pyqt4 入门

万泽[1] | 编者[2]

版本: 1.0

[1]作者:
[2]编者: 邮箱: a358003542@gmail.com。

前言

本文的编译使用了 python 宏包，这个宏包新建了一个 python 环境，在这个环境中写入的代码将会下如 pdf 文档中。如果你的 py 文件写完了，那么你可以使用 dopython 命令来生成整个 py 文件并执行这个 py 文件。命令格式如下：

`\dopython[0或者1]{py文件名}`

可选项如果不填的话是默认 1 也就是生成 py 文件并执行。py 文件名的格式是比如你填入 test，那么将生成 test.py 文件。如果你填入可选项 0，那么只是进行常规完结 py 文件操作（dopython 命令在你新开一个 py 文件之前一定要有。），而不执行该 py 文件。

py 文件的执行是 python3，也就是所谓的 py 文件的编写按照 python3 格式来。

目 录

第一个例子

刚开始

文件：expample001.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#### 序言部分
import sys
from PyQt4  import QtGui
###############
app001 = QtGui.QApplication(sys.argv)

widget001 = QtGui.QWidget()
widget001.resize(800, 600)
widget001.setWindowTitle('第一个程序')
widget001.show()

sys.exit(app001.exec_())
```

前面的注释部分就不用说了，然后导入 sys，是为了后面接受 sys.argv 参数。导入 QtGui 是为了后面创建 QWidget 类的实例。

任何程序都需要创建一个 QApplication 类的实例，这里是 app001，后面跟著数字 001 就是为了强调这是一个实例。

然后接下来创建 QWidget 类的实例 widget001，首先是引用类的__init__ 方法，然后 QWidget 类里面有 resize 方法，这个方法调整等下生成的程序窗口的大小。而 setWindowTitle 方法设置等下程序窗口上面的标题。show 方法就是显示这个窗口。

后面我们看到系统要退出是调用的 app001 实例的 exec_ 方法，这一句还不太清楚。

加上图标

```
#!/usr/bin/env python3
#-*- coding: utf-8 -*-
```

现在我在前面第一个程序的基础上稍作修改，来给这个程序加上图标。为了模拟 Texmaker，程序的名字就叫做 Texmaker。

```
import sys
from PyQt4  import QtGui
#######################
class MyQWidget(QtGui.QWidget):
    def __init__(self,parent=None):
        QtGui.QWidget.__init__(self,parent)
```

```
7
8           self.setGeometry(0, 0, 800, 600)
9           # 坐标 0 0 大小 1360 768
10          self.setWindowTitle('Texmaker')
11          self.setWindowIcon(QtGui.QIcon\
12          ('icons/texmaker.ico'))
13
14  app001 = QtGui.QApplication(sys.argv)
15  widget001 = MyQWidget()
16  widget001.show()
17  sys.exit(app001.exec_())
```

因为自己的 DIY 开始变多了，所以这里新建了一个类，名字就简单叫做 MyQWidget，然后重新定义了这个类的初始函数。首先是继承自 QtGui.QWidget 类，然后延续了该类的初始函数，而 parent 被默认为 None。

然后用 QWidget 的 setGeometry 方法来调整窗口的左上顶点的坐标和窗口的 X，Y 的大小。这里 0，0 表示从屏幕的最左上点开始显示，同样 800，600 类似前面的 resize 函数的配置。

setWindowTitle 方法前面谈论过了，这里加入图标是通过 setWindowIcon 方法来做到的。这个方法调用了 QtGui.QIcon 方法，不管这么多，后面跟的就是图标的存放路径，使用相对路径。在运行这个例子的时候，请随便弄个图标文件过来。

后面的和前面类似就不多说了。

窗口弹出提示信息

```
#!/usr/bin/env python3
#-*- coding: utf-8 -*-
import sys
from PyQt4  import QtGui
#######################
```

接下来要做的 DIY 就是让这个窗口可以弹出提示信息，就是鼠标停置一会儿会弹出一段小文字。

```
class MyQWidget(QtGui.QWidget):
    def __init__(self,parent=None):
        QtGui.QWidget.__init__(self,parent)

        self.setGeometry(0, 0, 800, 600)
        # 坐标 0 0 大小 1360 768
        self.setWindowTitle('Texmaker')
        self.setWindowIcon(QtGui.QIcon\
        ('icons/texmaker.ico'))
        self.setToolTip('<b> 看什么看、、</b>')
        #<b></b> 加粗
        QtGui.QToolTip.setFont(QtGui.QFont\
        ('微软雅黑', 10))

app001 = QtGui.QApplication(sys.argv)
```

```
16 widget001 = MyQWidget()
17 widget001.show()
18 sys.exit(app001.exec_())
```

上面这段代码和前面的代码的不同就在于 MyQWidget 类的初始函数新加入了两条命令。其中 setToolTip 方法设置具体显示的文本内容，而 之间的文字会加粗。然后后面那条命令是设置字体和字号的，我不太清楚这里随便设置系统的字体微软雅黑是不是有效。

退出的时候询问

目前程序点击那个叉叉图标关闭程序的时候将会直接退出，这里新加入一个询问机制。

```
1 #!/usr/bin/env python3
2 #-*- coding: utf-8 -*-
3 import sys
4 from PyQt4  import QtGui
5 ########################
```

接下来要做的 DIY 就是让这个窗口可以弹出提示信息，就是鼠标停置一会儿会弹出一段小文字。

```
1 class MyQWidget(QtGui.QWidget):
2     def __init__(self,parent=None):
```

```
3         QtGui.QWidget.__init__(self,parent)
4
5         self.setGeometry(0, 0, 800, 600)
6         # 坐标 0 0 大小 1360 768
7         self.setWindowTitle('Texmaker')
8         self.setWindowIcon(QtGui.QIcon\
9         ('icons/texmaker.ico'))
10        self.setToolTip('<b> 看什么看、、</b>')
11        #<b></b> 加粗
12        QtGui.QToolTip.setFont(QtGui.QFont\
13        ('微软雅黑', 10))
14
15    def closeEvent(self, event):
16        # 重新定义 colseEvent
17        reply = QtGui.QMessageBox.question\
18        (self, '信息',
19            "你确定要退出吗？",
20             QtGui.QMessageBox.Yes,
21             QtGui.QMessageBox.No)
22
23        if reply == QtGui.QMessageBox.Yes:
24            event.accept()
25        else:
26            event.ignore()
27
28 app001 = QtGui.QApplication(sys.argv)
29 widget001 = MyQWidget()
```

```
30 widget001.show()
31 sys.exit(app001.exec_())
```

这段代码重新了原来的 colseEvent 方法，这里调用的那个方法内部“信息”两个字是弹出的信息框的标题，后面是信息框里面显示的文字。这里具体代码我还不是很懂。

居中显示窗体

```
1 #!/usr/bin/env python3
2 #-*- coding: utf-8 -*-
3 import sys
4 from PyQt4  import QtGui
5 ########################
```

接下来要做的 DIY 是让窗体弹出的时候居中显示，前面是设置了窗体的起点坐标的，这里新建了一个 center 方法来确认窗体居中显示。

```
1 class MyQWidget(QtGui.QWidget):
2     def __init__(self,parent=None):
3         QtGui.QWidget.__init__(self,parent)
4 
5         #self.setGeometry(0, 0, 800, 600)
6         # 坐标 0 0 大小 1360 768
7         self.resize(800,600)
```

```
8         self.center()
9         self.setWindowTitle('Texmaker')
10        self.setWindowIcon(QtGui.QIcon\
11        ('icons/texmaker.ico'))
12        self.setToolTip('<b> 看什么看、、</b>')
13        #<b></b> 加粗
14        QtGui.QToolTip.setFont(QtGui.QFont\
15        ('微软雅黑', 10))
16
17    def center(self):
18        screen = QtGui.QDesktopWidget().screenGeometry()
19        # 接受屏幕几何
20        size =  self.geometry()
21        self.move((screen.width()-size.width())/2,\
22         (screen.height()-size.height())/2)
23
24    def closeEvent(self, event):
25        # 重新定义 colseEvent
26        reply = QtGui.QMessageBox.question\
27        (self, '信息',
28            "你确定要退出吗？",
29             QtGui.QMessageBox.Yes,
30             QtGui.QMessageBox.No)
31
32        if reply == QtGui.QMessageBox.Yes:
33            event.accept()
34        else:
```

```
35             event.ignore()
36
37 app001 = QtGui.QApplication(sys.argv)
38 widget001 = MyQWidget()
39 widget001.show()
40 sys.exit(app001.exec_())
```

这里做的改动就是新建了一个 center 方法，接受实例。然后对这个实例也就是窗口的具体位置做一些调整。前面使用了 resize 和 center 两个方法来调整窗口的大小和窗口的位置。

从 center 方法中我们可以看到 move 方法的 X，Y 是从屏幕的坐标原点（0，0）开始计算的。第一个参数 X 表示向右移动了多少宽度，Y 表示向下移动了多少高度。