Übersicht

GUI

PyQT

Graphical User Interface

GUI

GUI - Graphical User Interface

- Wikipedia sagt:

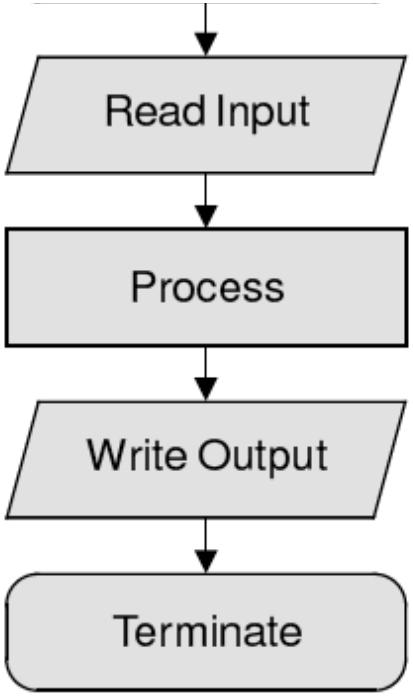
  *Eine **grafische Benutzeroberfläche** ist eine Software-Komponente, die dem Benutzer eines Computers die Interaktion mit der Maschine über grafische Symbole erlaubt. Die Darstellungen und Elemente (Arbeitsplatz, Symbole, Papierkorb, Menü) können meist unter Verwendung eines Zeigegerätes wie einer Maus gesteuert werden.*

- 1970 erstes System mit GUI – Xerox Alto von Xerox PARC
- 1983 Apple Lisa
- 1985 Microsoft Windows 1.03
- 1992 Mircosoft Windows 3.1
- 2000/2002 KDE (QT) / GNOME (GTK)
- 2009 Windows 7

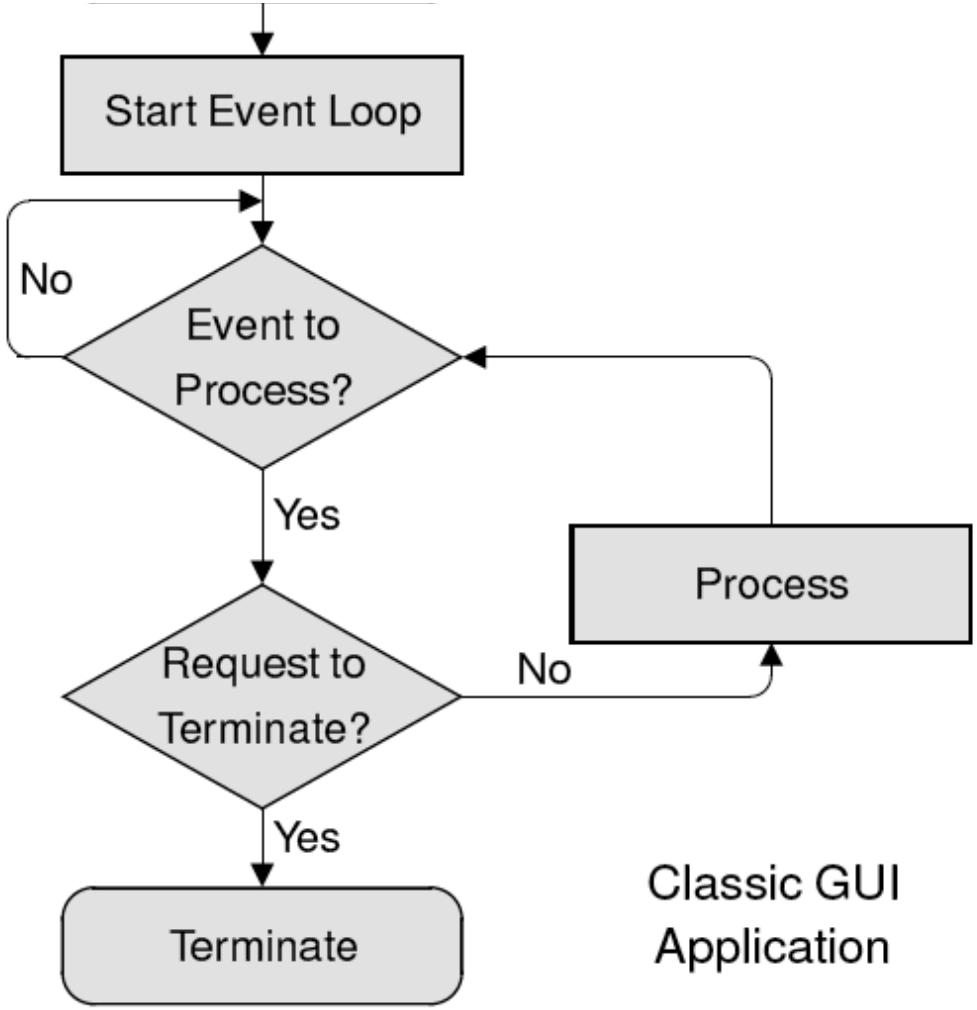
Alternativen

- Tkinter
  - basiert auf Tcl/Tk
  - kein Plattformkonformes GUI
- wxPython
  - basiert auf wxWindows
  - Wrapper für native Bibliotheken (Win32 Controls, GTK)
- PyGTK
  - basiert auf GTK (Gimp Toolkit)
  - bedingt plattformunabhängig
- Pythonwin
  - Binding für Windows MFC Bibliothek (nicht portierbar)

Programmablauf



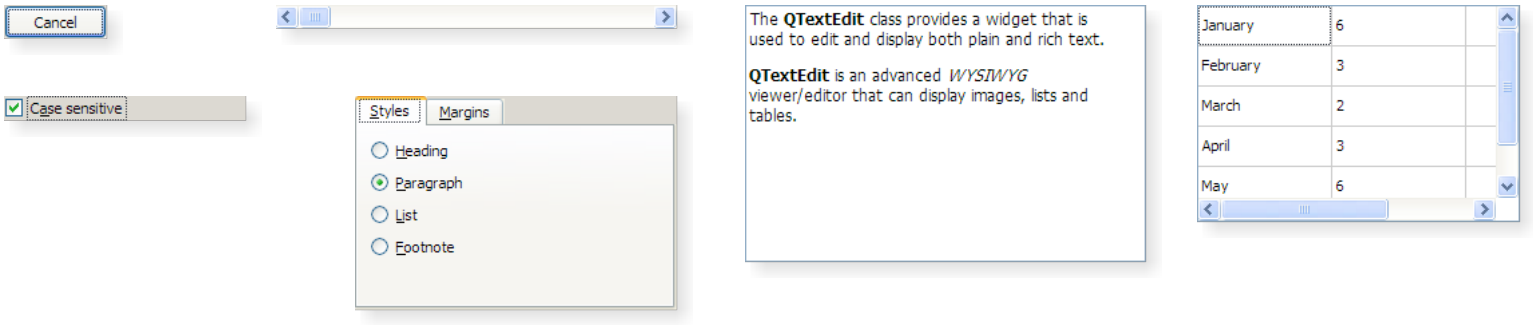
Classic Batch-processing Application



Classic GUI Application

Widget

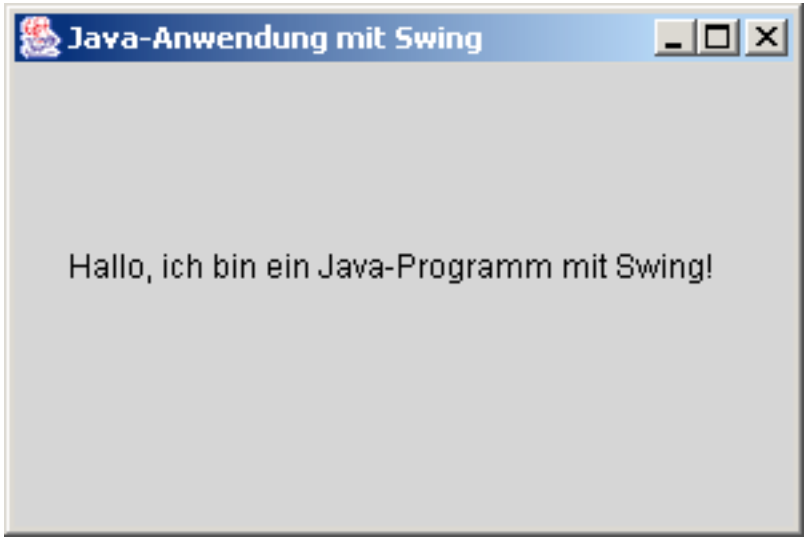
- Funktionales Element der GUI
- Abgeschlossener Funktionsumfang
- Graphische Repräsentation eines Models/Tätigkeit
- Bsp.: Button, Textfeld, Tabelle, Label



- Achtung: Abgrenzung!
  - Hier explizit nicht Google, Dashboard oder Yahoo Widgets

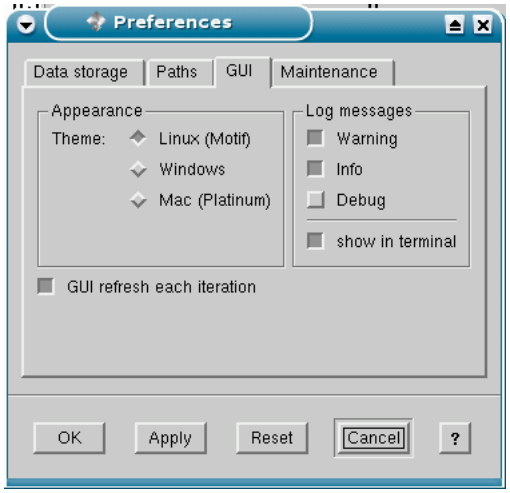
Frame/Window

- Containerfläche zur Darstellung von Widgets
- Layout Management
- Ziel für Systembuttons (Schliessen, Maximieren, Minimieren)
- Titelzeile (mit Icon)

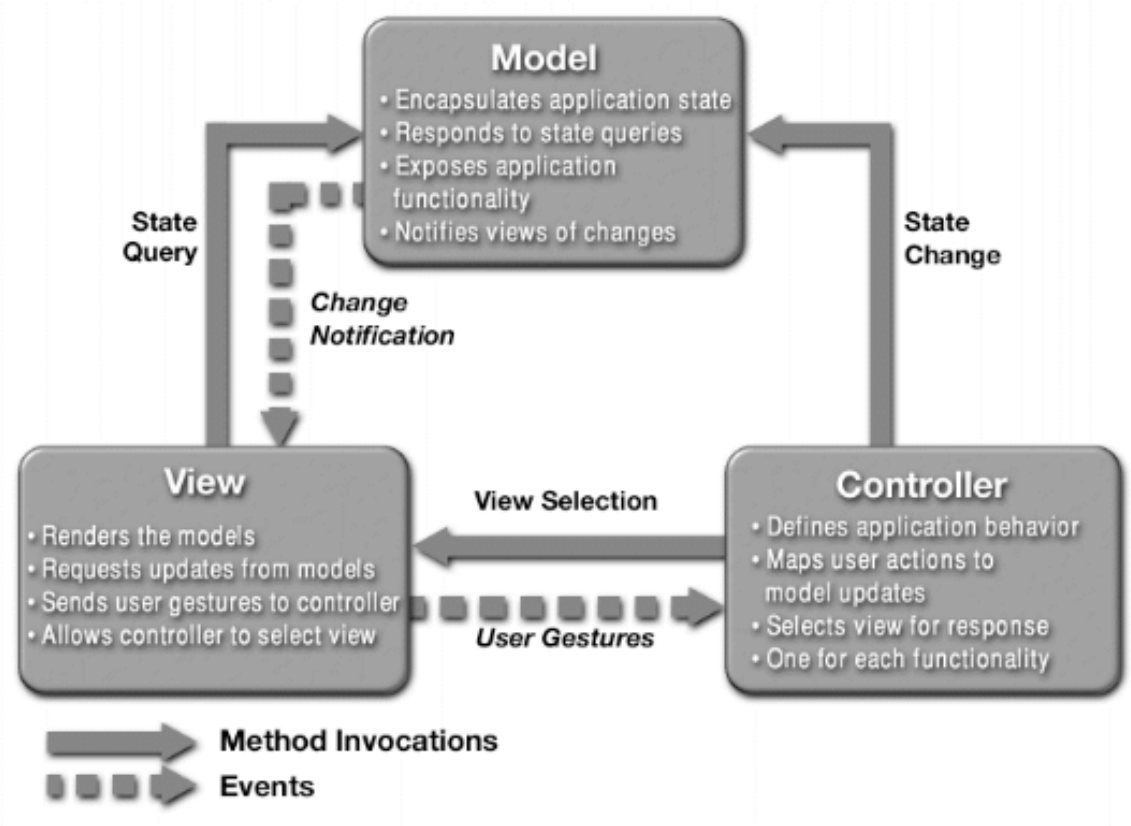


Dialog

- Top Level Window
- Aufgaben bezogen
- Kommunikation mit Nutzer
- Modal vs. Non-Modal
- Rückgabewert und Standard Buttons
  - Ok, Cancel, Apply, Reset



MVC – Model View Controller



Python QT Binding

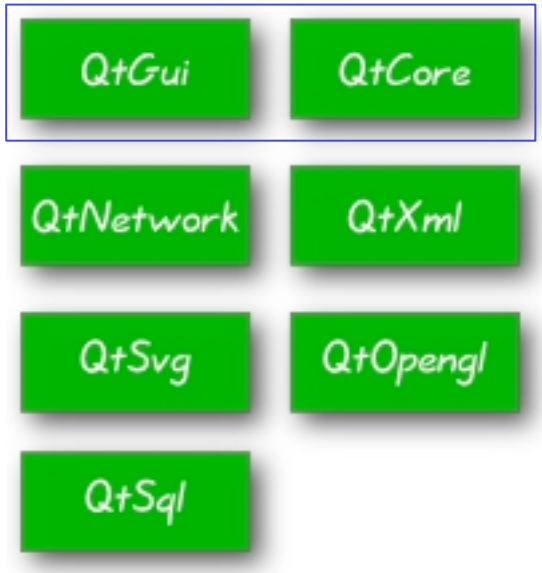
PYQT

PyQT

- QT
  - aktuell Version 4.7
  - entwickelt von Nokia, früher Trolltech
  - liegt KDE zugrunde
- PyQT
  - Binding für QT
- Python(x,y)
  - QT – 4.5.2
  - Python – 2.6

```
>>> import PyQt4
>>> help(PyQt4)
```

QT Bibliothek



Nützliche Webseiten

- Riverbank
  http://www.riverbankcomputing.co.uk/software/pyqt/intro

- QT 4.5 - Referenz
  http://doc.qt.nokia.com/4.5/index.html

- PyQT – Tutorial
  http://zetcode.com/tutorials/pyqt4/

Hello World

```
>>> import sys
>>> from PyQt4.QtCore import *
>>> from PyQt4.QtGui  import *

>>> app    = QApplication(sys.argv)

>>> widget = QWidget()
>>> widget.resize(250, 150)
>>> widget.setWindowTitle('Hello World')

>>> widget.show()

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

MyWidget

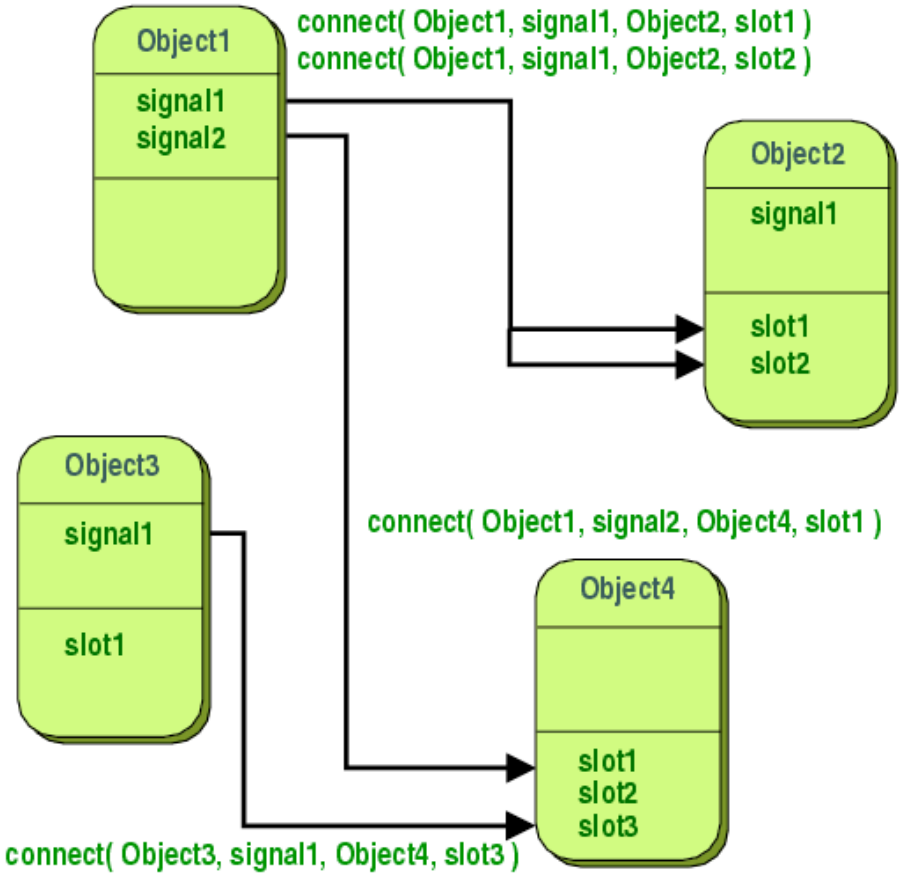
```
from PyQt4         import QtCore
from PyQt4         import QtGui
from PyQt4.QtGui   import QWidget, QIcon, QPushButton

class MyWidget(QWidget):

    def __init__(self, parent = None):

        QWidget.__init__(self, parent)
        self.setGeometry(300, 300, 250, 150)
        self.setWindowTitle('MyWidget')
        self.setWindowIcon(QIcon('icons/web.png'))

 if __name__ == "__main__":
        widget = MyWidget()
        widget.show()
```

Signals and Slots

- Signale & Slots sind spezielle Funktionen
- Wird zur Laufzeit gekoppelt

```
connect(Obj1, Fkt1, Obj2, Fkt2)
```

- Implementation des Observer-Pattern
- Alternative zu Callback Funktionen

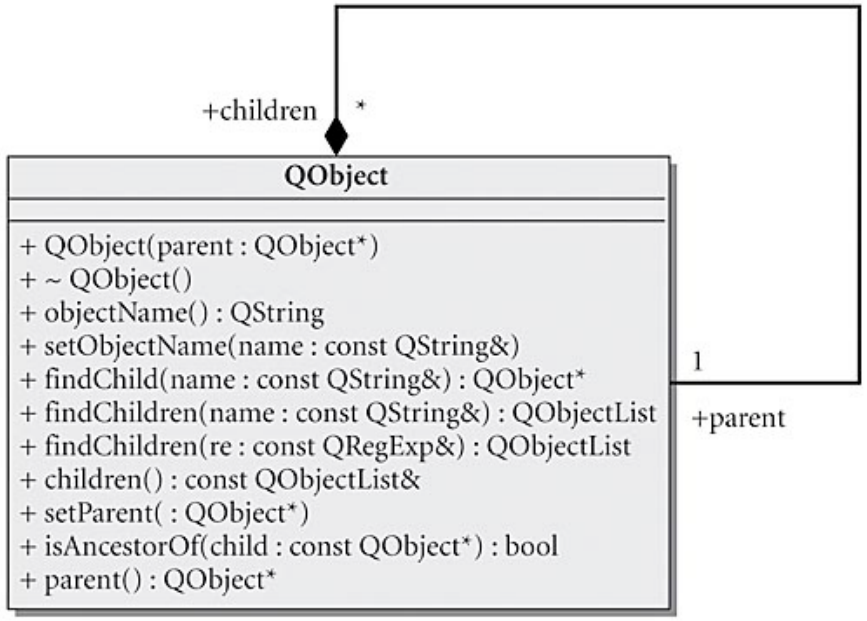


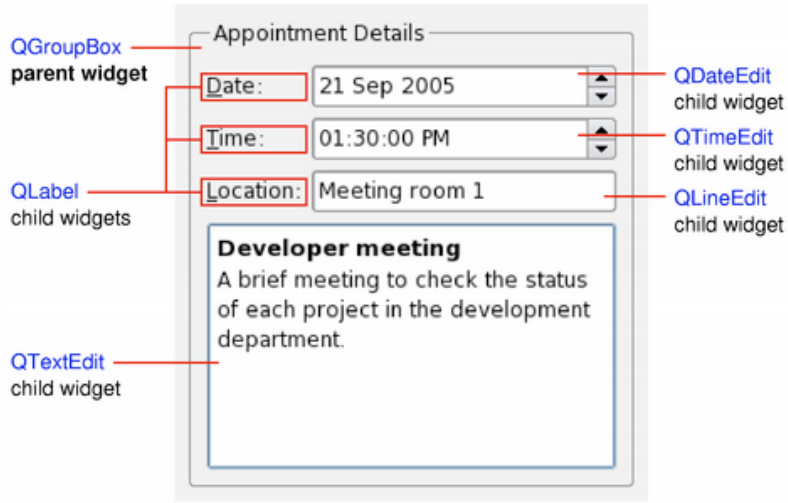
Signals & Slots - Beispiel

```
from PyQt4          import QtCore
from PyQt4          import QtGui
from PyQt4.QtGui    import QWidget, QIcon, QPushButton

class MyWidget(QWidget):

    def __init__(self, parent = None):

        QWidget.__init__(self, parent)
        self.setGeometry(300, 300, 250, 150)
        self.setWindowTitle('MyWidget')
        self.setWindowIcon(QIcon('icons/web.png'))
        quit = QPushButton('Close', self)
        quit.setGeometry(10, 10, 60, 35)
        self.connect( quit,         QtCore.SIGNAL('clicked()'),
                      QtGui.qApp, QtCore.SLOT('quit()') )
```

Composite

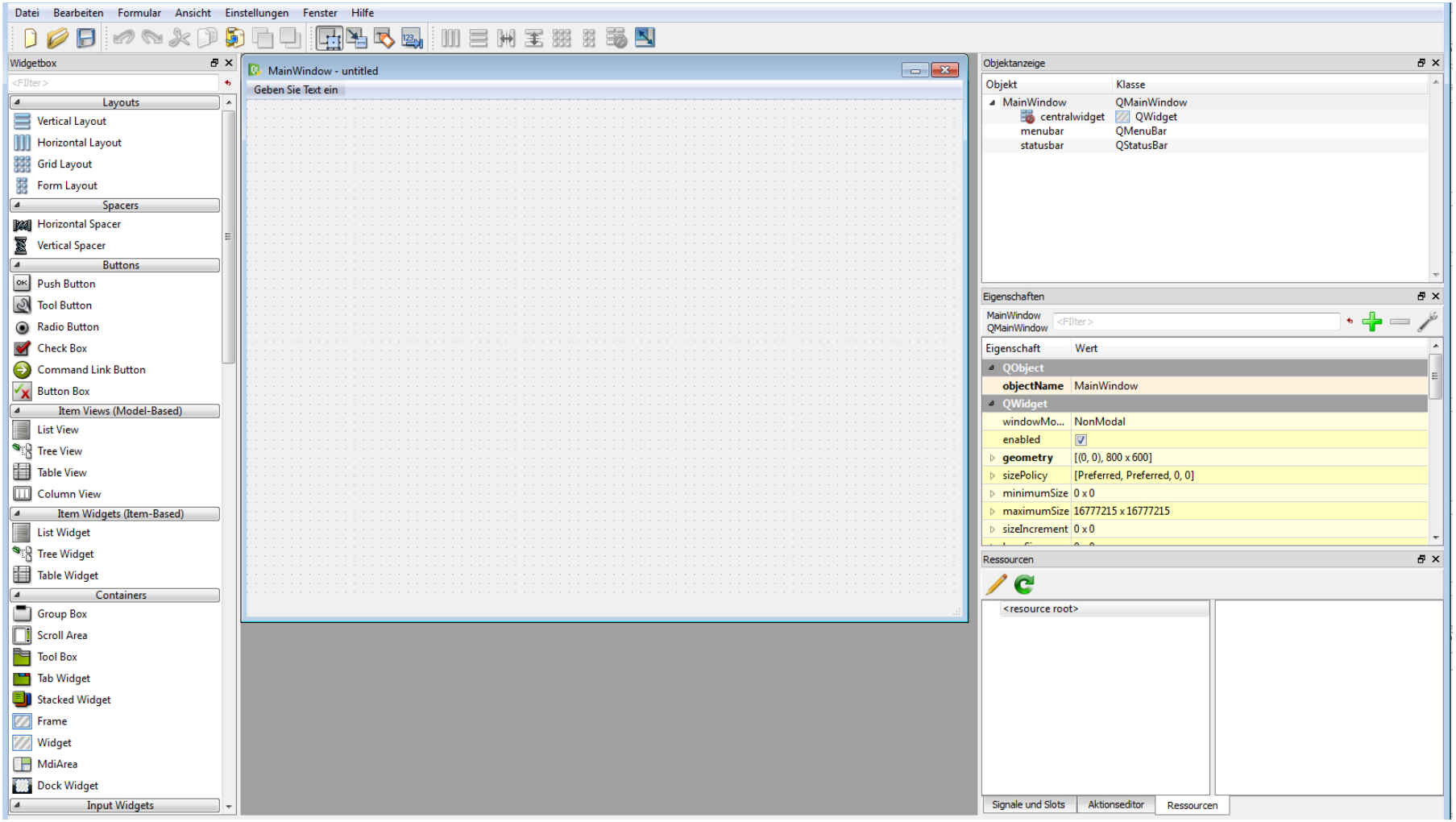
- Design Pattern zur Abbildung von Hierarchien
- Rekursives Zusammenbauen von Inhalten





TECHNISCHE UNIVERSITÄT DRESDEN

QT Designer



QT Designer

- Visuelles Erstellen von UIs
- Generiert *.ui Dateien basierend auf QML
  - XML basierte Beschreibungssprache
- Widgetnamen
  - können über Object Inspector eingestellt werden
  - können für Programmierung genutzt werden
- pyuic4 erstellt Python Datei aus *.ui Datei

```
C:\Python26\Lib\site-packages\PyQt4\pyuic4.bat –o <OutputDatei>
<InputDatei>
```

QTDesigner - Beispiel

- Ui_MainWindow.py
  - erzeugt über pyuic4 aus DuplicateFinder.ui

- In eigenes Projekt einfügen

```
from PyQt4.QtGui import QMainWindow
from Ui_MainWindow import Ui_MainWindow


class DuplicateFinderUI(QMainWindow):

    def __init__(self):
        QMainWindow.__init__(self)
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.ui.lineEdit.setText('C:\\Users\\')
```

Threads

- Thread
  - einzelner Programmablauf
  - unabhängig von anderen Threads (Nebenläufigkeit)
  - Hauptprogramm -> "Thread-0"

- Hauptklasse QThread (PyQt4.QtCore)
  - einfache Kommunikation über SIGNALs & SLOTs
  - 3 Phasen
    - Initialisierung (__init__ Method)
    - Start (start Methode) – ist gegeben, muss nicht programmiert werden
    - Laufzeit (run Methode)

- Alle Threads werden beendet, wenn "Thread-0" beendet wird!

Standard Dialoge

Operation	Erklärung
QColorDialog	Farbwähler
QErrorMessage	Fehlermeldung
QFileDialog	Datei- oder Verzeichnisauswahl
QFontDialog	Schriftartwahl
QInputDialog	Einfacher Dialog um einen Wert abzufragen
QMessageBox	Modaler Dialog zur Informationsdarstellung
QPrintDialog	Druckdialog
QProgressDialog	Zustandsanzeige
QDialog	Allgemeiner Dialog - Basisklasse aller Dialoge