Проект QT. QtDesigner, pyuic, два способа подключения uic-файла

План урока

1. QT Designer
2. Как подключить интерфейс к основной программе
3. Размещение виджетов

Аннотация

В этом уроке разбирается популярный способ создания графических интерфейсов — с помощью программы QT Designer.

1. QT Designer

Когда на прошлом занятии мы создавали интерфейсы «руками» и размещали виджеты «на глазок», вы наверняка подумали, что есть какой-то более простой способ.
И он действительно есть. Это программа QTDesigner, которая включена в сборку PyQT5.

Но для ее использования необходимо установить библиотеку **pyqt5-tools**.

```
pip install pyqt5-tools
```

Теперь программа находится на вашем компьютере по адресу:

```
Путь_до_папки_с_Python\Lib\site-packages\pyqt5_tools\designer.exe
```

Важно

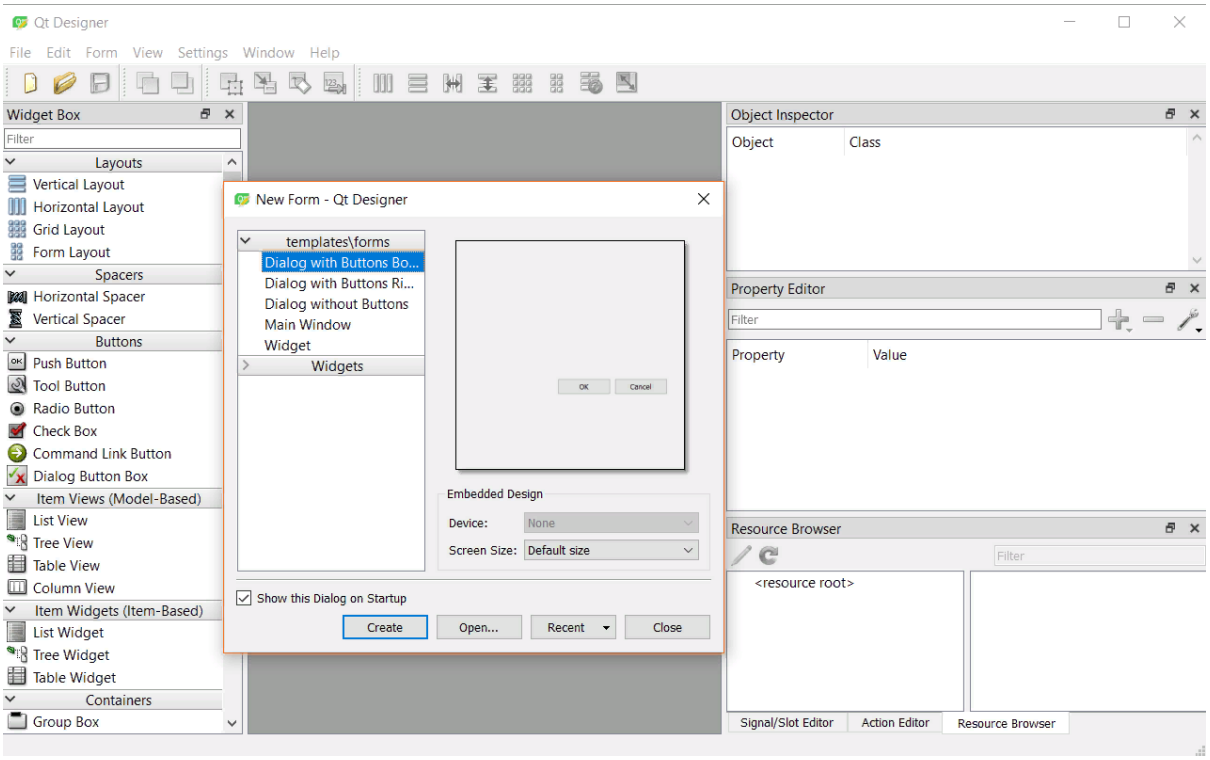
В MacOS для запуска надо открыть терминал и ввести команду:

```
open -a Designer
```

Если вы используете Linux, то действуйте по этой инструкции.

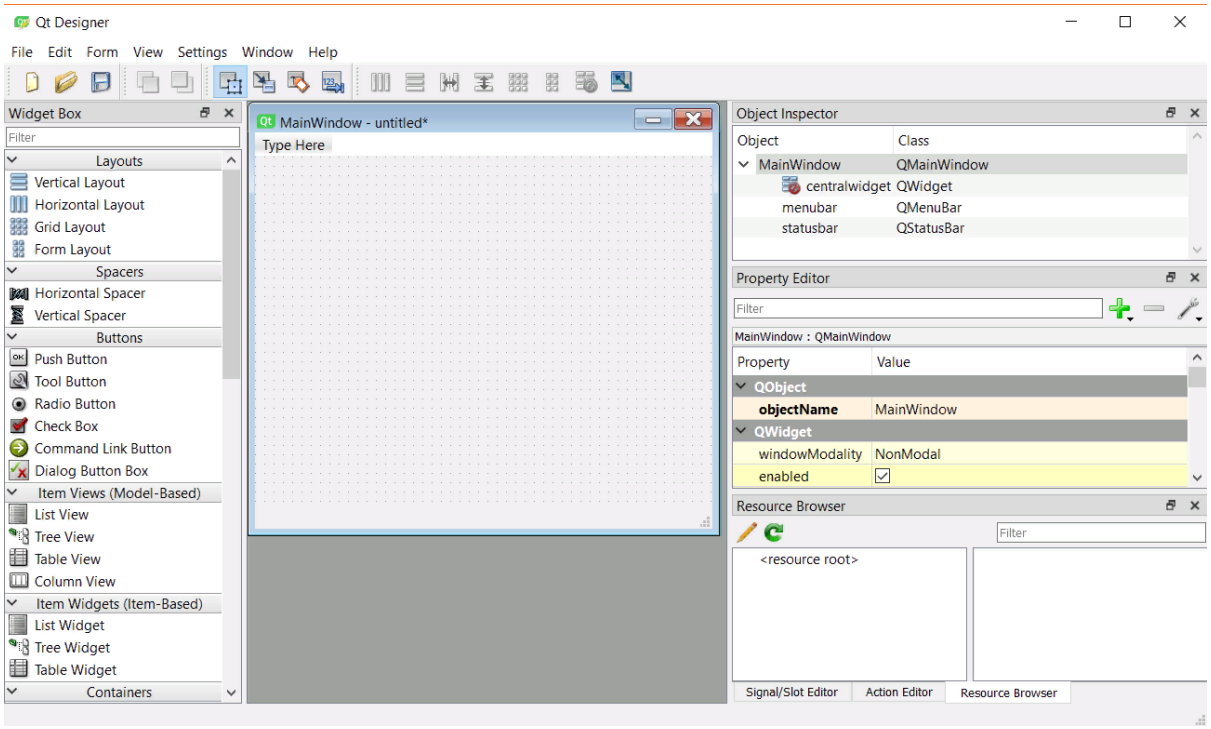
Создайте ярлык к этой программе, поскольку использовать её придётся часто.

Давайте запустим программу и посмотрим, что в ней можно сделать.



При запуске открывается окно с предложением выбрать шаблон для формы или виджет, на основе которого мы будем делать свой интерфейс. Выберем **Main Window**. Откроется пустое

окно.



Теперь рассмотрим, что у нас есть кроме пустой формы, которую мы будем заполнять.

Слева — меню виджетов, **Widget Box**. В нём они сгруппированы в зависимости от их функционала. Отдельно кнопки, отдельно виджеты для ввода данных и так далее.

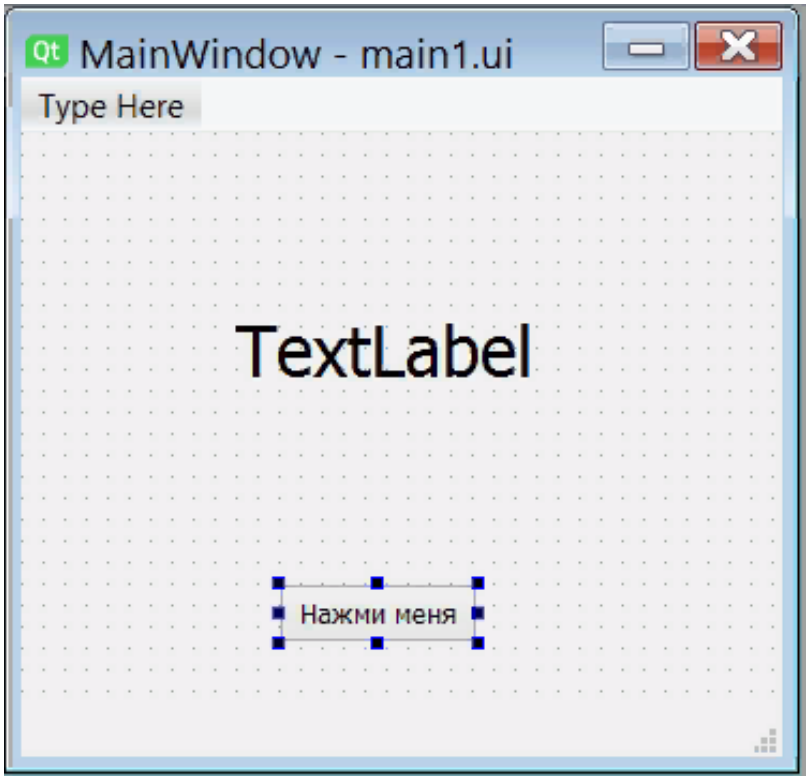
Справа — Инспектор Объектов (**Object Inspector**), Редактор Свойств (**Property Editor**) и Браузер Ресурсов (**Resourse Browser**). Остановимся на первых двух. В **Инспекторе Объектов** отображается информация об используемых виджетах. Для каждого виджета указывается его имя и класс. Кроме того, можно увидеть иерархическую структуру всего интерфейса.

Чтобы разместить виджет на форме, его надо просто перетащить из меню виджетов. При этом информация об этом виджете автоматически появится в Инспекторе Объектов.

Редактор свойств помогает изменять значения тех или иных атрибутов виджета (например, текст или размер). Расположение свойств меняется кнопкой с изображением гаечного ключа.

Попробуйте создать простейший интерфейс из кнопки (Push Button) и текстового поля (Text Label), поиграйтесь с **Редактором свойств**, разберитесь, где поменять название и размеры кнопки, как изменить шрифт в TextLabel. После того, как у вас вышло что-то похожее на картинку, ниже сохраните полученный дизайн. Для сохранения в меню «Файл» выберите вкладку «Сохранить как», найдите папку вашего проекта и впишите имя.

Помните, что название объекта (атрибут objectName) и текст, который может быть на нём показан (атрибут text) — это разные вещи.



Давайте посмотрим, как выглядит наш дизайн с точки зрения компьютера. Для этого мы должны открыть созданный нами файл с помощью любого текстового редактора. Лучше использовать не просто Блокнот, а, например, SublimeText. Когда мы откроем наш файл, то увидим, что это просто XML-документ, описывающий всю структуру GUI.

2. Как подключить интерфейс к основной программе

Теперь у нас есть дизайн, но нам надо подключить его к программе. Для этого есть два способа.

Способ первый, загрузка ui-файла

```
import sys
from PyQt5 import uic
from PyQt5.QtWidgets import QApplication, QWidget, QMainWindow


class MyWidget(QMainWindow):
    def __init__(self):
        super().__init__()
        uic.loadUi('main1.ui',self)
        self.pushButton.clicked.connect(self.run)

    def run(self):
        self.label.setText("OK")
```

```
app = QApplication(sys.argv)
ex = MyWidget()
ex.show()
sys.exit(app.exec_())
```

Для загрузки ui-файла импортируйте класс **uic**, а затем в конструкторе вызовите метод **loadUi** , где одним из параметров указывается файл с интерфейсом. В нашем примере он называется **main1.ui** и лежит в той же папке, что и запускаемый нами скрипт.

После выполнения метода **loadUi**, все виджеты становятся полями класса, имена для которых мы задали в редакторе свойств. Затем можно работать с ними точно так же, как в предыдущем уроке. Что мы и делаем, подключая обработчик нажатия кнопки.

Способ второй, использование pyuic

Независимо от способа подключения, файл UI конвертируется в класс на Python. В первом случае это происходит при запуске (метод **loadUi**), но замедляет программу. Поэтому часто файл конвертируют заранее, вручную.

Для этого нужна консольная утилита **pyuic5**.

Для того, чтобы ей воспользоваться, нужно открыть командную строку (терминал), перейти в ту папку, где лежит ваш ui-файл, и выполнить следующую команду:

```
pyuic5 ui_file.ui -o ui_file.py
```

В той же папке появится Python-овский файл с классом, аналогичным тому, что мы писали в первом уроке. Но в нём нет никаких функций, и если мы попробуем его запустить — ничего не произойдёт.

```
import sys
from PyQt5.QtWidgets import QApplication, QWidget, QMainWindow
from ui_file import Ui_MainWindow


class MyWidget(QMainWindow,Ui_MainWindow):
    def __init__(self):
        super().__init__()
        self.setupUi(self)
        self.pushButton.clicked.connect(self.run)

    def run(self):
        self.label.setText("OK")
```

```
app = QApplication(sys.argv)
ex = MyWidget()
ex.show()
sys.exit(app.exec_())
```

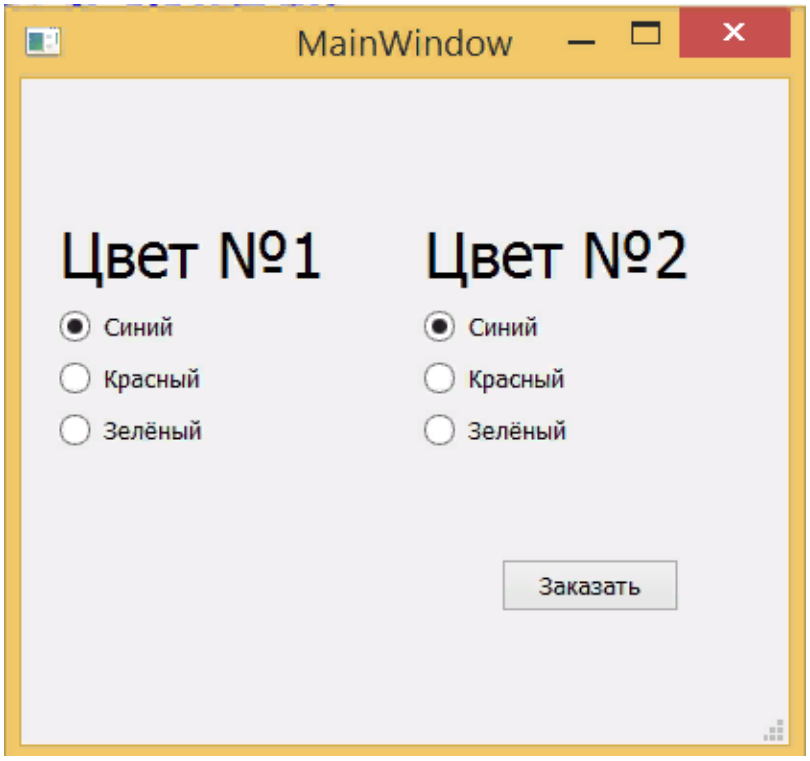
Здесь вы столкнулись с множественным наследованием. Предками нашего класса являются и **QMainWindow**, и **Ui_MainWindow**. От первого унаследованы методы, а от второго — дизайн. В остальном работа схожа: мы вызываем метод "setupUi", а затем просто работаем с полями.

3. Размещение виджетов

В Qt Designer можно размещать виджеты на экране не хаотично, а упорядоченно. Этот процесс называется разметкой.

Для этого есть виджеты, которые называются **Layout** (Разметка): Vertical, Horizontal, Grid и Form Layout.

Но они нужны не только для красивого размещения элементов интерфейса, но и для Radio Button. Когда мы работаем с ними, то можем выбрать только одну из кнопок. А что делать в том случае, когда у нас несколько факторов, которые нужно выбирать? Если мы просто разместим все Radio Button на нашем виджете, то никак не сможем выбрать два. А вот если часть из них поместить в какой-нибудь Layout — легко.



Кроме Layout, есть ещё группы. Чаще всего они нужны, когда у нас есть много виджетов с одинаковым функционалом. Например, у нас много Radio Button, но они все отвечают за один и тот же параметр — цвет. Чтобы объединить Radio Button в группу, необходимо выделить их, нажать на правую кнопку мыши, кликнуть пункт меню **Assign to button group** и выбрать либо **New button group**, либо уже созданную. Чтобы «повесить» на всю группу обработчик событий, нужно вызвать уже знакомый нам метод "connect". Но теперь он применяется не к одной кнопке, а сразу к группе.

```
self.buttonGroup.buttonClicked.connect(self.run)
```

А чтобы получить список всех кнопок группы, примените метод **.buttons()**.