PyQt 概観

山岸

2024 年 9 月 4 日

概 要

PyQt を触りはじめて、Fusion の API のように各 class の従属関係が複雑で、かつ、公式のリファレンス（量が多すぎる）ぐらいしか信頼できるドキュメントがないことに気が付いた。

分かったことを順次書いていってドキュメント化することが効率的なやり方だと考えたのでこの文書を作成する。各クラスの親子関係、呼び出し、追加の関係をまとめていく。

1 主要モジュール

まず、いくつかの class の集まりであり一つ上の概念であるモジュール（module）で主要なものを列挙する。

https://doc.qt.io/qt-6/qtmodules.html によれば主要モジュールは全部で 12 個あるそうだが、使いそうなのは Qt Core, Qt GUI, Qt Widgets の 3 つぐらいだろう。

1.1 Qt Core

QBasicTimer（読んで字のごとくタイマー機能）, QEvent（キーボード入力関連）, QFile（ファイルの読み込み、書き込み関連）などが入っている（重要なものを見つけたら適宜追加する）。QFile とか、python でやるうえでは不要では？ cpp とかでも他にやり方がありそうだし、まぁ謎。fileExist とかの関数は便利そう。

1.2 Qt GUI

Qt core の拡張（ここでいう拡張は恐らく継承の意味かも）それなのに、import する時には PyQt6.QtCore の下からではアクセスできないようになっている、不思議だなぁ。QActionEvent,QColor,QEnterEvent,QImage,QKeyEvent,QText ,iterator(この中に ParentFlame?妙だな),QFont などが入る。

1.3 Qt Widgets

本丸である。Event と Image 以外大体入ってそうだ。

2 QGridLayout

真の名-> PyQt6.QtWigets.QGridLayout() QHBoxLayout とか QVBoxLayout とかもあるが、幅の調整の自由度が低いので、Grid を使いたいところ。ウィンドウの縮小とかの処理が面倒かなという偏見がある。

2.1 method

addLayout() 子レイアウトを埋め込む。多分 QHBoxLayout とかでも使える。
addItem() item って何ぞ？
addWidget() 追加するやつ。QProgressBar() とか QGraphicsView() とか QTextEdit() が追加できる。

QProgressBar()	progress bar のインスタンス（widget class に属する）
QTextEdit()	テキスト入力（これも widget class）

3 QWidget

QWidget が、Qt が持つ全てのウィンドウを司るクラスの基になっているらしい。（へ〜）自作したいウィンドウクラスを作成する場合、継承元として指定するぐらいの使い道しかないと思われる。紛らわしいけど、真の名を PyQt6.QWidgets.QWidget()。type = <class 'PyQt6.sip.wrappertype'>

3.1 QWidget の子孫たち

こやつらはほとんど同じ扱いができるのでまとめておく。ここでいう同じ扱いとは、
・addWidget(instance,arg1,arg2) で GridLayout に追加できる
・hogehoge(発見したら追加)
である。

QGraphicsView()	QGraphicScene() を貼り付ける先（e.g. view.setScene(scene)）
QProgressBar()	progress bar のインスタンス
QTextEdit()	複数行入れられる入力マス
QLineEdit()	1 行のみの入力マス

4 QGraphicScene

QGraphicTextItem とか、QGraphicSimpleTextItem()addItem メソッドを使って貼り付けることができる。また、QtGUI.QPixmap を addPixmap メソッドを使って貼り付けることができる中間的なオブジェクトである。このオブジェクトを QGraphicsView() に貼り付けることで Widget を作成でき、さらにそのウィジットを addWidget メソッドでレイアウトに貼り付けることができる。

つまり、作成する順番としては（厳密には貼り付ける方向、なのかもしれないが）(QGraphicSimpleTextItem or QtGUI.QPixmap) -> QGraphicScene -> QGraphicsView -> Layout()

てな感じである。

Listing 1: sceneSample.py

```
1  import sys
2  from PyQt6.QtWidgets import(QWidget,
       QGraphicsTextItem,QGraphicsScene,QGraphicsView,
       QHBoxLayout,QApplication)
3
4  class Sample(QWidget):
5      def __init__(self) -> None:
6          super().__init__()
7          self.sampleUI()
8
9      def sampleUI(self)->None:
10         #create instance of QGraphicTextItem
11         txtboxA = QGraphicsTextItem()
12         txtboxB = QGraphicsTextItem()
13         txtboxA.setPlainText("A")
14         txtboxB.setPlainText("B")
15
16         #create instance of Scene
17         sceneA = QGraphicsScene()
18         sceneB = QGraphicsScene()
19
20         #create instance of Graphic
21         viewA = QGraphicsView()
22         viewB = QGraphicsView()
23
24         #貼る
```

```
25            sceneA.addItem(txtboxA)
26            sceneB.addItem(txtboxB)
27            viewA.setScene(sceneA)
28            viewB.setScene(sceneB)
29
30            layout = QHBoxLayout()
31            layout.addWidget(viewA)
32            layout.addWidget(viewB)
33
34            self.setLayout(layout)
35            self.setGeometry(300,300,350,300)
36            self.setWindowTitle('Review')
37            self.show()
38
39    if __name__ == '__main__':
40        app = QApplication(sys.argv)
41        smpl = Sample()
42        sys.exit(app.exec())
```

5 tips

5.1 How to turn Auto Completion On

PyQt6 を pip install しただけでは VSCode での予測変換が効かない。これではやってられない（あるオブジェクトに対してどの method や propaty が使えるかがわかりづらくなる）のでどうにかしよう。

”python.autoComplete.extraPaths”: [

”C: \\ Users \\ Appdata...path2PyQt6”

]

を setting.json の直下に書き込む