Pemrograman Desktop

Widgets-Qtwidge

yoga@trunojoyo.ac.id

Pendahuluan

- Qtwidget adalah nama yang diberikan kepada komponen UI tempat pengguna dapat berinteraksi.
- Antarmuka pengguna terdiri dari beberapa widget, disusun dalam jendela.
- Qt hadir dengan banyak pilihan widget yang tersedia, dan bahkan memungkinkan kita membuat widget khusus sendiri.

Contoh Aplikasi

```
import sys

from PyQt6.QtCore import Qt
from PyQt6.QtWidgets import QVBoxLayout  #
<2>
from PyQt6.QtWidgets import QWidget  # <1>
from PyQt6.QtWidgets import (
    QApplication,
    QCheckBox,
    QComboBox,
    QDateEdit,
    QDateTimeEdit,
    QDial,
    QDoubleSpinBox,
    QFontComboBox,
    QLabel,
    QLCDNumber,
    QLineEdit,
    QMainWindow,
    QProgressBar,
    QPushButton,
    QRadioButton,
    QSlider,
    QSpinBox,
    QTimeEdit,
)


# Subclass QMainWindow to customize your
application's main window
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("Widgets App")

        layout = QVBoxLayout()
        widgets = [
            QCheckBox,
            QComboBox,
```

Contoh Aplikasi

```
QDateEdit,
            QDateTimeEdit,
            QDial,
            QDoubleSpinBox,
            QFontComboBox,
            QLCDNumber,
            QLabel,
            QLineEdit,
            QProgressBar,
            QPushButton,
            QRadioButton,
            QSlider,
            QSpinBox,
            QTimeEdit,
        ]

        for w in widgets:
            layout.addWidget(w())
```

```
        widget = QWidget()
        widget.setLayout(layout)

        Widget will expand

        self.setCentralWidget(widget)


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

app.exec()
```

Contoh Widget

Widget	What it does
QCheckbox	A checkbox
QComboBox	A dropdown list box
QDateEdit	For editing dates
QDateTimeEdit	For editing dates and datetimes
QDial	Rotatable dial
QDoubleSpinbox	A number spinner for floats
QFontComboBox	A list of fonts
QLCDNumber	A quite ugly LCD display

QLabel	Just a label, not interactive
QLineEdit	Enter a line of text
QProgressBar	A progress bar
QPushButton	A button
QRadioButton	A group with only one active choice
QSlider	A slider
QSpinBox	An integer spinner
QTimeEdit	For editing times

QLabel

- QLabel, yang bisa dibilang salah satu widget paling sederhana yang tersedia di kotak peralatan Qt.
- Berupa satu baris sederhana teks yang dapat dposisikan pada aplikasi.
- Teks tersebut bisa di buat dengan memasukkan teks kedalam method yang disediakan.

    widget = QLabel("Hello") atau menggunakan method .setText()

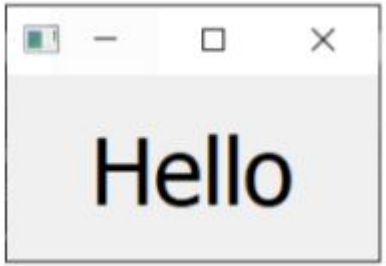
    widget = QLabel("1")# The label is created with the text 1
    widget.setText("2")# The label now shows 2

Contoh Aplikasi

```
import sys
from PyQt6.QtCore import Qt
from PyQt6.QtWidgets import QApplication,
QLabel, QMainWindow

class MainWindow(QMainWindow):
   def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")
        widget = QLabel("Hello")
        font = widget.font() ①
        font.setPointSize(30)
        widget.setFont(font)
        widget.setAlignment(Qt.AlignmentFlag.AlignHCenter
                                              |
Qt.AlignmentFlag.AlignVCenter
                ) ②
        self.setCentralWidget(widget)

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

Aplikasi Pertama

- Mendapatkan font saat ini, menggunakan<widget>.font(), memodifikasinya, lalu menerapkannya kembali. Hal ini memastikan tampilan font tetap sesuai dengan gaya font sistem.
- Penyelarasan ditentukan dengan menggunakan flag dari namespace Qt.



Namespace Qt

Perataan horizontal

Flag	Behavior
Qt.AlignmentFlag.AlignLeft	Aligns with the left edge.
Qt.AlignmentFlag.AlignRight	Aligns with the right edge.
Qt.AlignmentFlag.AlignHCenter	Centers horizontally in the available space.
Qt.AlignmentFlag.AlignJustify	Justifies the text in the available space.

Perataan vertikal

Flag	Behavior
Qt.AlignmentFlag.AlignTop	Aligns with the top.
Qt.AlignmentFlag.AlignBottom	Aligns with the bottom.
Qt.AlignmentFlag.AlignVCenter	Centers vertically in the available space.

Namespace Qt

- Kita dapat menggabungkan flag bersama-sama menggunakan pipa (|), namun perlu diingat bahwa hanya dapat menggunakan satu flag perataan vertikal atau horizontal dalam satu waktu.

```
align_top_left = Qt.AlignmentFlag.AlignLeft |
Qt.AlignmentFlag.AlignTop
```

Contoh Aplikasi

```
import os
import sys
from PyQt6.QtGui import QPixmap
from PyQt6.QtWidgets import QApplication,
QLabel, QMainWindow
basedir = os.path.dirname(__file__)
print("Current working folder:",
os.getcwd())①
print("Paths are relative to:", basedir) ②

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

widget.setPixmap(QPixmap(os.path.join(basedir,"gam
bar.jpg")))

          self.setCentralWidget(widget)

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

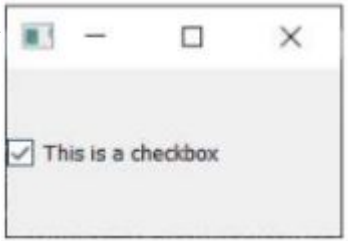
QPixmap

- Ubah kode untuk menambahkan .setScaledContents(True) ke label

```
widget.setPixmap(QPixmap(os.path.join(basedir,"gamb
ar.jpg")))

widget.setScaledContents(True)
```

QCheckBox

```
import sys
from PyQt6.QtCore import Qt
from PyQt6.QtWidgets import QApplication,
QCheckBox, QMainWindow

class MainWindow(QMainWindow):
   def __init__(self):
       super().__init__()
       self.setWindowTitle("My App")
       widget = QCheckBox("This is a
                    checkbox")
       widget.setCheckState(
             Qt.CheckState.Checked)
       # For tristate:
       widget.setCheckState(
           Qt.PartiallyChecked) # Or:
       widget.setTristate(True)
       widget.stateChanged.connect(
           self.show_state)
       self.setCentralWidget(widget)
   def show_state(self, s):
       print(Qt.CheckState(s) ==
           Qt.CheckState.Checked)
       print(s)

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



QCheckBox

Flag	Behavior
Qt.CheckState.Checked Qt.CheckState.Unchecked	Item is checked Item is unchecked
Qt.CheckState.PartiallyChecked	Item is partially checked

QComboBox

- QComboBox adalah daftar drop-down, ditutup secara default dengan panah untuk membukanya.
- Kita dapat memilih satu item dari daftar, dengan item yang dipilih saat ini ditampilkan sebagai label pada widget.
- ComboBox cocok untuk memilih pilihan dari daftar pilihan yang panjang.

QCheckBox

```
import sys
from PyQt6.QtCore import Qt
from PyQt6.QtWidgets import QApplication,
QComboBox, QMainWindow

class MainWindow(QMainWindow):
  def __init__(self):
      super().__init__()
      self.setWindowTitle("My App")
      widget = QComboBox()
      widget.addItems(["One", "Two",
          "Three"])
      widget.currentIndexChanged.connect(
            self.index_changed)
      widget.currentTextChanged.connect(
                self.text_changed)
      self.setCentralWidget(widget)
  def index_changed(self, i): # i is an int
      print(i)

  def text_changed(self, s): # s is a str
      print(s)

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

QListWidget

```
import sys
from PyQt6.QtWidgets import QApplication,
QListWidget, QMainWindow

class MainWindow(QMainWindow):
  def __init__(self):
      super().__init__()
      self.setWindowTitle("My App")
      widget = QListWidget()
      widget.addItems(["One", "Two",
             "Three"])
      widget.currentItemChanged.connect(
             self.index_changed)
      widget.currentTextChanged.connect(
             self.text_changed)
      self.setCentralWidget(widget)

  def index_changed(self, i):
      # Not an index, i is a QListItem
      print(i.text())

  def text_changed(self, s): # s is a str
      print(s)

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

QLineEdit

```
import sys
from PyQt6.QtCore import Qt
from PyQt6.QtWidgets import QApplication,
QLineEdit, QMainWindow

class MainWindow(QMainWindow):
  def __init__(self):
      super().__init__()
      self.setWindowTitle("My App")
      widget = QLineEdit()
      widget.setMaxLength(10)
      widget.setPlaceholderText(
          "Enter your text")
      # widget.setReadOnly(True)
      # uncomment this to make readonly
      widget.returnPressed.connect(
          self.return_pressed)
      widget.selectionChanged.connect(
          self.selection_changed)
      widget.textChanged.connect(
          self.text_changed)
      widget.textEdited.connect(
          self.text_edited)
      self.setCentralWidget(widget)

  def return_pressed(self):
      print("Return pressed!")
      self.centralWidget().setText("BOOM!")

  def selection_changed(self):
      print("Selection changed")

print(self.centralWidget().selectedText())
```

QLineEdit

```
  def text_changed(self, s):
      print("Text changed...")
      print(s)

  def text_edited(self, s):
      print("Text edited...")
      print(s)

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

QSpinBox dan QDoubleSpinBox

```
import sys
from PyQt6.QtWidgets import QApplication,
QMainWindow, QSpinBox

class MainWindow(QMainWindow):
  def __init__(self):
      super().__init__()
      self.setWindowTitle("My App")
      widget = QSpinBox()
      # Or: widget = QDoubleSpinBox()
      widget.setMinimum(-10)
      widget.setMaximum(3)
      # Or: widget.setRange(-10,3)
      widget.setPrefix("$")
      widget.setSuffix("c")
      widget.setSingleStep(3)
      # Or e.g. 0.5 for QDoubleSpinBox
      widget.valueChanged.connect(
       s  elf.value_changed)
      widget.textChanged.connect(
      self.value_changed_str)
      self.setCentralWidget(widget)

  def value_changed(self, i):
      print(i)

  def value_changed_str(self, s):
      print(s)

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

QSlider

```
import sys
from PyQt6.QtCore import Qt
from PyQt6.QtWidgets import QApplication,
QMainWindow, QSlider

class MainWindow(QMainWindow):
  def __init__(self):
      super().__init__()
      self.setWindowTitle("My App")
      widget = QSlider()
      widget.setMinimum(-10)
      widget.setMaximum(3)
      # Or: widget.setRange(-10,3)
      widget.setSingleStep(3)
      widget.valueChanged.connect(
         self.value_changed)
      widget.sliderMoved.connect(
         self.slider_position)
      widget.sliderPressed.connect(
         self.slider_pressed)
      widget.sliderReleased.connect(
         self.slider_released)
      self.setCentralWidget(widget)

  def value_changed(self, i):
      print(i)

  def slider_position(self, p):
      print("position", p)

  def slider_pressed(self):
      print("Pressed!")

  def slider_released(self):
      print("Released")

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

QDial

```
import sys
from PyQt6.QtCore import Qt
from PyQt6.QtWidgets import QApplication,
QDial, QMainWindow

class MainWindow(QMainWindow):
  def __init__(self):
      super().__init__()
      self.setWindowTitle("My App")
      widget = QDial()
      widget.setRange(-10, 100)
      widget.setSingleStep(1)
      widget.valueChanged.connect(
          self.value_changed)
      widget.sliderMoved.connect(
          self.slider_position)
      widget.sliderPressed.connect(
          self.slider_pressed)
      widget.sliderReleased.connect(
          self.slider_released)
      self.setCentralWidget(widget)

  def value_changed(self, i):
      print(i)

  def slider_position(self, p):
      print("position", p)

  def slider_pressed(self):
      print("Pressed!")

  def slider_released(self):
      print("Released")

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