Inhaltsverzeichnis

QtDesigner Tricks

Installieren von PyQt in eine normale Python- Umgebung

In Linux

```
pip install pyqt6 pyside6 PyQt6Designer
```

in Windows

```
pip install pyqt6-tools
```

in Windows dann starten mit `pyqt6-tools designer`

Widgets in Layouts verpacken

Wenn man GUI- Oberflächen für Python mit dem QtDesigner macht, ist das ja fast einfach - fast..., denn die einzelnen Widgets jeweils in ihre jeweils übergeordneteten Layouts zu drücken, artet bei mir immer zu einem Akt der Verzweiflung aus, weil ich jedesmal vergessen habe, wie man vorgehen muß. Darum habe ich es mir jetzt mal aufgeschrieben:

- Am Anfang Finger weg von den Layouts!
- Stattdessen erst alle Einzelelemente (Widgets) in den gewünschten Containern anordnen und im Fenster verteilen
- Dann die Layouts von innen nach ausen anlegen, Containern kann auch per 'Add Layout..' ein Layout zugewiesen werden, sie übernehmen dann also quasi die Layout- Funktion
  - die beteiligten Widgets markieren
  - über 'Add Layout' den markierten Widgets ihr jeweils übergeordnetes Layout zuweisen
  - dann die jeweiligen Layouts weiter zusammenfassen
  - in der obersten Ebene aber kann dem dem „Top“ Widget so kein Layout mehr zuweisen, was aber notwendig ist, damit die Gesamtheit im Fenster verteilt wird.
  - Für diese oberste Zuweisung muß man das Top- Widget auswählen und dann im QtDesigner- Menu den Punkt 'Form' auswählen. Da kann man dann dem Top- Widget auch ein Layout zuweisen und dann endlich passen sich die Widgets dynamisch der Fenstergröße an.

Wenn man daran zerbricht, dass ein Layout partout nicht das übergeordnete Widget ausfüllen will: Das übergeordnete Widget mit der rechten Maustaste anklicken und dann `Layout→Layout as Grid` anwählen... [1)]

Compilieren der .UI- Dateien

```
pyuic5 dialog.ui > dialog.py
```

Kleines Beispiel

```
import sys

from PyQt5.QtWidgets import (
    QApplication, QDialog, QMainWindow, QMessageBox
)
from PyQt5.uic import loadUi

from  SAEJ_2799_ui import Ui_MainWindow

class Window(QMainWindow, Ui_MainWindow):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.setupUi(self)
        self.connectSignalsSlots()

    def connectSignalsSlots(self):
        self.pushButton_connect.clicked.connect(self.connect)
        self.pushButton_send.clicked.connect(self.send)
        self.pushButton_clear_log.clicked.connect(self.clear_log)
        self.action_About.triggered.connect(self.about)


if __name__ == "__main__":
    app = QApplication(sys.argv)
    win = Window()
    win.show()
    sys.exit(app.exec())
```

Unterschiede zwischen interner und externer, model based Datenhaltung

Es gibt UI Elemente, die ihre Daten intern speichern, und welche, die mit einem externen Datenmodell syncronisieren. Je nach Anwendung nimmt man das eine oder andere. Die einzelnen Elemente sind auf der QT- Website erklärt

[1)]

https://stackoverflow.com/a/3494326