



# PyQt:

## GUI на Python при помощи Qt

# Введение

- **Qt** («Кьют») — кроссплатформенный фреймворк для разработки приложений (в первую очередь на C++).
- **PyQt** — расширение для Python, обеспечивающее привязку к фреймворку Qt.



- И то, и другое — строго говоря, не только графический интерфейс, но и доступ к БД, поддержка локализации интерфейса, интеграция с HTML-движком WebKit, поддержка воспроизведения видео и аудио и кое-что ещё.

# История создания PyQt

- Первый релиз **PyQt** вышел ещё в 1998 году (ср. Python — в 1991) 
- Последняя версия на данный момент — **PyQt5** (на базе Qt 5). Только в ней добавлена (в дополнение к поддерживавшимся уже в PyQt4 Windows, Linux и Mac OS X) также поддержка Android и iOS.
- Тем не менее, **PyQt4** остаётся пока, похоже, более популярной, и под неё гораздо больше тьюториалов и справок, есть книжки и т.д.

# Установка PyQt5

- Устанавливается через **PIP**:

*pip install PyQt5*

(или *pip3 install PyQt5*, если у Вас Питон обеих версий)

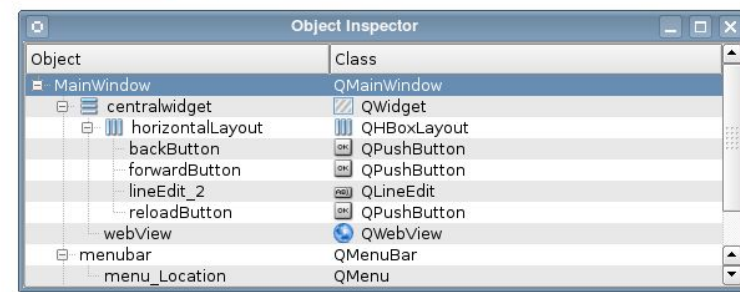
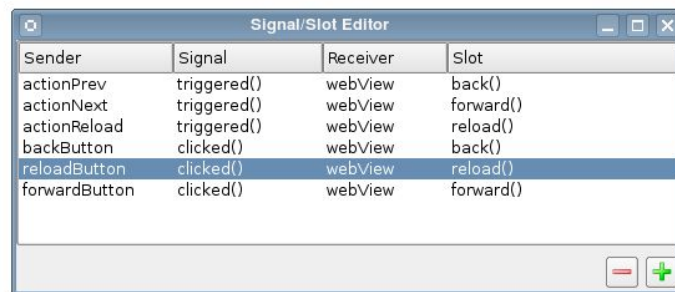
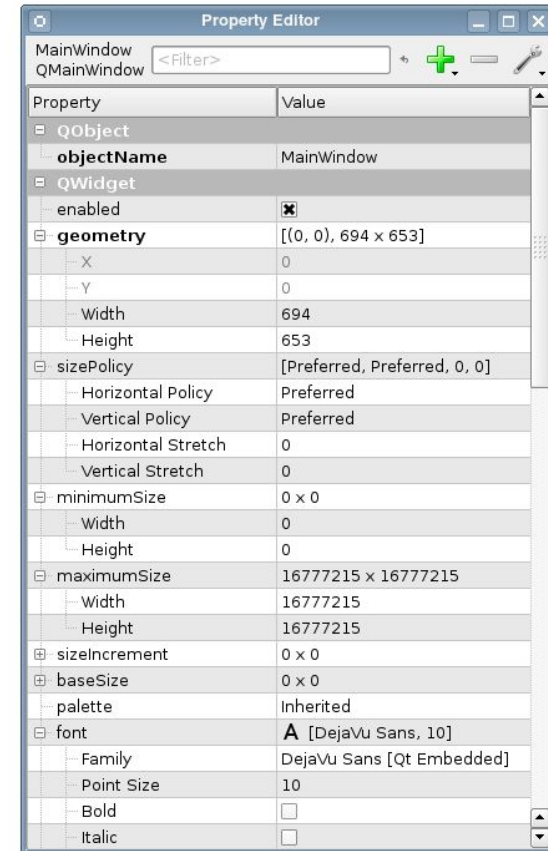
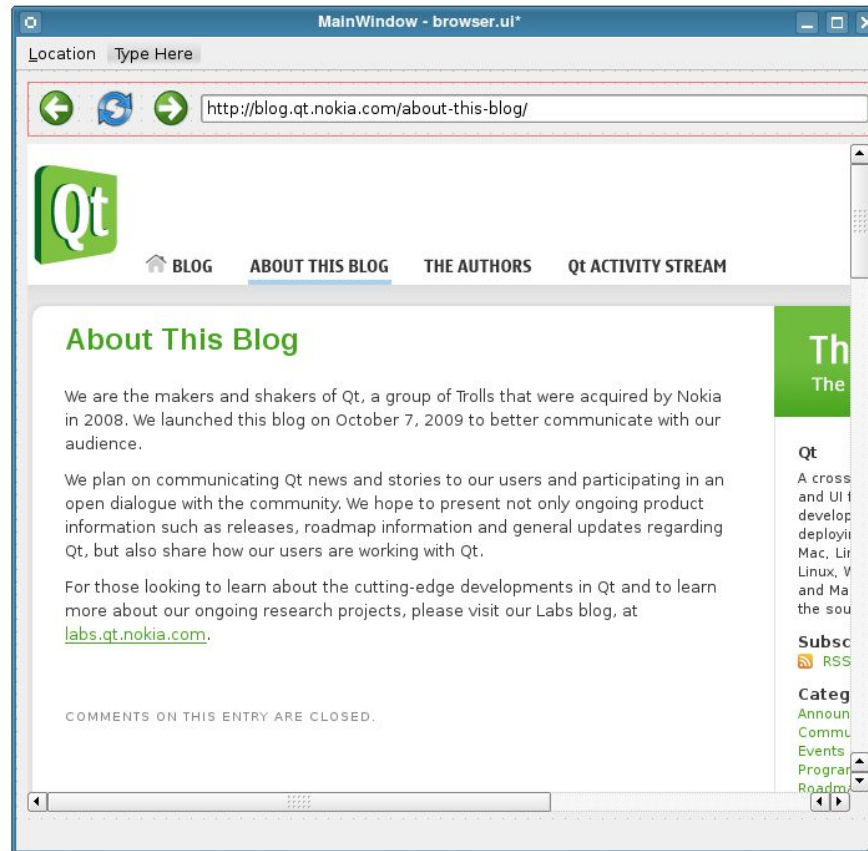
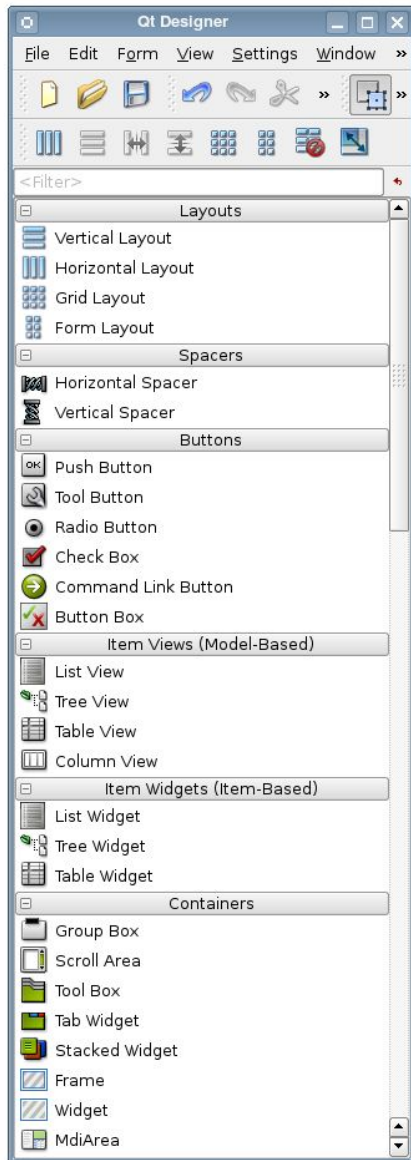
# Создание окна с помощью программы Qt Designer

- Если вы ранее пользовались Visual Studio или Delphi, то вспомните, что размещение компонентов на форме производили с помощью мыши. Щелка.пи левой кнопкой мыши на соответствующей кнопке на панели инструментов и перетаскивали компонент на форму. Далее с помощью инспектора свойств производили настройку значений некоторых свойств, а остальные свойства получали значения по умолчанию. При этом весь код генерировался автоматически. Произвести аналогичную операцию в PyQt позволяет программа Qt Designer, которая входит в состав установленных компонентов.

# Qt Designer

- В PyQt есть интеграция с Qt-шной программой **Qt Designer** (Qt Creator) (дизайнер графического интерфейса пользователя)
- при помощи приложения *pyuic* можно преобразовывать файлы Qt Designer в код на Python.

# Qt Designer





# Приложения с оконным интерфейсом

- NB! для приложений с оконным графическим интерфейсом используется расширение файлов не *.py*, а *.pyw*
  - при их запуске не открывается отдельным окном консоль Питона

# “Hello, World!” на PyQt5

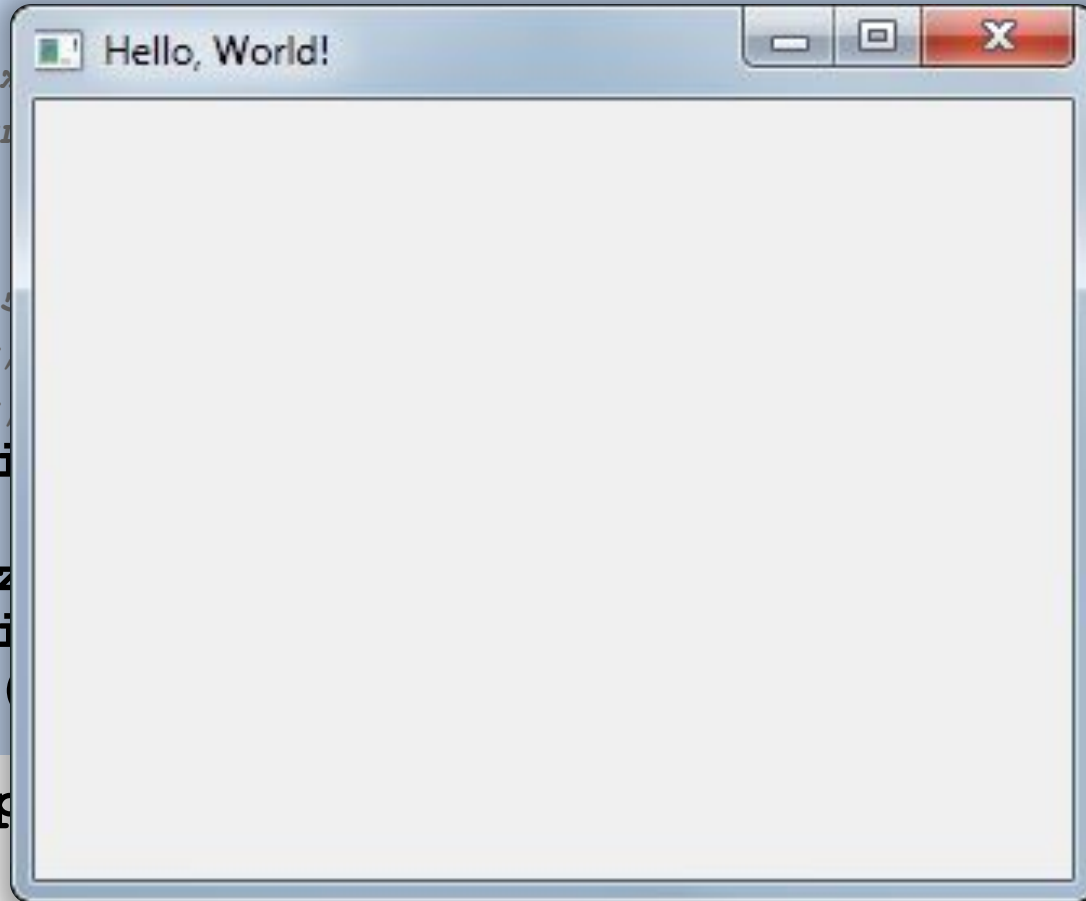
```
import sys
from PyQt5.QtWidgets import QApplication, QWidget
```

```
# Каждое приложение
# sys.argv - список
application =
```

```
# QWidget - базовый
# пользовательский
# без родителя
widget = QWidget
```

```
widget.resize(400, 300)
widget.setWindowTitle('Hello, World!')
widget.show()
```

```
sys.exit(application.exec_())
```



вить заголовок

кла приложения

# “Hello, World!” на PyQt4

```
import sys
from PyQt4.QtGui import QApplication, QWidget

# Каждое приложение должно создать объект QApplication
# sys.argv - список аргументов командной строки
application = QApplication(sys.argv)

# QWidget - базовый класс для всех объектов интерфейса
# пользователя; если использовать для виджета конструктор
# без родителя, такой виджет станет окном
widget = QWidget()

widget.resize(320, 240) # изменить размеры виджета
widget.setWindowTitle("Hello, World!") # установить заголовок
widget.show() # отобразить окно на экране

sys.exit(application.exec_()) # запуск основн. цикла приложения
```

# Добавляем кнопку

```
import sys
from PyQt5.QtWidgets import QApplication, QWidget,
                                QPushButton

application = QApplication(sys.argv)

widget = QWidget()

widget.resize(200, 120)
widget.setWindowTitle("Button test")

btn = QPushButton('Close me', widget) # добавляем кнопку
btn.clicked.connect(QApplication.instance().quit)
# присоединяем к ней метод, который будет выполняться при нажатии
btn.resize(btn.sizeHint())
# устанавливаем размер кнопки;
# sizeHint() подстраивает размер под текст
btn.move(50, 50) # устанавливаем расположение кнопки в окне

widget.show() # только потом показываем окно!
```

# Вызываем свой метод кнопкой

```
import sys
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton,
                                QMessageBox

class MyWidget(QWidget): # создаём на основе стандартного свой виджет с
                           блэkdжеком и шлюхами

    def __init__(self):
        super().__init__()

        self.initUI()

    def initUI(self): # переопределяем стандартную инициализацию интерфейса
        self.resize(200, 120) # NB! теперь всё через self
        self.setWindowTitle("Button test 2")
```

# Вызываем свой метод кнопкой

```
btn = QPushButton('Pop up!', self)
btn.clicked.connect(self.pop_up_hello_world)
# вешаем на кнопку на кастомный метод (см. ниже)
btn.resize(btn.sizeHint())
btn.move(50, 50)

self.show()
```

```
def pop_up_hello_world(self):
```

```
# создаём свой метод (который будет вызываться кнопкой) ВНУТРИ нашего класса
# — тогда можно будет его повесить на кнопку
```

```
    QMessageBox.information(self, "Title", "Hello, World!")
```

```
# выплёваем окошко на экран с заданным заголовком и текстом
```

```
if __name__ == '__main__':
```

```
    app = QApplication(sys.argv)
```

```
    my_widget = MyWidget() # создаём экземпляр нашего виджета, и он
                           # запускается
```

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

# PyInstaller

- Приложение, использующее PyQt, можно потом ещё и собрать вместе с интерпретатором и всеми модулями в **исполняемый файл** (не требующий отдельной установки всего этого) для Windows / Linux / Mac OS и др.  
при помощи **PyInstaller**

# ООП-стиль создания окна

- Библиотека PyQt написана в объектно-ориентированном стиле (ООП-стиле) и содержит более 600 классов. Иерархия наследования всех классов имеет слишком большой размер, поэтому приводить ее в книге нет возможности. Тем не менее, чтобы показать зависимости, при описании компонентов иерархия наследования конкретного класса будет показываться.





Спасибо