Pemrograman Desktop

Pertemuan 5 : Action, Toolbar and Menu

yoga@trunojoyo.ac.id

Pendahuluan

- Salah satu elemen antarmuka pengguna yang paling sering dilihat adalah toolbar.
- Toolbar adalah bilah ikon dan/atau teks yang digunakan untuk melakukan tugas-tugas umum dalam suatu aplikasi.
- Toolbar Qt mendukung tampilan ikon, teks, dan juga dapat berisi Qtwidget standar apa pun.
- Untuk tombol, pendekatan terbaik adalah dengan memanfaatkan sistem QAction untuk menempatkan tombol pada toolbar.

Contoh Aplikasi

```
import sys

from PyQt6.QtCore import Qt
from PyQt6.QtWidgets import (
    QApplication,
    QLabel,
    QMainWindow,
    QToolBar,
)

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")
        label = QLabel("Hello!")
        label.setAlignment(
          Qt.AlignmentFlag.AlignCenter)
        self.setCentralWidget(label)

        toolbar = QToolBar("My main toolbar")
        self.addToolBar(toolbar)

    def onMyToolBarButtonClick(self, s):
        print("click", s)


app = QApplication(sys.argv)

window = MainWindow()
window.show()

app.exec()
```

Pendahuluan

- QAction adalah kelas yang menyediakan cara untuk mendeskripsikan antarmuka pengguna abstrak.
- Artinya dalam bahasa Inggris adalah Anda dapat mendefinisikan beberapa elemen antarmuka dalam satu objek, disatukan oleh efek interaksi dengan elemen tersebut.
- Sebagai contoh, fungsi-fungsi yang umum terdapat pada toolbar dan juga menu.

Contoh Aplikasi

```
import sys

from PyQt6.QtCore import Qt
from PyQt6.QtGui import QAction
from PyQt6.QtWidgets import (
    QApplication,
    QLabel,
    QMainWindow,
    QToolBar,
)

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")
        label = QLabel("Hello!")
        label.setAlignment(
           Qt.AlignmentFlag.AlignCenter)
        self.setCentralWidget(label)

        toolbar = QToolBar("My main toolbar")
        self.addToolBar(toolbar)

        button_action = QAction(
            "Your button", self)
        button_action.setStatusTip(
            "This is your button")
        button_action.triggered.connect(
             self.onMyToolBarButtonClick)
        toolbar.addAction(button_action)

    def onMyToolBarButtonClick(self, s):
        print("click", s)

app = QApplication(sys.argv)

window = MainWindow()
window.show()

app.exec()
```

Pendahuluan

- Kita membuat fungsi yang akan menerima sinyal dari QAction sehingga dapat melihat apakah fungsi tersebut berjalan.
- Selanjutnya, mendefinisikan QAction itu sendiri.
- Saat membuat instance, kita dapat memberikan label untuk tindakan dan/atau ikon.
- Juga harus meneruskan QObject apa pun untuk bertindak sebagai induk aksi - di sini kita meneruskan self sebagai referensi ke jendela utama kita.

Pendahuluan

- Untuk QAction, elemen induk diteruskan sebagai parameter akhir.
- Selanjutnya, kita dapat memilih untuk menyetel tip status — teks ini akan ditampilkan pada bilah status setelah kita memilikinya.
- Kemudian menghubungkan sinyal .triggered ke fungsi kustom.
- Sinyal ini akan menyala setiap kali QAction 'dipicu' (atau diaktifkan).

Contoh Aplikasi

```
import sys

from PyQt6.QtCore import Qt
from PyQt6.QtGui import QAction
from PyQt6.QtWidgets import (
    QApplication,
    QLabel,
    QMainWindow,
    QStatusBar,
    QToolBar,
)
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")
        label = QLabel("Hello!")
        label.setAlignment(
              Qt.AlignmentFlag.AlignCenter)
        self.setCentralWidget(label)
        toolbar = QToolBar("My main toolbar")
        self.addToolBar(toolbar)
        button_action = QAction(
              "Your button", self)
        button_action.setStatusTip(
              "This is your button")
        button_action.triggered.connect(
              self.onMyToolBarButtonClick)
        toolbar.addAction(button_action)
        self.setStatusBar(QStatusBar(self))

    def onMyToolBarButtonClick(self, s):
        print("click", s)

app = QApplication(sys.argv)
window = MainWindow()
window.show()
app.exec()
```

Pendahuluan

- Selanjutnya kita akan mengaktifkan QAction — jadi mengklik akan mengaktifkannya, dan mengklik lagi akan mematikannya.
- Untuk melakukan ini, kita cukup memanggil setCheckable(True) pada objek QAction.

```
button_action.setCheckable(True)
```

QAction

- Kita dapat membuat objek QIcon dengan meneruskan jalur file ke kelas.
- Memuat ikon menggunakan teknik direktori dasar yang telah dipelajari di bab Widget. Ini memastikan file dapat ditemukan di mana pun ketika menjalankan skrip.
- Untuk menambahkan ikon ke QAction (dan juga tombolnya) perlu meneruskannya sebagai parameter pertama saat membuat QAction.
- Perlu memberi tahu toolbar seberapa besar ikon, jika tidak, ikon akan dikelilingi oleh banyak padding.
- Besaran ikon dapat dikonfigurasi dengan memanggil .setIconSize() dengan objek QSize.

Contoh Aplikasi

```
import os
import sys

from PyQt6.QtCore import QSize, Qt
from PyQt6.QtGui import QAction, QIcon
from PyQt6.QtWidgets import (
    QApplication,
    QLabel,
    QMainWindow,
    QStatusBar,
    QToolBar,
)

basedir = os.path.dirname(__file__)

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")
        label = QLabel("Hello!")
        label.setAlignment(
             Qt.AlignmentFlag.AlignCenter)
        self.setCentralWidget(label)
        toolbar = QToolBar("My main toolbar")
        toolbar.setIconSize(QSize(16, 16))
        self.addToolBar(toolbar)
        button_action = QAction(
            QIcon(os.path.join(
               basedir, "bug.png")),
            "Your button",
            self,
        )
        button_action.setStatusTip(
            "This is your button")
        button_action.triggered.connect(
            self.onMyToolBarButtonClick)
        button_action.setCheckable(True)
        toolbar.addAction(button_action)
        self.setStatusBar(QStatusBar(self))

    def onMyToolBarButtonClick(self, s):
        print("click", s)

app = QApplication(sys.argv)
window = MainWindow()
window.show()
app.exec()
```

QAction

- Qt menggunakan pengaturan default sistem operasi untuk menentukan apakah akan menampilkan ikon, teks, atau ikon dan teks di toolbar.
- Namun dapat menggantinya dengan menggunakan .setToolButtonStyle.
- Slot ini menerima salah satu tanda berikut dari Qt.namespace :

Flag	Behavior
Qt.ToolButtonStyle.ToolButtonIconOnly	Icon only, no text
Qt.ToolButtonStyle.ToolButtonTextOnly	Text only, no icon
Qt.ToolButtonStyle.ToolButtonTextBesideIcon	Icon and text, with text beside the icon
Qt.ToolButtonStyle.ToolButtonTextUnderIcon	Icon and text, with text under the icon
Qt.ToolButtonStyle.ToolButtonFollowStyle	Follow the host desktop style

Contoh Aplikasi

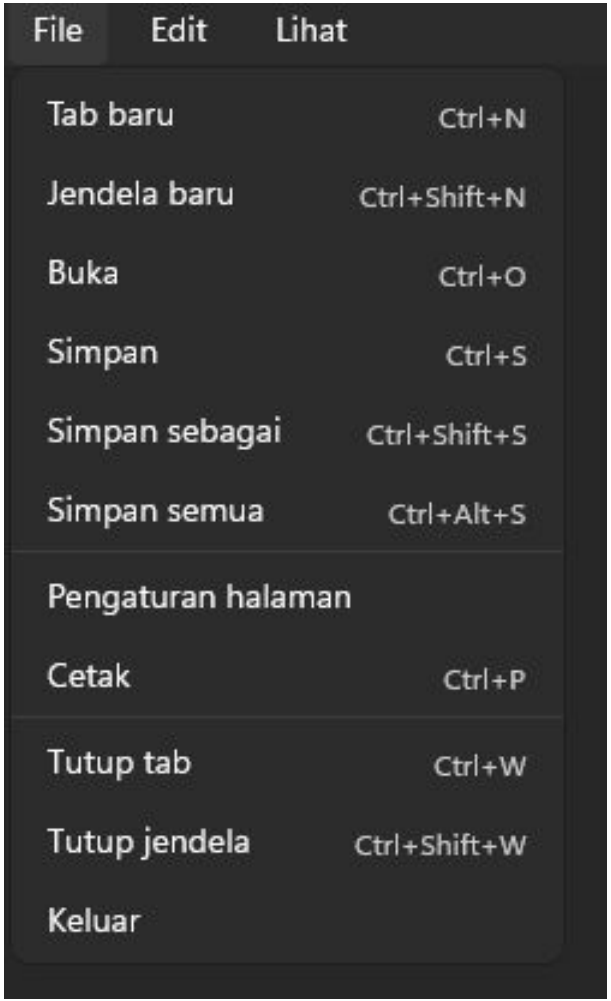
```
import os
import sys

from PyQt6.QtCore import QSize, Qt
from PyQt6.QtGui import QAction, QIcon
from PyQt6.QtWidgets import (
    QApplication,
    QCheckBox,
    QLabel,
    QMainWindow,
    QStatusBar,
    QToolBar,
)

basedir = os.path.dirname(__file__)

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")
        label = QLabel("Hello!")
        label.setAlignment(
              Qt.AlignmentFlag.AlignCenter)
        self.setCentralWidget(label)
        toolbar = QToolBar("My main toolbar")
        toolbar.setIconSize(QSize(16, 16))
        self.addToolBar(toolbar)

        button_action = QAction(
            QIcon(os.path.join(
                 basedir, "bug.png")),
            "Your button",
            self,
        )
        button_action.setStatusTip(
               "This is your button")
        button_action.triggered.connect(
               self.onMyToolBarButtonClick)
        button_action.setCheckable(True)
        toolbar.addAction(button_action)
        toolbar.addSeparator()
        button_action2 = QAction(
            QIcon(os.path.join(
                basedir, "bug.png")),
            "Your button2",
            self,
        )
```

Contoh Aplikasi

```
        button_action2.setStatusTip("This is your
button2")

button_action2.triggered.connect(self.onMyToolBarB
uttonClick)
        button_action2.setCheckable(True)
        toolbar.addAction(button_action2)

        toolbar.addWidget(QLabel("Hello"))
        toolbar.addWidget(QCheckBox())

        self.setStatusBar(QStatusBar(self))

    def onMyToolBarButtonClick(self, s):
        print("click", s)

app = QApplication(sys.argv)
window = MainWindow()
window.show()
app.exec()
```

Menu

- Menu adalah komponen standar UI lainnya.
- Mereka mengizinkan akses ke semua fungsi aplikasi standar.
- Ada beberapa menu standar - misalnya, File, Edit, dan Bantuan.
- Menu dapat disarangkan untuk membuat hierarki fungsi dan sering kali mendukung serta menampilkan pintasan keyboard untuk akses cepat ke fungsinya.



Contoh Aplikasi

```
import os
import sys

from PyQt6.QtCore import QSize, Qt
from PyQt6.QtGui import QAction, QIcon
from PyQt6.QtWidgets import (
    QApplication,
    QCheckBox,
    QLabel,
    QMainWindow,
    QStatusBar,
    QToolBar,
)

basedir = os.path.dirname(__file__)

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")
        label = QLabel("Hello!")
```

```
label.setAlignment(

   Qt.AlignmentFlag.AlignCenter)
 self.setCentralWidget(label)
 toolbar = QToolBar("My main toolbar")
 toolbar.setIconSize(QSize(16, 16))
 self.addToolBar(toolbar)
 button_action = QAction(
     QIcon(os.path.join(
              basedir, "bug.png")),
     "&Your button",
     self,
 )
 button_action.setStatusTip(
        "This is your button")
 button_action.triggered.connect(
        self.onMyToolBarButtonClick)
 button_action.setCheckable(True)
 toolbar.addAction(button_action)
 toolbar.addSeparator()
 button_action2 = QAction(
     QIcon(os.path.join(
              basedir, "bug.png")),
     "Your &button2",
     self,
 )
```

Contoh Aplikasi

```
        button_action2.setStatusTip(
             "This is your button2")
        button_action2.triggered.connect(
             self.onMyToolBarButtonClick)
        button_action2.setCheckable(True)
        toolbar.addAction(button_action2)
        toolbar.addWidget(QLabel("Hello"))
        toolbar.addWidget(QCheckBox())
        self.setStatusBar(QStatusBar(self))
        menu = self.menuBar()
        file_menu = menu.addMenu("&File")
        file_menu.addAction(button_action)
        file_menu.addSeparator()
        file_menu.addAction(button_action2)

    def onMyToolBarButtonClick(self, s):
        print("click", s)

app = QApplication(sys.argv)
window = MainWindow()
window.show()
app.exec()
```

Submenu

- Untuk menambahkan submenu, cukup membuat menu baru dengan memanggil addMenu() pada menu induk.
- Kemudian dapat menambahkan tindakan seperti biasa. Misalnya:

```
file_submenu = file_menu.addMenu("Submenu")
file_submenu.addAction(button_action2)
```