for Python

Графический интерфейс (GUI)

Графический интерфейс пользователя сокр (Graphical User Interface) — это способ взаимодействия пользователя с компьютером с использованием графических элементов, таких как окна, кнопки и меню.

Требования к GUI

Наглядность. DWIM — Do What I Mean. Интерфейс должен быть интуитивно ясным.

Понятность. Пользователь должен понимать, где что можно найти, куда нажимать и так далее.

Удобство. Пользоваться графическим интерфейсом должно быть удобно — это то, ради чего он в принципе создавался. Это очень широкое понятие. Для его измерения даже есть специальные метрики: по ним определяют, насколько быстро и полно человек может разобраться в интерфейсе.

Эффективность. Эффективный интерфейс — это такой, который помогает достичь нужного результата как можно быстрее.

Обратная связь. Графический интерфейс должен быть отзывчивым, показывать пользователю, что происходит в данный момент.

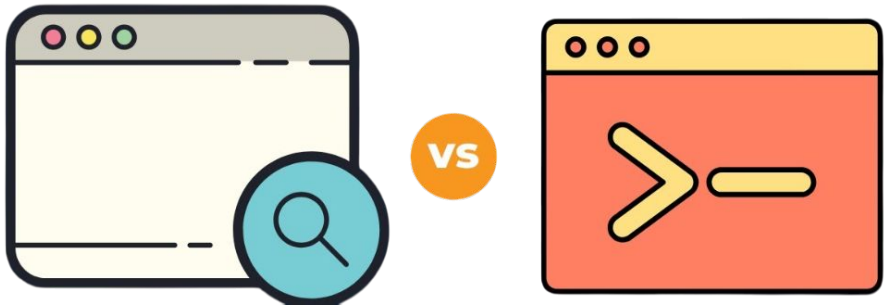
Последовательность. Последовательность означает, что элементы должны быть расположены в правильном порядке — таком, который удобен пользователю.

Красота. Интерфейс должен быть привлекательным: иметь приятную цветовую гамму, грамотно продуманные очертания.

Преимущества и недостатки GUI

+ Удобство
+ Широкая аудитория
+ Понятность
+ Скорость работы

- Производительность
- Профессиональное ПО для разработчиков нередко не имеет GUI

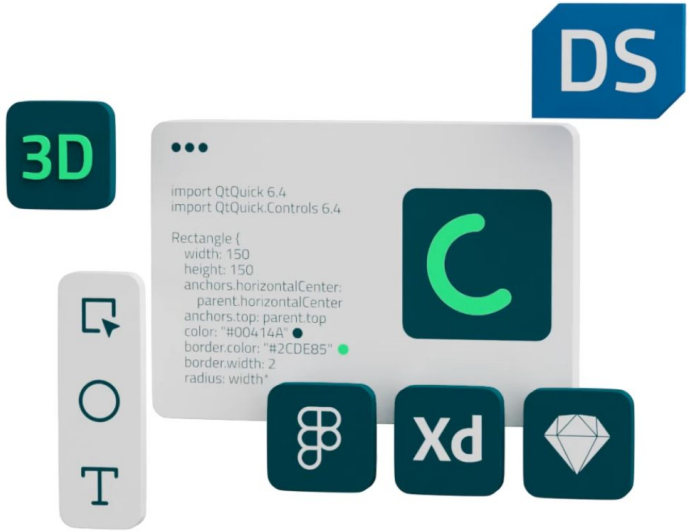


Qt

Qt — фреймворк для разработки кроссплатформенного программного обеспечения на языке программирования C++.

Для многих языков программирования существуют библиотеки, позволяющие использовать преимущества Qt:

- Python — PyQt, PySide;
- Ruby — QtRuby;
- Java — QtJambi;
- PHP — PHP-Qt и другие.



Что входит в Qt

Модули Qt:

- QtCore — ядро фреймворка.
- QtGUI — компоненты для создания интерфейсов.
- QtNetwork — функции для работы с сетевыми соединениями.
- и др.

Кроме библиотеки и ее модулей, Qt содержит дополнительное ПО, утилиты, справочники и внутренние языки.

- Qt Creator
- Qt Assistant
- QT Linguist
- Qt Designer
- Qt Quick
- QML

PyQt6 и PySide6

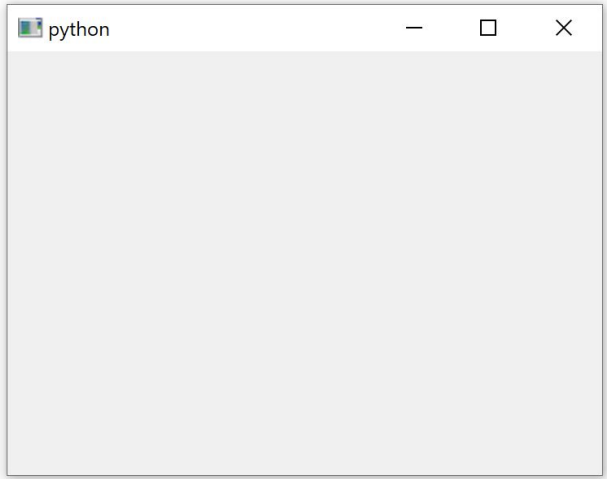
Оба пакета оборачивают одну и ту же библиотеку — Qt6 — и поэтому имеют на 99,9% идентичные API.

	PyQt6	PySide6
First stable release	Jan 2021	Dec 2020
Developed by	Riverbank Computing Ltd.	Qt
License	GPL or commercial	LGPL
Platforms	Python 3	Python 3

PyQt — это библиотека Python для создания приложений с графическим интерфейсом с помощью инструментария Qt. Созданная в Riverbank Computing, PyQt является свободным ПО (по лицензии GPL) и разрабатывается с 1999 года.

Установка: pip install pyqt6 или pip install pyside6

```python
from PyQt6.QtWidgets import QApplication, QWidget

import sys # Только для доступа к аргументам командной строки

# Приложению нужен один (и только один) экземпляр QApplication.
# Передаём sys.argv, чтобы разрешить аргументы командной строки для приложения.
# Если не будете использовать аргументы командной строки, QApplication([]) тоже
app = QApplication(sys.argv)

# Создаём виджет Qt — окно.
window = QWidget()
window.show()  # Важно: окно по умолчанию скрыто.

# Запускаем цикл событий.
app.exec()
```

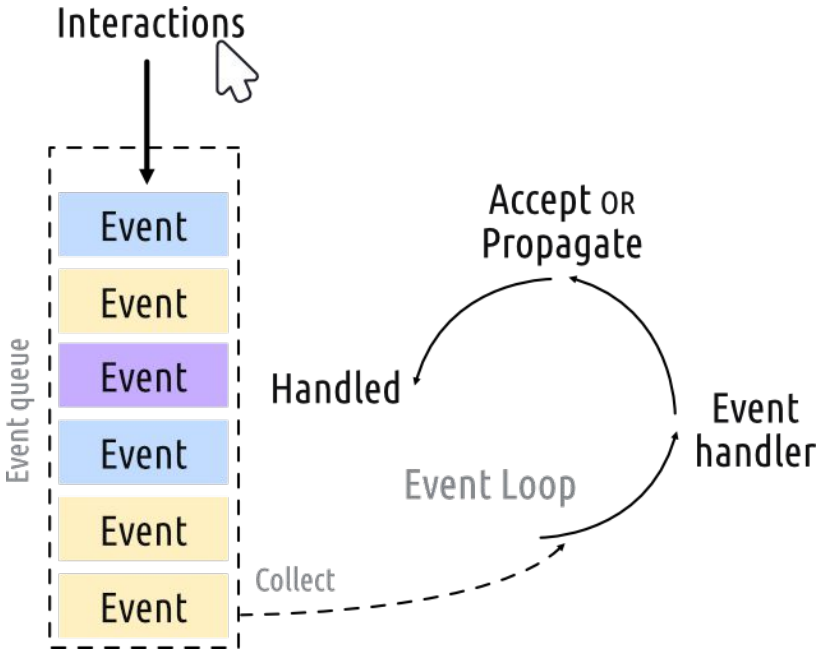


QApplication и цикл событий

Основной элемент всех приложений в Qt — класс **QApplication**.

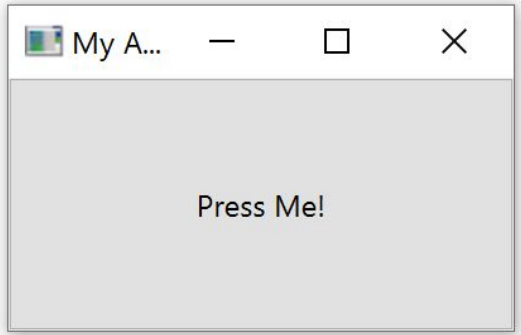
Для работы каждому приложению нужен один — и только один — объект QApplication, который содержит **цикл событий** приложения.

Это основной цикл, управляющий всем взаимодействием пользователя с графическим интерфейсом:



QMainWindow и подкласс окна

```
import sys

from PyQt6.QtCore import QSize, Qt
from PyQt6.QtWidgets import QApplication, QMainWindow, QPushButton


# Подкласс QMainWindow для настройки главного окна приложения
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("My App")
        button = QPushButton("Press Me!")

        # Устанавливаем центральный виджет Window.
        self.setCentralWidget(button)


app = QApplication(sys.argv)

window = MainWindow()
window.show()

app.exec()
```



Документация: https://doc.qt.io/qtforpython-6/PySide6/QtWidgets/QMainWindow.html#PySide6.QtWidgets.QMainWindow

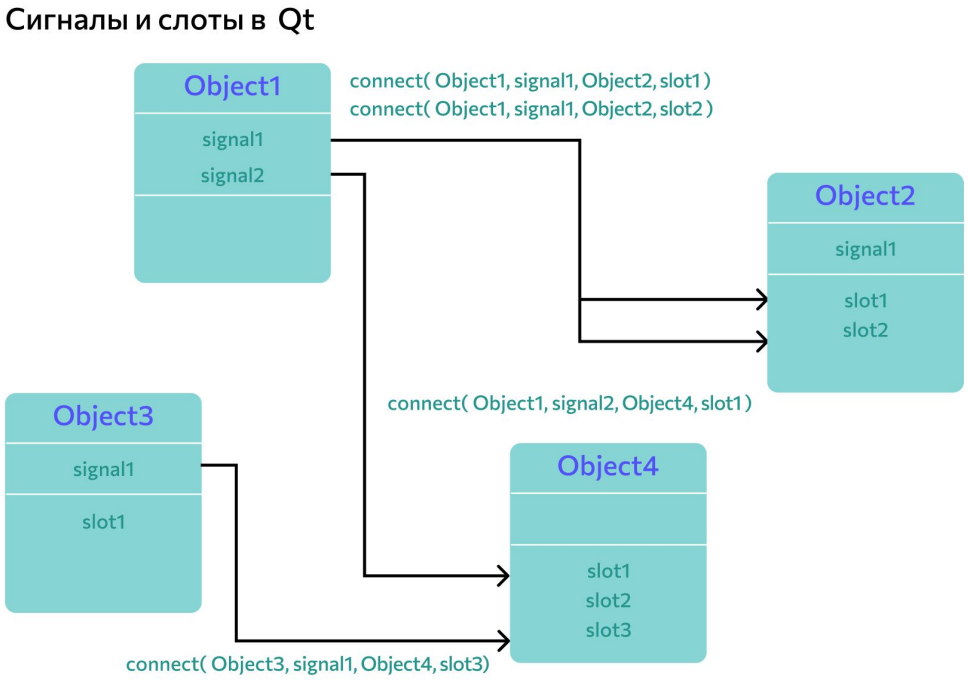
Изменение размеров окон и виджетов

В Qt размеры определяются с помощью объекта QSize (`from PyQt6.QtCore import QSize`).

```
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("My App")

        button = QPushButton("Press Me!")

        self.setFixedSize(QSize(400, 300))

        # Устанавливаем центральный виджет Window.
        self.setCentralWidget(button)
```

Кроме .setFixedSize() можно также вызвать .setMinimumSize() и .setMaximumSize(), чтобы установить минимальный и максимальный размеры соответственно.

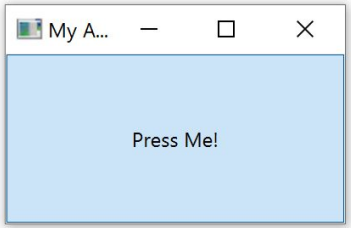
Слоты и сигналы



Сигналы — это уведомления, отправляемые виджетами, когда что-то происходит.

Слоты в Qt — это приёмники сигналов. Слотом в приложении на Python можно сделать любую функцию (или метод), просто подключив к нему сигнал.

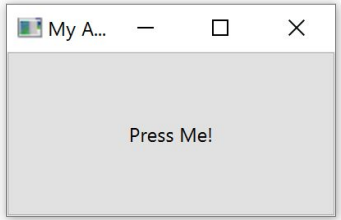
```python
import sys
from PyQt6.QtWidgets import QApplication, QMainWindow, QPushButton


class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("My App")

        button = QPushButton("Press Me!")
        button.setCheckable(True)
        button.clicked.connect(self.the_button_was_clicked)

        # Устанавливаем центральный виджет Window.
        self.setCentralWidget(button)

    def the_button_was_clicked(self):
        print("Clicked!")


app = QApplication(sys.argv)

window = MainWindow()
window.show()

app.exec()
```

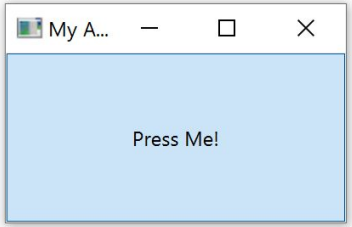



```
(.venv) D:\PycharmProjects\pyqt_example>python example4_slots.py
Clicked!
Clicked!
Clicked!
Clicked!
Clicked!
```

Получение и хранение данных

```
from PyQt6.QtWidgets import QApplication, QMainWindow, QPushButton

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.button_is_checked = True

        self.setWindowTitle("My App")

        button = QPushButton("Press Me!")
        button.setCheckable(True)
        button.clicked.connect(self.the_button_was_toggled)
        button.setChecked(self.button_is_checked)

        self.setCentralWidget(button)

    def the_button_was_toggled(self, checked):
        self.button_is_checked = checked
        print(self.button_is_checked)

app = QApplication([])
window = MainWindow()
window.show()
app.exec()
```

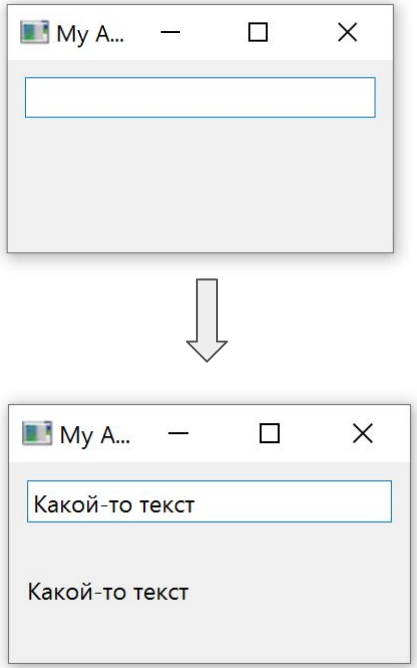





```
(.venv) D:\PycharmProjects\pyqt_example>python example6_data_slots.py
False
True
False
True
```

Подключение виджетов друг к другу напрямую

```
import sys
from PyQt6.QtWidgets import QApplication, QMainWindow, QLabel, QLineEdit, QVBoxLayout, QWidget


class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")

        self.label = QLabel()
        self.input = QLineEdit()

        self.input.textChanged.connect(self.label.setText)

        layout = QVBoxLayout()
        layout.addWidget(self.input)
        layout.addWidget(self.label)

        container = QWidget()
        container.setLayout(layout)

        # Устанавливаем центральный виджет Window.
        self.setCentralWidget(container)


app = QApplication(sys.argv)

window = MainWindow()
window.show()

app.exec()
```



События

Обработчики событий мыши

Обработчик	Событие
`mouseMoveEvent`	Мышь переместилась
`mousePressEvent`	Кнопка мыши нажата
`mouseReleaseEvent`	Кнопка мыши отпущена
`mouseDoubleClickEvent`	Обнаружен двойной клик

```
def mouseMoveEvent(self, e):
    self.label.setText("mouseMoveEvent")
```

События управления мышью

Метод	Возвращает
`.button()`	Конкретную кнопку, вызвавшую данное событие
`.buttons()`	Состояние всех кнопок мыши (флаги OR)
`.position()`	Относительную позицию виджета в виде целого `QPoint` .

```
def mousePressEvent(self, e):
    if e.button() == Qt.LeftButton:
        # здесь обрабатываем нажатие левой кнопки
        self.label.setText("mousePressEvent LEFT")
```

Виджеты (QtWidgets)

Документация: https://doc.qt.io/qtforpython-6/PySide6/QtWidgets/index.html#list-of-classes

Widget	What it does
QCheckbox	A checkbox
QComboBox	A dropdown list box
QDateEdit	For editing dates and datetimes
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 toggle set, with only one active item
QSlider	A slider
QSpinBox	An integer spinner
QTimeEdit	For editing times

QLabel

https://doc.qt.io/qtforpython-6/PySide6/QtWidgets/QLabel.html#PySide6.QtWidgets.QLabel

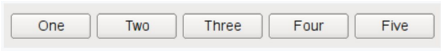
```
from PyQt6.QtWidgets import QApplication, QMainWindow, QLabel
from PyQt6.QtCore import Qt

class MainWindow(QMainWindow):

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


app = QApplication([])
window = MainWindow()
window.show()
app.exec()
```

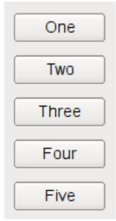
widget.setPixmap(QPixmap('test.jpg')) - используется для вставки изображений

Разметка виджетов

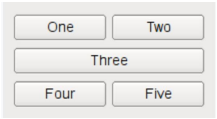
https://doc.qt.io/qtforpython-6/overviews/layout.html

1. **BoxLayout (QBoxLayout)**: Этот менеджер размещения располагает виджеты в горизонтальном или вертикальном порядке. Вы можете использовать QHBoxLayout (Horizontal Layout) для горизонтального расположения или QVBoxLayout (Vertical Layout) для вертикального.





2. **GridLayout (QGridLayout)**: С помощью этого менеджера размещения виджеты можно организовать в виде сетки. Вы указываете количество строк и столбцов, а затем добавляете виджеты в определенные ячейки.




```
window = QWidget()
button1 = QPushButton("One")
button2 = QPushButton("Two")
button3 = QPushButton("Three")
button4 = QPushButton("Four")
button5 = QPushButton("Five")
```

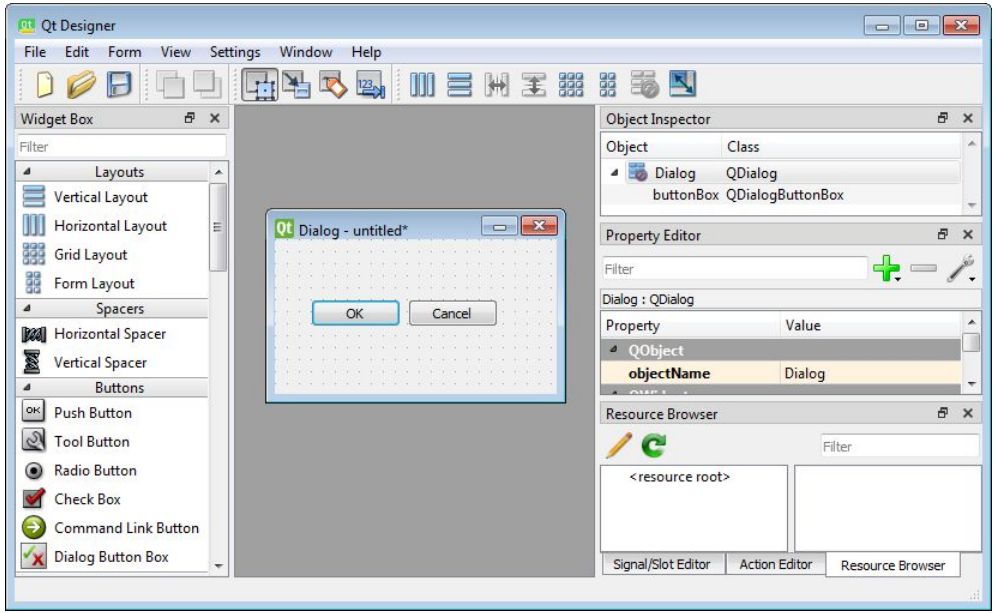
3. **FormLayout (QFormLayout)**: Этот тип разметки удобен для создания форм с метками и полями для ввода. Он автоматически располагает метки и соответствующие поля ввода вертикально.


```
layout = QHBoxLayout(window)
layout.addWidget(button1)
layout.addWidget(button2)
layout.addWidget(button3)
layout.addWidget(button4)
layout.addWidget(button5)
window.show()
```

Qt Designer

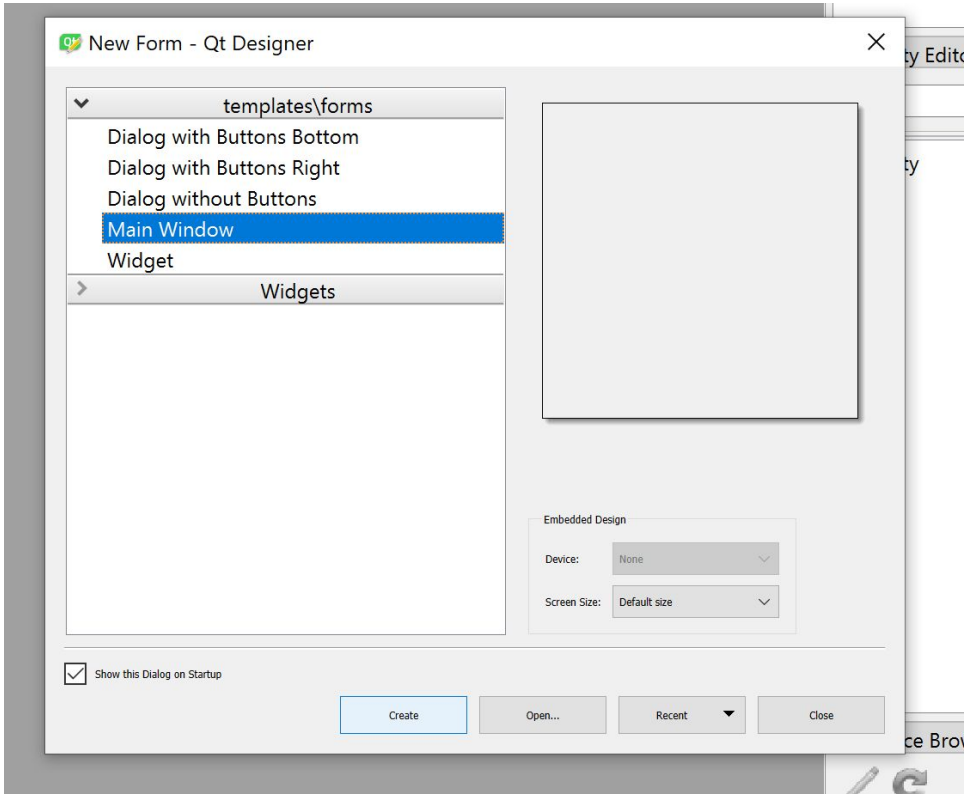
Qt Designer — кроссплатформенная свободная среда для разработки графических интерфейсов (GUI) для программ, использующих библиотеку Qt.

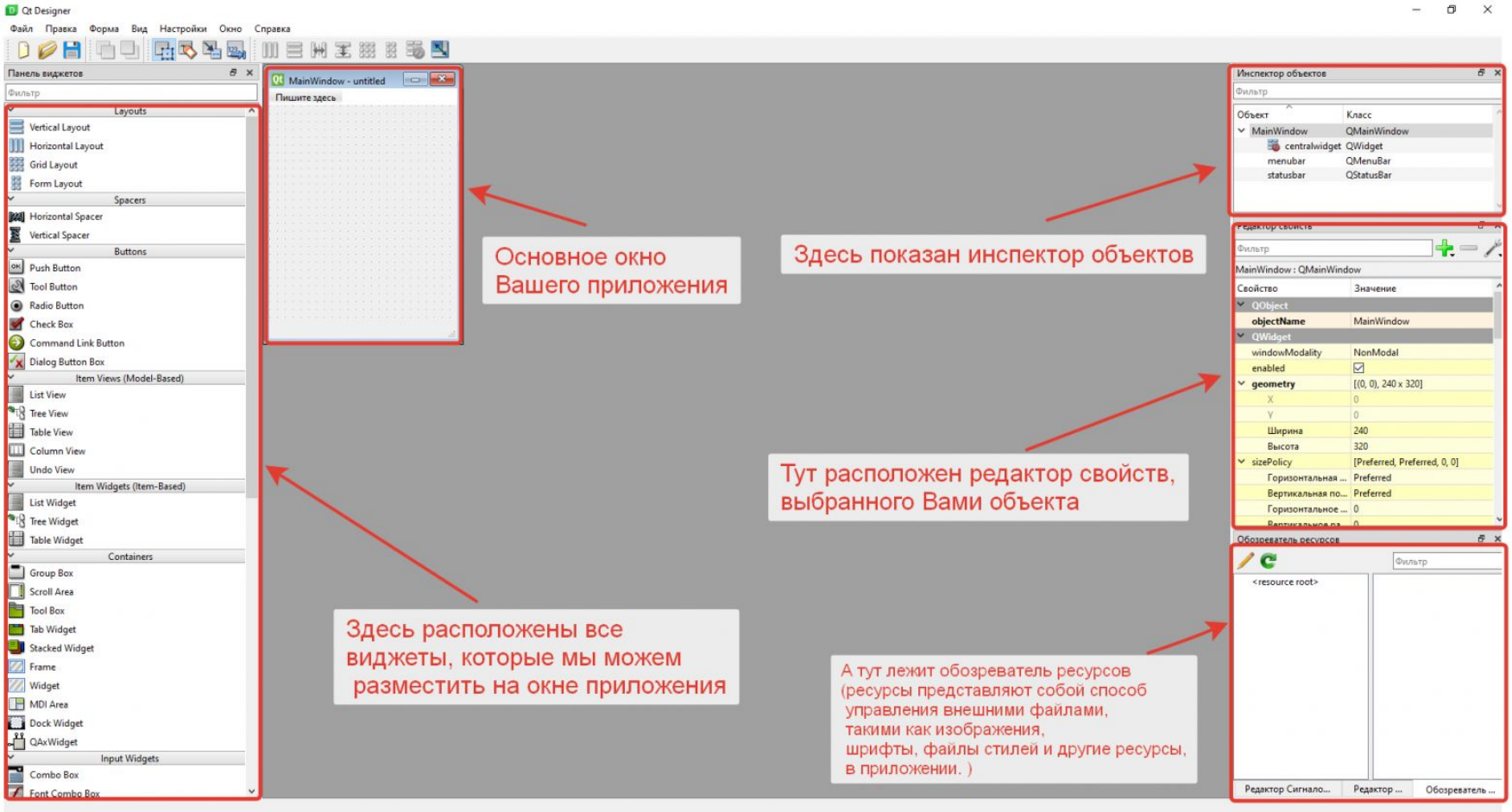
Ссылка на отдельное приложение: https://build-system.fman.io/qt-designer-download



При открытии приложения появится диалоговое окно с выбором шаблона Вашего будущего окна интерфейса.

Выберем Main Windows.

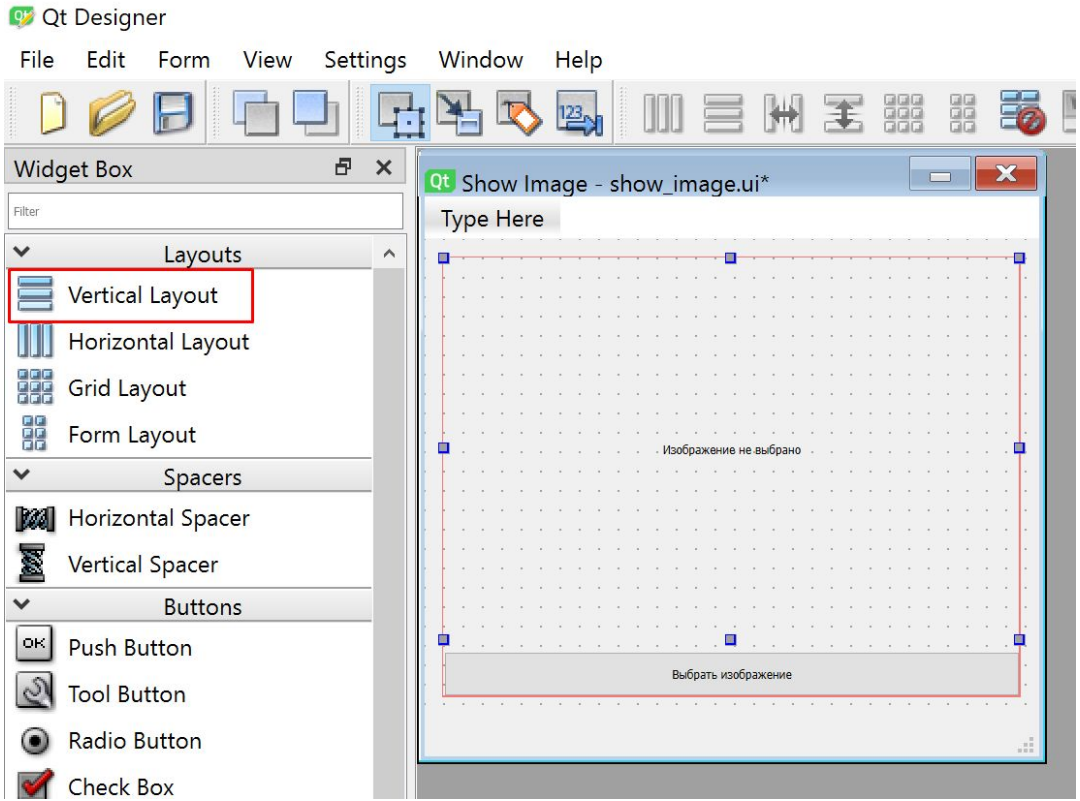



Qt Designer
Файл Правка Форма Вид Настройки Окно Справка
Панель виджетов
Фильтр
Layouts
Vertical Layout
Horizontal Layout
Grid Layout
Form Layout
Spacers
Horizontal Spacer
Vertical Spacer
Buttons
Push Button
Tool Button
Radio Button
Check Box
Command Link Button
Dialog Button Box
Item Views (Model-Based)
List View
Tree View
Table View
Column View
Undo View
Item Widgets (Item-Based)
List Widget
Tree Widget
Table Widget
Containers
Group Box
Scroll Area
Tool Box
Tab Widget
Stacked Widget
Frame
Widget
MDI Area
Dock Widget
QAxWidget
Input Widgets
Combo Box
Font Combo Box
MainWindow - untitled
Пишите здесь
Основное окно Вашего приложения
Здесь показан инспектор объектов
Здесь расположены все виджеты, которые мы можем разместить на окне приложения
Тут расположен редактор свойств, выбранного Вами объекта
А тут лежит обозреватель ресурсов (ресурсы представляют собой способ управления внешними файлами, такими как изображения, шрифты, файлы стилей и другие ресурсы, в приложении.)
Инспектор объектов
Фильтр
Объект
Класс
MainWindow
QMainWindow
centralwidget
QWidget
menubar
QMenuBar
statusbar
QStatusBar
Редактор свойств
Фильтр
MainWindow : QMainWindow
Свойство
Значение
QObject
objectName
MainWindow
QWidget
windowModality
NonModal
enabled
geometry
[(0, 0), 240 x 320]
X
0
Y
0
Ширина
240
Высота
320
sizePolicy
[Preferred, Preferred, 0, 0]
Горизонтальная ...
Preferred
Вертикальная по...
Preferred
Горизонтальное ...
0
Обозреватель ресурсов
Фильтр
<resource root>
Редактор Сигнало...
Редактор ...
Обозреватель ...

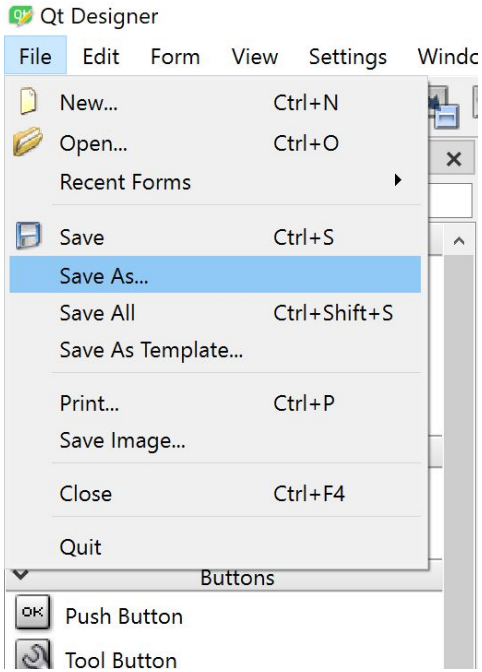
Создадим простой интерфейс приложения:

QPushButton - при нажатии появится проводник для выбора изображения.

QLabel - тут будет отображено выбранное изображение.



После завершения работы по созданию интерфейса сохраняем файл в директорию проекта. Сохраняемый файл имеет расширение .ui. Данный файл имеет xml разметку, в котором прописаны элементы, созданные на интерфейсе.



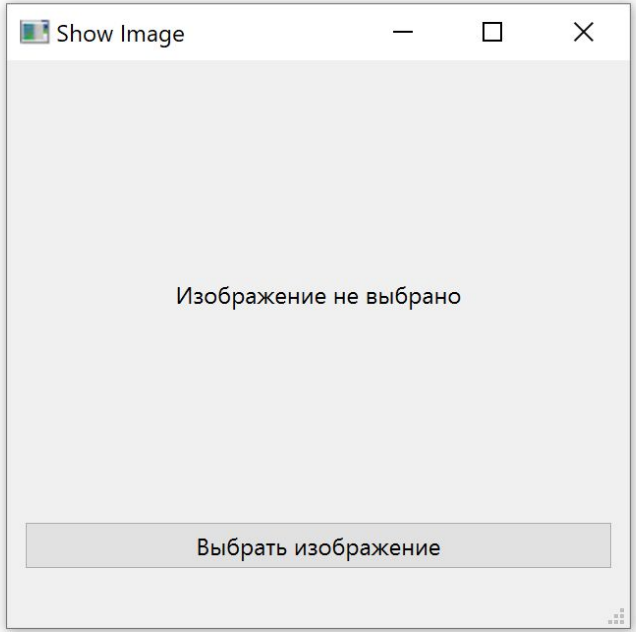
show_image.ui

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>317</width>
    <height>289</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Show Image</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <layout class="QVBoxLayout" name="verticalLayout_2">
    <item>
     <layout class="QVBoxLayout" name="verticalLayout" stretch="0,0">
      <property name="sizeConstraint">
       <enum>QLayout::SetDefaultConstraint</enum>
      </property>
      <item>
       <widget class="QLabel" name="label">
        <property name="text">
         <string>TextLabel</string>
```

Использование .ui файла

Для загрузки .ui файлов мы можем использовать uic модуль PyQt6, а именно uic.loadUI() метод. В качестве аргумента он принимает имя файла UI и загружает его, создавая полнофункциональный объект PyQt6.

```
from PyQt6.QtWidgets import QApplication
from PyQt6 import uic

if __name__ == '__main__':
    app = QApplication([])

    window = uic.loadUi('show_image.ui')
    window.show()

    app.exec()
```



Конвертация .ui-файла в Python

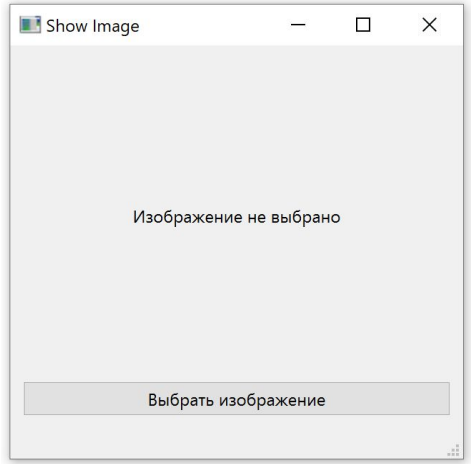
С помощью команды: pyuic6 show_image.ui -o ShowImage.py

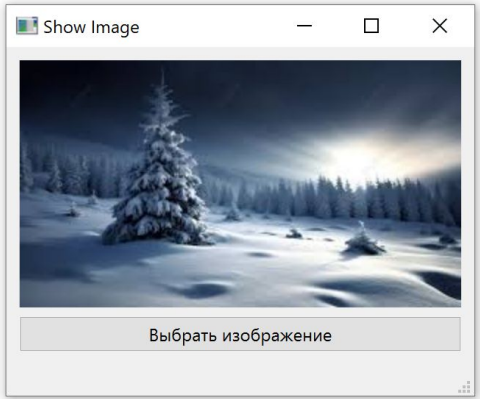
```python
from PyQt6.QtWidgets import QApplication, QMainWindow
from ShowImage import Ui_MainWindow

class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)

if __name__ == '__main__':
    app = QApplication([])
    window = MainWindow()
    window.show()
    app.exec()
```

ShowImage.py

```python
# Form implementation generated from reading ui file 'show_image.ui'
#
# Created by: PyQt6 UI code generator 6.7.1
#
# WARNING: Any manual changes made to this file will be lost when pyuic6 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt6 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(317, 289)
        self.centralwidget = QtWidgets.QWidget(parent=MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.verticalLayout_2 = QtWidgets.QVBoxLayout(self.centralwidget)
        self.verticalLayout_2.setObjectName("verticalLayout_2")
        self.verticalLayout = QtWidgets.QVBoxLayout()
        self.verticalLayout.setSizeConstraint(QtWidgets.QLayout.SizeConstraint.SetDefaultConstraint)
        self.verticalLayout.setObjectName("verticalLayout")
        self.label = QtWidgets.QLabel(parent=self.centralwidget)
        self.label.setTextFormat(QtCore.Qt.TextFormat.AutoText)
        self.label.setScaledContents(False)
        self.label.setAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
        self.label.setObjectName("label")
```

```python
from PyQt6.QtWidgets import QApplication, QMainWindow, QFileDialog
from PyQt6.QtGui import QPixmap
from ShowImage import Ui_MainWindow

class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)

        self.ui.pushButton.clicked.connect(self.open_image)

    def open_image(self):
        filename, _ = QFileDialog.getOpenFileName(
            self, 'Выбрать картинку', '.', 'Картинки (*.png *.jpg *.jpeg) '
        )
        if filename:
            pixmap = QPixmap(filename)
            self.ui.label.setPixmap(pixmap)
            self.resize(pixmap.width(), pixmap.height())

if __name__ == '__main__':
    app = QApplication([])

    window = MainWindow()
    window.show()

    app.exec()
```





Полезные ссылки

Документация:

https://doc.qt.io/qtforpython-6/api.html

https://doc.qt.io/

Учебник PyQt6:

https://www.pythonguis.com/tutorials/pyqt6-creating-your-first-window/

Перевод:

https://habr.com/ru/companies/skillfactory/articles/599599/