QT6 ????????? ????

Компоновщики (Layouts).

Нужны для автоматического упорядочивания и изменения размеров виджетов при изменении размера окна. Без Layouts виджеты имеют фиксированные позиции и размеры. Импортируются все типы компоновщиков через

```
from PyQt6.QtWidgets import QHBoxLayout
```

Типы Layouts в Qt Designer

Vertical Layout (Вертикальный компоновщик)	Располагает виджеты сверху вниз в столбик.
	`layout = QVBoxLayout()`
	`layout.addWidget(button1)`
	`layout.addWidget(button2)`

Horizontal Layout (Горизонтальный компоновщик)	Располагает виджеты слева направо в строку. `QHBoxLayout()`

Grid Layout (Сеточный компоновщик)	Располагает виджеты в таблице (строках и столбцах).
	`layout = QGridLayout()`
	`layout.addWidget(button1, 0, 0)  # строка 0, столбец 0`
	`layout.addWidget(button2, 0, 1)  # строка 0, столбец 1`

Form Layout (Формовый компоновщик)	Идеален для форм (метка + поле ввода). Располагает виджеты в две колонки: labels слева, поля справа. `QFormLayout()`

```
class Window(QWidget):
    def __init__(self):
        super().__init__()
```

```
        self.setGeometry(200,200, 700, 400)
        self.mainlayout = QVBoxLayout()

        bt1 = QPushButton("one")
        bt2 = QPushButton("two")
        bt3 = QPushButton("three")
        bt4 = QPushButton("four")

        self.mainlayout.addWidget(bt1)
        self.mainlayout.addWidget(bt2)
        self.mainlayout.addWidget(bt3)
        self.mainlayout.addWidget(bt4)

        self.setLayout(self.mainlayout)
```

Базовый подход:

- Перетащите виджеты на форму
- Выделите несколько виджетов (зажав Ctrl)
- Нажмите правой кнопкой → "Lay out" → Выберите нужный тип
- Или используйте панель инструментов сверху (кнопки с иконками Layouts)

Вложенные Layouts:

```
Main Vertical Layout
├── Horizontal Layout (для кнопок)
│   ├── Кнопка "Открыть"
│   ├── Кнопка "Сохранить"
│   └── Кнопка "Выход"
└── Text Edit (занимает оставшееся пространство)
```

Еще класс, просто для примера. Соотношение 1:2

```
class Window(QWidget):
    '''
    Вложенные Layout
    '''
    def __init__(self):
        super().__init__()
        self.setGeometry(200,200, 700, 400)
```

```
        self.setWindowTitle("Python GUI Development")
        self.mainlayout = QVBoxLayout()

        lbl1 = QLabel("one label")
        lbl2 = QLabel("two label")

        bt1 = QPushButton("one")
        bt2 = QPushButton("two")

        self.innerlayout1 = QHBoxLayout()
        self.innerlayout1.addWidget(lbl1, stretch=1)
        self.innerlayout1.addWidget(bt1,stretch=2)
        self.mainlayout.addLayout(self.innerlayout1)

        self.innerlayout2 = QHBoxLayout()
        self.innerlayout2.addWidget(lbl2, stretch=1)
        self.innerlayout2.addWidget(bt2,stretch=2)
        self.mainlayout.addLayout(self.innerlayout2)

        self.setLayout(self.mainlayout)
```

Растяжения (Stretch):

```
В коде: layout.addStretch()
В Designer: есть специальный виджет "Horizontal Spacer" / "Vertical Spacer"
```

Выравнивание: В Property Editor настраивается

- layoutStretch — пропорции растяжения
- alignment — выравнивание содержимого
- spacing — расстояние между виджетами, пустое место

```
self.mainlayout.addSpacing(100)
```

- margin — отступ от краев

Пожелания при использовании Layouts

- Всегда используйте центральный Layout для главного окна
- Не смешивайте абсолютное позиционирование и Layouts
- Тестируйте ресайз окна
- Используйте Spacers для гибких промежутков

- Настройте stretch factors для управления пропорциями

Stretch Factors

Stretch factor — числовое значение, определяющее пропорцию, в которой виджеты делят доступное пространство при растяжении окна.

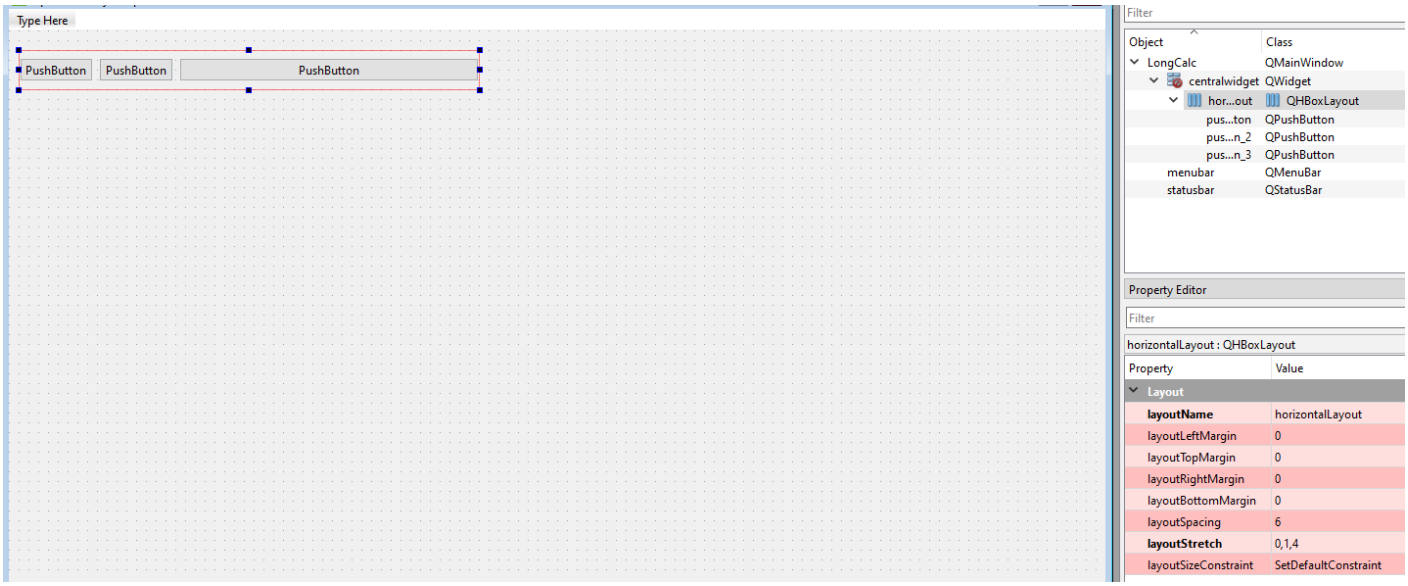
Базовые принципы:

- По умолчанию stretch factor = 0
- Виджет сохраняет свой минимальный размер
- Не растягивается при увеличении окна
- Чем больше значение, тем больше пространства получает виджет
- Важны относительные пропорции

Настройка при создании

Способ 1: Панель свойств (Property Editor)

- Выберите Layout в иерархии объектов
- Найдите свойство layoutStretch или layoutRowStretch / layoutColumnStretch
- Введите значения через запятую



В данном примере первая кнопка не будет менять размеры, вторая будет занимать 20% от оставшегося свободного места, третья 80%.

Способ 2:Через свойства виджетов внутри Layout

У каждого виджета есть свойство sizePolicy → horizontalStretch / verticalStretch

Способ 3: Через код

```
# при добавлении нового виджета
self.mainlayout.addWidget(bt1, stretch=1)
self.mainlayout.addWidget(bt2, stretch=3)
self.mainlayout.addWidget(bt3, stretch=0)
self.mainlayout.addWidget(bt4, stretch=5)
```

Настройка после создания

Способ 1: Использовать setStretch(index, stretch)

```
# Предположим, у вас уже есть Layout с виджетами
self.mainlayout.addWidget(bt1, stretch=1)
self.mainlayout.addWidget(bt2, stretch=3)
self.mainlayout.addWidget(bt3, stretch=0)
self.mainlayout.addWidget(bt4, stretch=5)

# Позже меняем stretch для bt4 (индекс 3, так как индексы начинаются с 0)
self.mainlayout.setStretch(3, 2)  # Меняем с 5 на 2
```

Способ 2: Получить индекс виджета динамически

```
# Находим индекс виджета bt4 в Layout
index = self.mainlayout.indexOf(bt4)
if index != -1:  # -1 означает "не найден"
    self.mainlayout.setStretch(index, 2)
```

Способ 3: Пересоздать Layout (более кардинальный)

```
# Удаляем все виджеты из Layout
while self.mainlayout.count():
    item = self.mainlayout.takeAt(0)
    if item.widget():
        item.widget().hide()

# Добавляем заново с новыми stretch factors
self.mainlayout.addWidget(bt1, stretch=1)
self.mainlayout.addWidget(bt2, stretch=3)
self.mainlayout.addWidget(bt3, stretch=0)
self.mainlayout.addWidget(bt4, stretch=2)  # Новое значение!
```

Способ 4: Изменить через setSizePolicy виджета.

```
# Получаем текущую политику размеров
policy = bt4.sizePolicy()

# Устанавливаем горизонтальный/вертикальный stretch
policy.setHorizontalStretch(2)  # Для Horizontal Layout
policy.setVerticalStretch(2)    # Для Vertical Layout

bt4.setSizePolicy(policy)
bt4.update()  # Обновляем виджет
```

Важно: Этот метод влияет на поведение виджета во всех Layout, где он находится!

Способ 5: Временное отключение обновления для избежания мерцания при изменении:

```
# Блокируем обновление
self.setUpdatesEnabled(False)

# Меняем stretch
index = self.mainlayout.indexOf(bt4)
self.mainlayout.setStretch(index, 2)

# Включаем обновление и форсируем перерасчет
self.setUpdatesEnabled(True)
self.mainlayout.invalidate()  # Помечаем Layout как невалидный
self.mainlayout.activate()    # Принудительно пересчитываем
```

Особенности работы:

- Изменения применяются немедленно — Layout пересчитается при следующем обновлении интерфейса
- Индексы начинаются с 0 и соответствуют порядку добавления
- Spacer'ы тоже имеют индексы — учитывайте это при поиске
- setStretch() работает только для QBoxLayout (QVBoxLayout, QHBoxLayout)
- Для QGridLayout используйте setRowStretch() и setColumnStretch()
- Для QFormLayout сложнее — лучше удалить и добавить заново

Проверка текущих значений:

```
# Получить текущий stretch factor
current_stretch = self.mainlayout.stretch(3)  # Для индекса 3
print(f"Текущий stretch: {current_stretch}")
```

```
# Получить список всех stretch factors
for i in range(self.mainlayout.count()):
    widget = self.mainlayout.itemAt(i).widget()
    stretch = self.mainlayout.stretch(i)
    if widget:
        print(f"Индекс {i}: {widget.text()} - stretch={stretch}")
```

Класс окна

Для управления классом окна, класс создается, затем настраиваются нужные свойства

```
from PyQt6.QtWidgets import QApplication, QWidget
from PyQt6.QtGui import QIcon
import sys

class Window(QWidget):
    def __init__(self):
        super().__init__()
        self.setGeometry(200,200, 700, 400)

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

Свойства:

self.setWindowIcon(QIcon('pyqt6lessons\images\python.png'))
self.setStyleSheet('background-color:green')
self.setWindowOpacity(0.5)

Свойство	Применение
Размеры окна	self.setGeometry(x,y, height, width) self.setGeometry(200,200, 700, 400)
Заголовок окна	self.setWindowTitle("Python GUI Development")

Revision #5
Created 28 January 2026 05:39:09 by Admin
Updated 29 January 2026 06:57:04 by Admin