

# [笔记][LIKE-Python-GUI编程-PyQt5][12]

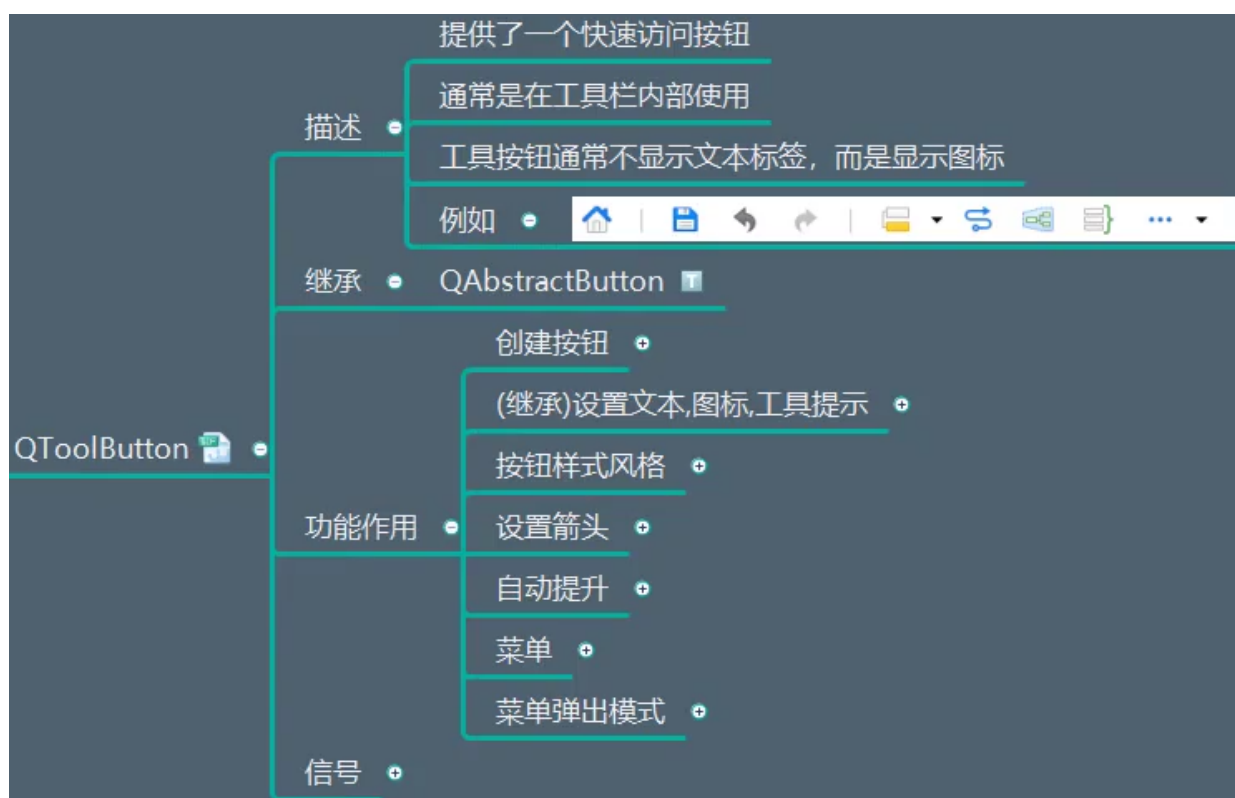
PyQt5

[笔记][LIKE-Python-GUI编程-PyQt5][12]

- 095. Python-GUI编程-PyQt5-QToolButton-创建和基本显示操作
- 096. Python-GUI编程-PyQt5-QToolButton-工具按钮样式设置
- 097. Python-GUI编程-PyQt5-QToolButton-箭头类型操作
- 098. Python-GUI编程-PyQt5-QToolButton-自动提升
- 099. Python-GUI编程-PyQt5-QToolButton-菜单和弹出模式
- 100. Python-GUI编程-PyQt5-QToolButton-可用信号

## 095. Python-GUI编程-PyQt5-QToolButton-创建和基本显示操作

工具按钮


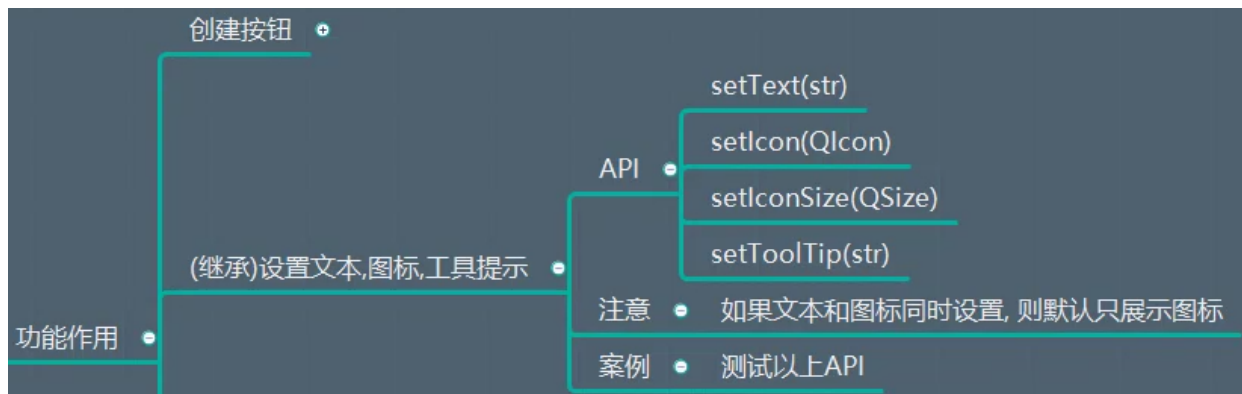


一个窗口从上到下：

- 标题栏
- 菜单栏
- 工具栏
  - 工具栏里的工具按钮，一般只显示图标，不显示文字
- 工作区域
- 状态栏

小技巧：PyCharm 智能提示后面的文字标注了该方法是属于哪一个类的。

```
# 设置文本
tb.setText
m setText(self, p_str) QAbstractButton
```

```
# 0. 导入需要的包和模块
import sys
from PyQt5.Qt import *

# 1. 创建一个应用程序对象
app = QApplication(sys.argv)

# 2. 控件的操作
# 2.1 创建控件
window = QWidget()
# 2.2 设置控件
window.setWindowTitle('QToolButton使用')
window.resize(500, 500)

# 构造函数 QToolButton(父控件)
tb = QToolButton(window)

# 设置文本
# 注意: 如果同时设置了文本和图标, 则不显示文本!
tb.setText('工具')

# 设置图标
tb.setIcon(QIcon('img/打开.png'))
tb.setIconSize(QSize(60, 60))

# 设置工具提示文本
# setToolTip 属于 QWidget
```

```
tb.setToolTip('打开文件')
```

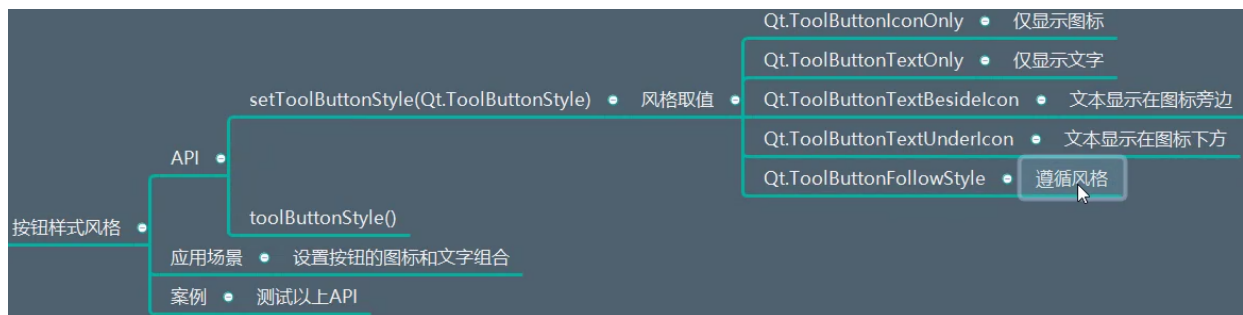
```
# 2.3 展示控件
```

```
window.show()
```

```
# 3. 应用程序的执行，进入到消息循环
```

```
sys.exit(app.exec_())
```

## 096. Python-GUI编程-PyQt5-QToolButton-工具按钮样式设置



```
# 0. 导入需要的包和模块
```

```
import sys
```

```
from PyQt5.Qt import *
```

```
# 1. 创建一个应用程序对象
```

```
app = QApplication(sys.argv)
```

```
# 2. 控件的操作
```

```
# 2.1 创建控件
```

```
window = QWidget()
```

```
# 2.2 设置控件
```

```
window.setWindowTitle('')
```

```
window.resize(500, 500)
```

```
tb = QToolButton(window)
```

```
tb.setText('打开')
```

```
tb.setIcon(QIcon('img/打开.png'))
```

```
tb.setIconSize(QSize(60, 60))
```

```
tb.setToolTip('打开文件')
```

```
# 设置工具按钮样式
```

```
# Qt.ToolButtonIconOnly 仅显示图标
```

```
# Qt.ToolButtonTextOnly 仅显示文字
```

```
# Qt.ToolButtonTextBesideIcon 文本显示在图标旁边
```

```
# Qt.ToolButtonTextUnderIcon 文本显示在图标下方
```

```

# Qt.ToolButtonFollowStyle 遵循风格
tb.setToolButtonStyle(Qt.ToolButtonIconOnly)
tb.setToolButtonStyle(Qt.ToolButtonTextOnly)
tb.setToolButtonStyle(Qt.ToolButtonTextBesideIcon) # 类似于普通的 QPushButton
tb.setToolButtonStyle(Qt.ToolButtonTextUnderIcon)
tb.setToolButtonStyle(Qt.ToolButtonFollowStyle)

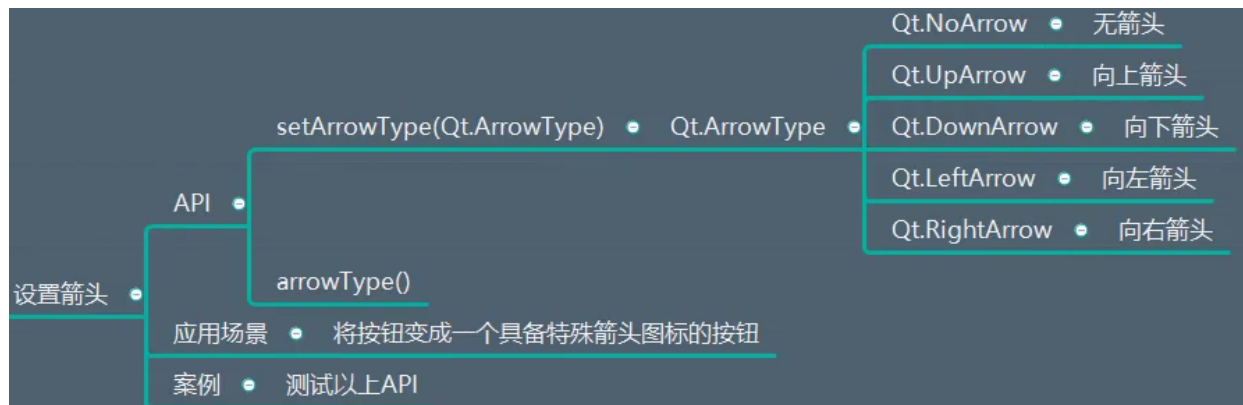
# 获取 toolButtonStyle()
# 注意返回的是数字
print('当前工具按钮样式: ', tb.toolButtonStyle())

# 2.3 展示控件
window.show()

# 3. 应用程序的执行, 进入到消息循环
sys.exit(app.exec_())

```

## 097. Python-GUI编程-PyQt5-QToolButton-箭头类型操作



```

# 0. 导入需要的包和模块
import sys
from PyQt5.Qt import *

# 1. 创建一个应用程序对象
app = QApplication(sys.argv)

# 2. 控件的操作
# 2.1 创建控件
window = QWidget()
# 2.2 设置控件
window.setWindowTitle('箭头类型')
window.resize(500, 500)

```

```

tb = QToolButton(window)

# Qt.NoArrow 无箭头
# Qt.UpArrow 向上箭头
# Qt.DownArrow 向下箭头
# Qt.LeftArrow 向左箭头
# Qt.RightArrow 向右箭头
tb.setArrowType(Qt.NoArrow)
tb.setArrowType(Qt.UpArrow)
tb.setArrowType(Qt.DownArrow)
tb.setArrowType(Qt.LeftArrow)
tb.setArrowType(Qt.RightArrow)

# 注意：如果设置了箭头和图标，箭头的优先级高
tb.setText('前进')
tb.setIcon(QIcon('img/Python.png'))
tb.setToolButtonStyle(Qt.ToolButtonTextBesideIcon)

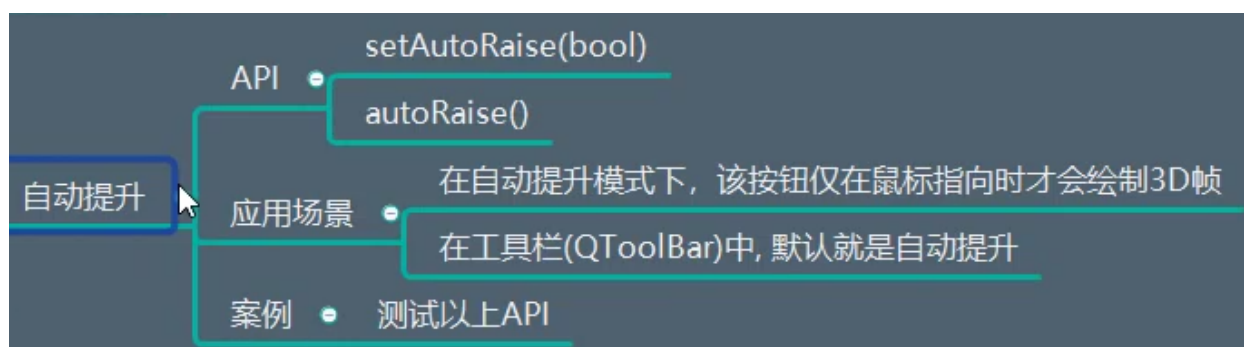
# 获取箭头类型
print('当前箭头类型：', tb.arrowType())

# 2.3 展示控件
window.show()

# 3. 应用程序的执行，进入到消息循环
sys.exit(app.exec_())

```

## 098. Python-GUI编程-PyQt5-QToolButton-自动提升



```

# 0. 导入需要的包和模块
import sys
from PyQt5.Qt import *

# 1. 创建一个应用程序对象

```



```
# 0. 导入需要的包和模块
import sys
from PyQt5.Qt import *

# 1. 创建一个应用程序对象
app = QApplication(sys.argv)

# 2. 控件的操作
# 2.1 创建控件
window = QWidget()
# 2.2 设置控件
window.setWindowTitle('工具按钮的菜单')
window.resize(500, 500)

btn = QPushButton(window)
btn.setText('一般按钮')
btn.move(100, 100)

# 创建菜单
menu = QMenu(btn)
# 添加子菜单
sub_menu = QMenu(menu)
sub_menu.setTitle('新建')
sub_menu.setIcon(QIcon('img/新建.png'))
menu.addMenu(sub_menu)
# 添加分割线
menu.addSeparator()
# 添加行为
action = QAction(QIcon('img/打开.png'), '打开', menu)
menu.addAction(action)
action.triggered.connect(lambda: print('点击了打开!'))

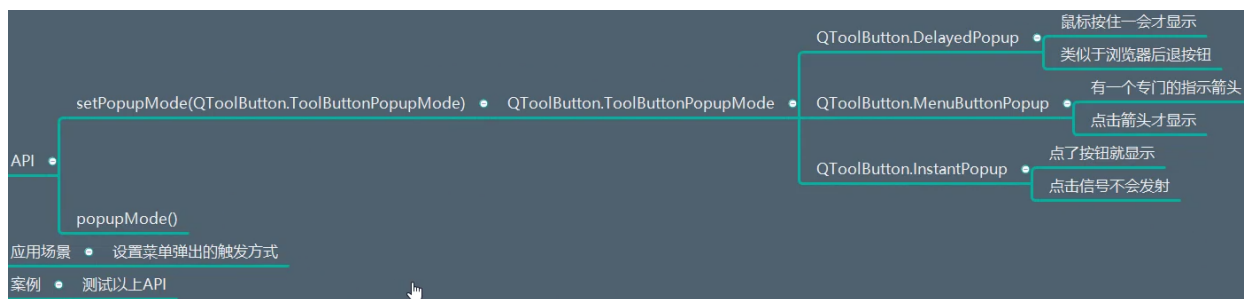
# 设置菜单
btn.setMenu(menu)

# 2.3 展示控件
window.show()

# 3. 应用程序的执行, 进入到消息循环
sys.exit(app.exec_())
```

---

## 工具按钮的菜单弹出模式



默认是按住停一会儿才会出来菜单，也就是 `QToolButton.DelayedPopup` 应用场景，比如浏览器中的长按后退显示历史记录



```
# 0. 导入需要的包和模块
import sys
from PyQt5.Qt import *

# 1. 创建一个应用程序对象
app = QApplication(sys.argv)

# 2. 控件的操作
# 2.1 创建控件
window = QWidget()
# 2.2 设置控件
window.setWindowTitle('菜单弹出')
window.resize(500, 500)

tb = QToolButton(window)
tb.setText('工具')
tb.setArrowType(Qt.RightArrow)
tb.setToolButtonStyle(Qt.ToolButtonTextBesideIcon)

# 创建菜单
menu = QMenu(tb)
# 添加子菜单
sub_menu = QMenu(menu)
sub_menu.setTitle('新建')
sub_menu.setIcon(QIcon('img/新建.png'))
menu.addMenu(sub_menu)
# 添加分割线
menu.addSeparator()
# 添加行为
action = QAction(QIcon('img/打开.png'), '打开', menu)
menu.addAction(action)
action.triggered.connect(lambda: print('点击了打开!'))

# 设置菜单
```



```

# setMenu 是 QToolButton 类的方法
tb.setMenu(menu)

# 设置菜单弹出方式
# 默认是延迟弹出: QToolButton.DelayedPopup
# 菜单按钮弹出: 点击右侧向下的箭头才可以弹出菜单
# tb.setPopupMode(QToolButton.MenuButtonPopup)
# 立即弹出: 点击整体按钮弹出菜单
tb.setPopupMode(QToolButton.InstantPopup)

# 默认延迟弹出 (QToolButton.DelayedPopup) 时, 长按点击再松开不会发送 clicked 信号
# 菜单按钮弹出 (QToolButton.MenuButtonPopup) 时, 点击左侧发送信号, 点击右侧向下箭头不发射
# 立即弹出 (QToolButton.InstantPopup) 时, 不管怎么点都不会发射 clicked 信号
tb.clicked.connect(lambda: print('工具按钮被点击了'))

# 2.3 展示控件
window.show()

# 3. 应用程序的执行, 进入到消息循环
sys.exit(app.exec_())

```

## 100. Python-GUI编程-PyQt5-QToolButton-可用信号



```

# 0. 导入需要的包和模块
import sys
from PyQt5.Qt import *

# 1. 创建一个应用程序对象
app = QApplication(sys.argv)

# 2. 控件的操作
# 2.1 创建控件
window = QWidget()
# 2.2 设置控件
window.setWindowTitle('菜单弹出')
window.resize(500, 500)

```

```

tb = QPushButton(window)
tb.setText('工具')
tb.setArrowType(Qt.RightArrow)
tb.setToolButtonStyle(Qt.ToolButtonTextBesideIcon)

# 创建菜单
menu = QMenu(tb)
# 添加子菜单
sub_menu = QMenu(menu)
sub_menu.setTitle('新建')
sub_menu.setIcon(QIcon('img/新建.png'))
menu.addMenu(sub_menu)
# 添加分割线
menu.addSeparator()
# 添加行为
open_action = QAction(QIcon('img/打开.png'), '打开', menu)
menu.addAction(open_action)
open_action.triggered.connect(lambda: print('点击了打开!'))
# 设置数据
open_action.setData([1, 2, 3])

# 再添加一个行为
exit_action = QAction(QIcon('img/退出.png'), '退出', menu)
menu.addAction(exit_action)
# 设置数据
exit_action.setData({'name': '进击的团子'})

# 设置菜单
# setMenu 是 QPushButton 类的方法
tb.setMenu(menu)

# 设置菜单弹出方式
# 默认是延迟弹出: QPushButton.DelayedPopup
# 菜单按钮弹出: 点击右侧向下的箭头才可以弹出菜单
# tb.setPopupMode(QPushButton.MenuButtonPopup)
# 立即弹出: 点击整体按钮弹出菜单
tb.setPopupMode(QPushButton.InstantPopup)

# 这里可以接受一个参数 QAction
def do_action(act):
    print('点击了行为!', act)
    # 给每个行为绑定不同的数据
    # 可以根据此数据做不同的操作
    print('绑定的数据是:', act.data())

# 打印了两句:
# 有一句是 QAction 自己发射的信号
# 另外一句是 QPushButton 发射的信号
tb.triggered.connect(do_action)
# 用 QPushButton 的信号优点: 可以统一汇总, 不用每个 QAction 都写一个槽函数

```

```
# 2.3 展示控件
```

```
window.show()
```

```
# 3. 应用程序的执行，进入到消息循环
```

```
sys.exit(app.exec_())
```

---

完成于 201810241558