<https://blog.csdn.net/dragon101788/article/details/77803901>

# Basic

import sys

from PyQt5.QtWidgets import QApplication, QWidget

1.Startup

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

app = QApplication(sys.argv)

#每一个PyQt5应用都必须创建一个应用对象.sys.argv参数是来自命令行的参数列表.Python脚本可以从shell里运行.这是我们如何控制我们的脚本运行的一种方法.

w = QWidget()

#QWidget窗口是PyQt5中所有用户界口对象的基本类.我们使用了QWidget默认的构造器.默认的构造器没有父类.一个没有父类的窗口被称为一个window.

w.resize(250, 150)

w.move(300, 300)

w.setWindowTitle('Simple')

w.show()

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

2.Class style（OOP）

import sys

from PyQt5.QtWidgets import QApplication, QWidget

from PyQt5.QtGui import QIcon

class Example(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.initUI()

def initUI(self):

self.setGeometry(300, 300, 300, 220)

self.setWindowTitle('Icon')

self.setWindowIcon(QIcon('web.png'))

self.show()

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

app = QApplication(sys.argv)

ex = Example()

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

## QPushButton

from PyQt5.QtWidgets import (QWidget, QToolTip,

QPushButton, QApplication)

from PyQt5.QtGui import QFont

QToolTip.setFont(QFont('SansSerif', 10)) #设置字体

self.setToolTip('This is a <b>QWidget</b> widget')#设置窗口提示信息

btn = QPushButton('Button', self) #设置按钮与按钮上信息

btn.setToolTip('This is a <b>QPushButton</b> widget') #设置按钮提示信息

from PyQt5.QtCore import QCoreApplication

qbtn.clicked.connect(QCoreApplication.instance().quit) #按钮信号发出

#在PyQt5中的事件处理系统是用信号和槽机制构建起来的.如果我们点击按钮,点击信号就会发出.槽可以是一个Qt槽或任何Python可调用的.QCoreApplication包含了主要的事件循环;它处理并分派所有事件.instance()方法给我们它当前的实例.注意QCoreApplication是用QApplication创建的.点击信息被连接到quit()方法,它会终止程序.通信是在发送者和接收者这两个对象之间完成的.这里的发送者是点击按钮,接收者是程序对象.

## closeEvent

from PyQt5.QtWidgets import QWidget, QMessageBox, QApplication

def closeEvent(self, event): #类方法重写

reply = QMessageBox.question # QMessageBox.question抛出对话框，

(self, 'Message', #对话框title

"Are you sure to quit?", #对话框内容

QMessageBox.Yes | QMessageBox.No, #对话框Button选项

QMessageBox.No) #关闭对话框的默认返回值

if reply == QMessageBox.Yes:

event.accept() #接受关闭信号

else:

event.ignore() #无视关闭信号

## Center display\*

import sys

from PyQt5.QtWidgets import QWidget, QDesktopWidget, QApplication

def initUI(self):

self.resize(250, 150)

self.center()

self.setWindowTitle('Center')

self.show()

def center(self):

qr = self.frameGeometry() #获取自身位置信息

cp = QDesktopWidget().availableGeometry().center() #计算屏幕信息

qr.moveCenter(cp) #计算如果放置中央的信息

self.move(qr.topLeft()) #将放置中央的信息中坐上坐标传给move方法

## Status Bar

from PyQt5.QtWidgets import QMainWindow, QApplication

def initUI(self):

self.statusBar().showMessage('Ready') #状态栏

## Menu

from PyQt5.QtWidgets import QMainWindow, QAction, qApp, QApplication

from PyQt5.QtGui import QIcon

class Example(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.initUI()

def initUI(self):

#配置一个菜单动作操作exitAction

exitAction = QAction(QIcon('exit.png'), '&Exit', self) #菜单动作操作配置

exitAction.setShortcut('Ctrl+Q') #菜单动作操作快捷键

exitAction.setStatusTip('Exit application') #鼠标指向时状态栏的提示信息

exitAction.triggered.connect(qApp.quit) #菜单动作操作连接回调

self.statusBar() #默认空状态栏

menubar = self.menuBar() #配置菜单

fileMenu = menubar.addMenu('&File') #添加一个类菜单

fileMenu.addAction(exitAction) #给菜单添加一个动作操作

self.setGeometry(300, 300, 300, 200)

self.setWindowTitle('Menubar')

self.show()

*对比：qbtn.clicked.connect(QCoreApplication.instance().quit) #按钮信号发出*

## Toolbar

def initUI(self):

exitAction = QAction(QIcon('exit24.png'), 'Exit', self)

exitAction.setShortcut('Ctrl+Q')

exitAction.triggered.connect(qApp.quit)

self.toolbar = self.addToolBar('Exit')

self.toolbar.addAction(exitAction)

self.setGeometry(300, 300, 300, 200)

self.setWindowTitle('Toolbar')

self.show()

## QtextEdit

textEdit = QTextEdit()

self.setCentralWidget(textEdit)

## keyPressEvent

def keyPressEvent(self, e):

if e.key() == Qt.Key\_Escape:

self.close()

## mousePressEvent

## buttonClicked

def initUI(self):

btn1 = QPushButton("Button 1", self)

btn1.move(30, 50)

btn2 = QPushButton("Button 2", self)

btn2.move(150, 50)

btn1.clicked.connect(self.buttonClicked)

btn2.clicked.connect(self.buttonClicked)

self.statusBar()

self.setGeometry(300, 300, 290, 150)

self.setWindowTitle('Event sender')

self.show()

def buttonClicked(self):

sender = self.sender()

self.statusBar().showMessage(sender.text() + ' was pressed')

## pyqtSignal

import sys

from PyQt5.QtCore import pyqtSignal, QObject

from PyQt5.QtWidgets import QMainWindow, QApplication

class Communicate(QObject):

closeApp = pyqtSignal() #信号创建\*

class Example(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.initUI()

def initUI(self):

self.c = Communicate()

self.c.closeApp.connect(self.close) #信号连接

self.setGeometry(300, 300, 290, 150)

self.setWindowTitle('Emit signal')

self.show()

def mousePressEvent(self, event):

self.c.closeApp.emit() #发送信号

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

app = QApplication(sys.argv)

ex = Example()

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