"));  16.清除：    tableWidget->clear();//清除所有可见数据（包括表头），行还在    tableWidget->clearContents();//只清除表中数据，不清除表头内容    tableWidget->setRowCount(0)；//连行也清除掉  15.一些零碎的知识点代码：    int row = tableWidget->rowCount();//获取表格中当前总行数    tableWidget->setRowCount(row+1);//添加一行    tableWidget->removeRow(row);//清除已有的行列    Int row1 = tableWidget->currentItem()->row();//当前选中行    bool focus = tableWidget->isItemSelected(tableWidget->currentItem());//判断是否选中一行    QString proName = tableWidget->item(row, col)->text();//获取某一格内容    setShowGrid(true);//显示表格线    verticalHeader()->setVisible(false);//隐藏左边垂直    QHeaderView \*headerView = horizontalHeader();    headerView->setMovable(false);//去除表头的移动    headerView->resizeSection(0,284);//设置第一列宽    headerView->resizeSection(1,127);//设置第二列宽    headerView->setResizeMode(QHeaderView::Fixed);//列表不能移动    headerView->setClickable(false);//不响应鼠标单击    setEditTriggers(QTableWidget::NoEditTriggers);//不能编辑    setSelectionBehavior(QTableWidget::SelectRows);//一次选中一行    setSelectionMode(QAbstractItemView::SingleSelection);//只能单选    /\*QScrollBar \*scrollBar = horizontalScrollBar();    scrollBar->hide();\*/    setHorizontalScrollBarPolicy(Qt::ScrollBarAlwaysOff);//去掉水平滚动条    setVerticalScrollMode(QAbstractItemView::ScrollPerItem);//垂直滚动条按项移动    setAutoScroll(false);//去掉自动滚动  17.排序：    tableWidget->sortByColumn(0,Qt::AscendingOrder);//顾名思义，该函数意思是将某列按升序/降序的方式排列 |