Информатика

Баранов М.А.

Осень 2023

Обзор графических библиотек

- Tkinter;
- wxPython;
- PyGTK;
- PyQT/PyKDE;
- Pythonwin;
- pyFLTK;
- AWT, JFC, Swing;
- Anygui;
- PythonCard.

О PyQt/PySide

- **PyQt/PySide** — набор расширений (биндингов, привязок) на уровне API графического фреймворка Qt для Python.
- Фреймворки практически полностью реализует возможности Qt. Это более 600 классов, более 6000 функций и методов, включая:
  - набор виджетов графического интерфейса;
  - стили виджетов;
  - доступ к базам данных с помощью SQL (ODBC, MySQL, PostgreSQL, Oracle);
  - поддержку интернационализации (i18n);
  - парсер XML;
  - интеграцию с WebKit, движком рендеринга HTML;
  - поддержку воспроизведения видео и аудио.
- Существует 3 версии: PyQt6, PyQt5 и PyQt4 (PySide6, PySide2, PySide), поддерживающие соответствующие версии Qt.

Доступные платформы.

Платформа	Описание
Linux/Unix	
X11	Qt для оконного менеджера X (Linux, FreeBSD, HP-UX, Solaris, AIX, и т. д.)
Wayland	Qt для Wayland.
Встраиваемые Linux-системы	Qt для встраиваемых систем: КПК, смартфонов, и т. д.
Android	Qt для Android, ранее известный как Necessitas.
Платформы Apple	
OS X	Qt для Apple OS X; поддерживает приложения на Cocoa.
iOS	Qt для iOS платформ (iPhone, iPad).
Платформы Microsoft	
Windows	Qt для Microsoft Windows XP, Vista, 7, 8 и 10.
Windows CE	Qt для Windows CE 6 и Windows Embedded Compact 7.
Windows RT	Поддержка для основанных на WinRT приложениях для Windows 8 и Windows Phone 8.

Различия в поставке.

Категория/Фреймворк	PyQt	PySide
Лицензия	GPL или коммерческая лицензия	LGPL
Версия Qt	v5.15.6/v6.0.2 (PyQt6)	v5.15.2/v6.2.1+(PySide6)
Платформа	Python 3+	Python 3 и Python 2.7 (только для Linux и MacOS)
Первый стабильный выпуск	Апрель 2016/Январь 2021	Июль 2018/Декабрь 2020
Состав пакета	Основное ядро ~50Mb	Полный пакет со всеми инструментами ~120Mb

Пишите открытое/свободное ПО -- можно использовать как PyQt5, так и PySide 2.o
Пишите закрытое/коммерческое ПО -- бесплатно можно использовать только PySide 2, а для использования PyQt5 потребуется покупать коммерческую лицензию.

Полная поставка включает в себя Qt Designer — дизайнер графического интерфейса пользователя. Программу pyuic — генерирует Python код из файлов, созданных в Qt Designer. Это делает PyQt очень полезным инструментом для быстрого прототипирования. Кроме того, можно добавлять новые графические элементы управления, написанные на Python, в Qt Designer.
Программа QTranslator — служит для локализации интерфейса.

Различия в коде.

- 1. Конвертация Ui – файлов в Py:

```
pyside2-uic mainwindow.ui -o MainWindow.py
pyuic5 mainwindow.ui -o MainWindow.py
```

  - Pyside2
  - PyQt5

- 2. Использование exec() или exec_():

```
if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    myapp = FirstForm()
    myapp.show()
    sys.exit(app.exec_())
```

  - PySide2 – только exec_(), т.к. присутствует поддержка Python 2;
  - PyQt5 – возможны оба варианта, т.к. в Python 3 “*exec*” не является ключевым/ зарезервированным словом.

- 3. Слоты и сигналы:

```
from PySide2.QtCore import Signal, Slot  # для PySide2
from PyQt5.QtCore import pyqtSignal, pyqtSlot  # для PyQt5

myCustomPySide2Signal = Signal()  # для PySide2
myCustomPyQt5Signal = pyqtSignal()  # для PyQt5

@Slot  # для PySide2
def myCustomPySide2Slot():
    pass

@pyqtSlot  # для PyQt5
def myCustomPyQt5Slot():
    pass
```

Обеспечение обратной совместимости:

```
import pip._internal.operations.freeze

for requirement in pip._internal.operations.freeze.freeze(local_only=True):
    print(requirement)
    if 'PyQt5' in requirement:  # PyQt5
        print("Work with PyQt5")
        from PyQt5 import QtGui, QtWidgets, QtCore
        from PyQt5.QtCore import pyqtSignal as Signal, pyqtSlot as Slot
        break

    elif 'PySide2' in requirement:  # PySide2
        print("Work with PySide2")
        from PySide2 import QtGui, QtWidgets, QtCore
        from PySide2.QtCore import Signal, Slot
        break
```

Модули Qt.

QtCore	**QtGui**	**QtWidgets**
QtConcurrent	QtHelp	QtNetwork
QtOpenGL	QtOpenGLFunctions	QtOpenGLWidgets
QtPrintSupport	QtQml	QtQuick
QtQuickControls2	QtQuickWidgets	QtSql
QtSvg	QtSvgWidgets	QtTest
QtUiTools	QtXml	

- **QtCore** – основные функции, не связанные с графическим интерфейсом.
- **QtGui** – расширяет функциональность графического интерфейса.
- **QtWidgets** – работа с виджетами Qt.
- **QtConcurrent** – высокоуровневое API для работы с потоками.
- **QtHelp** – интеграция онлайн-документации в приложения.
- **QtNetwork** – позволяет писать клиент-серверные (TCP/IP) приложения.
- **QtOpenGL/QtOpenGLFunctions/QtOpenGLWidgets** –работа с 2D/ 3D графикой.
- **QtPrintSupport** – класс обеспечивающий поддержку печати.

- **QtQml/QtQuick/QtQuickControls2/QtQuickWidgets** – API для использования Qt QML (Qt Meta/Modeling Language) и создания настраиваемых высокодинамичных графических пользовательских интерфейсов с плавными переходами и эффектами.
- **QtSql** – содержит драйвера для обеспечения интеграции БД приложением.
- **QtSvg/QtSvgWidgets** – для работы с файлами SVG.
- **QtTest** – классы для модульного тестирования.
- **QtUiTools** – служит для обработки форм, созданных с помощью Qt Designer.
- **QtXml** – обеспечивает работу с потоками чтения и записи XML документов и реализацию их в форме SAX (Simple API for XML) и DOM (Document Object Model)

Создание окна через QMainWindow:

Код для примера, запустить окно можно и без него

```python
import sys
from PySide2 import QtWidgets


class MyWidgets(QtWidgets.QMainWindow):

    def __init__(self, parent=None):
        QtWidgets.QMainWindow.__init__(self, parent)
        # menuBar отсутствует у QWidgets
        self.fileMenu = self.menuBar().addMenu('File')
        self.fileMenu.addAction("Open")
        # toolBar отсутствует у QWidgets
        self.toolBar = self.addToolBar("Edit")
        self.toolBar.addAction("Edit")
        # statusBar отсутствует у QWidgets
        self.appStatusBar = self.statusBar()
        self.appStatusBar.showMessage("Status: Ok!")

if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)

    myWindow = MyWidgets()
    myWindow.show()

    app.exec_()
```

Создание окна через QWidget:

```python
import sys
from PySide2 import QtWidgets


class MyWidgets(QtWidgets.QWidget):

    def __init__(self, parent=None):
        QtWidgets.QWidget.__init__(self, parent)


if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)

    myWindow = MyWidgets()
    myWindow.show()

    app.exec_()
```

Варианты добавления виджетов

- Напрямую:

```
self.pushButton = QtWidgets.QPushButton("Кнопка", self)
```

- Через компоновку:

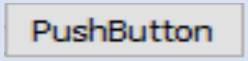
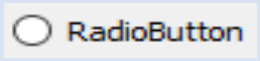
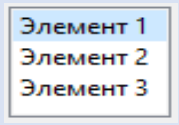
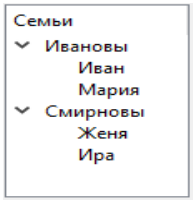
QMainWindow

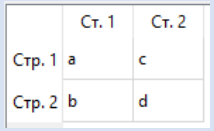
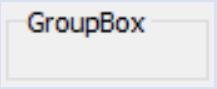
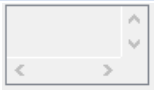
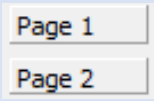
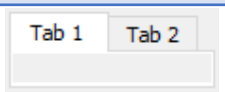
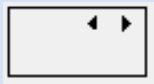
```
centralWidget = QtWidgets.QWidget()
self.setCentralWidget(centralWidget)
layout = QtWidgets.QHBoxLayout()
self.abc = QtWidgets.QPushButton("abc")
layout.addWidget(self.abc)

centralWidget.setLayout(layout)
```

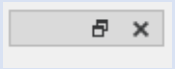
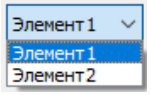
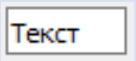
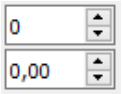
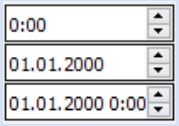
QWidget

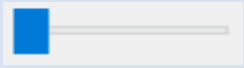
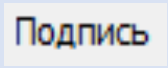
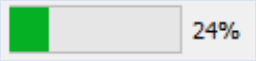
```
layout = QtWidgets.QHBoxLayout()
self.abc = QtWidgets.QPushButton("abc")
layout.addWidget(self.abc)
self.setLayout(layout)
```

- Через QtDesigner.

Внешний вид	Название	Примечание	Наследование
	QPushButton	Командная кнопка.	QWidget -> QAbstractButton -> QPushButton
	QToolButton	QTollButton. **(Создается в QToolBar)**	QWidget -> QAbstractButton -> QToolButton
	QRadioButton	Переключатель.	QWidget -> QAbstractButton -> QRadioButton
	QCheckBox	«Флажок»	QWidget -> QAbstractButton -> QCheckBox
	QCommandLinkButton	Ссылка на команду	QWidget -> QAbstractButton -> QPushButton -> QCommandLinkButton
	QDialogButtonBox	Макет группы кнопок	QWidget -> QDialogButtonBox
	QListView QListWidget QUndoView	Список элементов	QWidget -> QFrame -> QAbstractScrollArea -> QAbstractItemView -> QListView -> QListWidget/QUndoView
	QTreeView QTreeWidget	Древовидный список элементов	QWidget -> QFrame -> QAbstractScrollArea -> QAbstractItemView -> QTreeView -> QTreeWidget

Внешний вид	Название	Примечание	Наследование
	QTableView QTableWidget	Таблица	QWidget -> QFrame -> QAbstractScrollArea -> QAbstractItemView -> QTableView -> QTableWidget
	QColumnView	Таблица столбцов	QWidget -> QFrame -> QAbstractScrollArea -> QAbstractItemView -> QColumnView
	QGroupBox	Группировка виджетов с заголовком	QWidget -> QGroupBox
	QScrollArea	Зона с прокруткой	QWidget -> QFrame -> QAbstractScrollArea -> QScrollArea
	QToolBox	«Аккордеон»	QWidget -> QFrame -> QToolBox
	QTabWidget	Зона с вкладками	QWidget -> QTabWidget
	QStackedWidget	Стек окон	QWidget -> QFrame -> QStackedWidget
	QFrame	Группировка виджетов	QWidget -> QFrame

Внешний вид	Название	Примечание	Наследование
	QWidget	Базовый класс	QObject & QPaintDevice -> QWidget
	QMdiArea	Окна MDI (многодокументный интерфейс)	QWidget -> QFrame -> QAbstractScrollArea -> QMdiArea
	QDockWidget	«Плавающее» окно	QWidget -> QDockWidget
	QComboBoxWidget QFontComboBox	Выпадающий список	QWidget -> QComboBox -> QFontComboBox
	QLineEdit QTextEdit QPlainTextEdit QTextBrowser	Зона редактирования текста	QWidget -> QLineEdit QWidget -> QFrame -> QAbstractScrollArea -> QTextEdit QWidget -> QFrame -> QAbstractScrollArea -> QPlainTextEdit QWidget -> QFrame -> QAbstractScrollArea -> QTextEdit -> QTextBrowser
	QSpinBox QDoubleSpinBox	Счётчик с увеличением	QWidget -> QAbstractSpinBox -> QSpinBox QWidget -> QAbstractSpinBox -> QDoubleSpinBox
	QTimeEdit QDateEdit QDateTimeEdit	Работа с временем и датой	QWidget -> QAbstractSpinBox -> QDateTimeEdit -> QTimeEdit QWidget -> QAbstractSpinBox -> QDateTimeEdit -> QDateEdit QWidget -> QAbstractSpinBox -> QDateTimeEdit ->

Внешний вид	Название	Примечание	Наследование
	QDial	Контроль диапазона	QWidget -> QAbstractSlider -> QDial
	QScrollBar	Полоса прокрутки	QWidget -> QAbstractSlider -> QScrollBar
	QSlider	Слайдер	QWidget -> QAbstractSlider -> QSlider
	QKeySequenceEdit	Проверка нажатия	QWidget -> QKeySequenceEdit
	QLabel	Подпись	QWidget -> QFrame -> QLabel
	QGraphicsView	Отображение графики (объектов QGraphicsScene)	QWidget -> QFrame -> QAbstractScrollArea -> QGraphicsView
	QCalendarWidget	Календарь	QWidget -> QCalendarWidget
	QLCDNumber	«Цифровое табло»	QWidget -> QFrame -> QLCDNumber
	QProgressBar	Шкала выполнения	QWidget -> QProgressBar
	QOpenGLWidget	Отображение 2D/3D	QWidget -> QOpenGLWidget

Смысл концепции

- Модель – «обёртка» над данными, которая позволяет управлять и взаимодействовать с ними.
- Представление – служит для отображения элементов на экране. Одну модель можно установить сразу в несколько представлений.
- Модель выделения – управляет выделением данных в модели.
- Промежуточная модель – «прослойка» между основной моделью и представлением. Служит для сортировки и фильтрации данных без изменения порядка следования элементов в базовой модели.
- Делегат – обеспечивает компонент для вывода и редактирования данных.

Виды моделей

- QStringListModel – список строк. Отображение через QListView, QComboBox;
- QStringItemModel – двумерная таблица. Отображение через QTableView, QTreeView.
- QStandardItem – создание элементов и вложенных структур.

Виды представлений

- QListView – простой список.
- QTableView – таблица.
- QTreeView – иерархический список.

Классы основных модулей Qt

- QtCore: https://doc.qt.io/qt-5/qtcore-module.html
- QtGui: https://doc.qt.io/qt-5/qtgui-module.html
- QtWidgets: https://doc.qt.io/qt-5/qtwidgets-module.html
- QtNetwork: https://doc.qt.io/qt-5/qtnetwork-module.html
- QtSQL: https://doc.qt.io/qt-5/qtsql-module.html

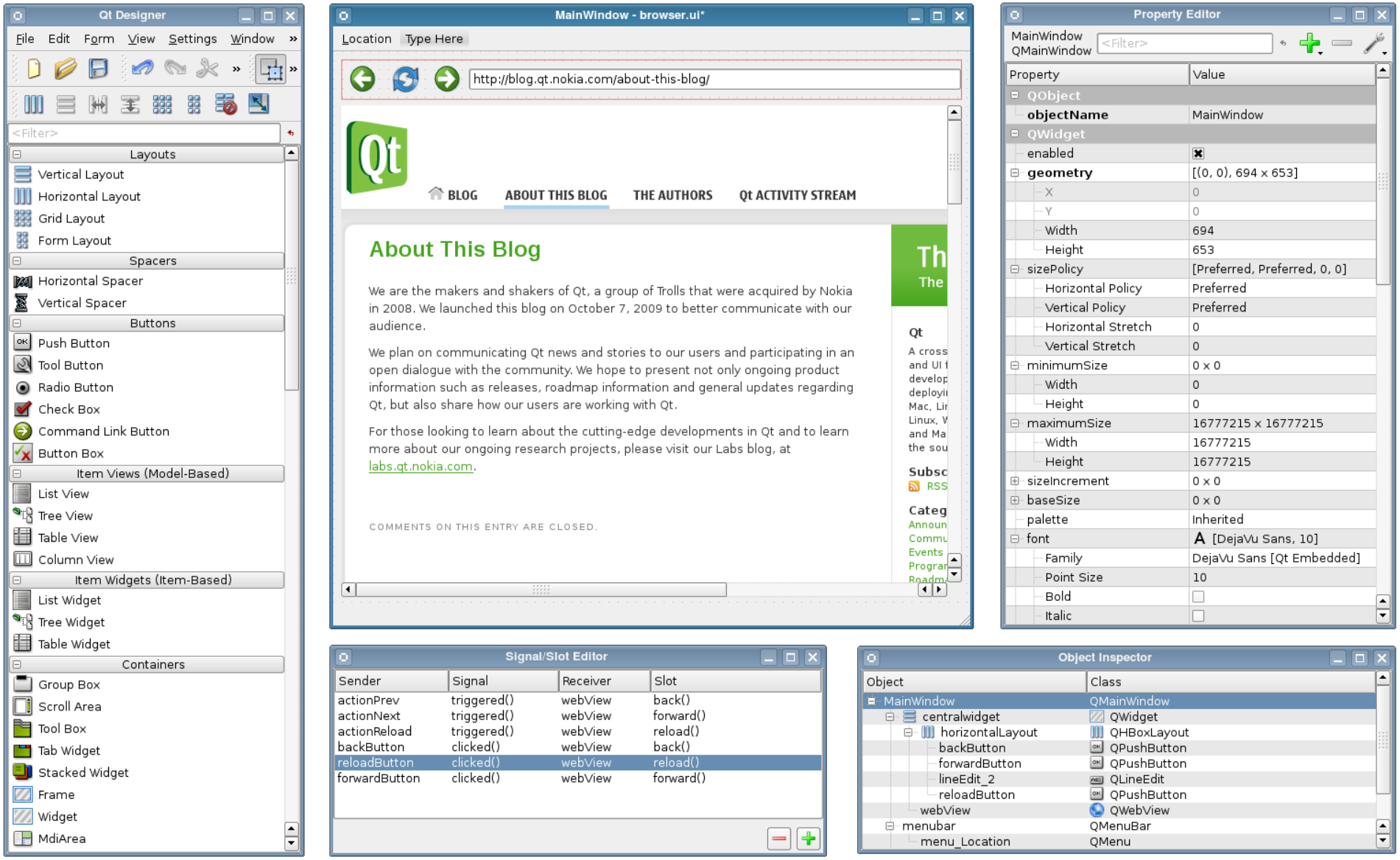
Qt Designer

- 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


Qt Designer
File Edit Form View Settings Window
<Filter>
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
Button Box
Item Views (Model-Based)
List View
Tree View
Table View
Column View
Item Widgets (Item-Based)
List Widget
Tree Widget
Table Widget
Containers
Group Box
Scroll Area
Tool Box
Tab Widget
Stacked Widget
Frame
Widget
MdiArea
MainWindow - browser.ui*
Location Type Here
http://blog.qt.nokia.com/about-this-blog/
BLOG
ABOUT THIS BLOG
THE AUTHORS
Qt ACTIVITY STREAM
About This Blog
We are the makers and shakers of Qt, a group of Trolls that were acquired by Nokia in 2008. We launched this blog on October 7, 2009 to better communicate with our audience.
We plan on communicating Qt news and stories to our users and participating in an open dialogue with the community. We hope to present not only ongoing product information such as releases, roadmap information and general updates regarding Qt, but also share how our users are working with Qt.
For those looking to learn about the cutting-edge developments in Qt and to learn more about our ongoing research projects, please visit our Labs blog, at labs.qt.nokia.com.
COMMENTS ON THIS ENTRY ARE CLOSED.
Property Editor
MainWindow
QMainWindow
<Filter>
Property
Value
QObject
objectName
MainWindow
QWidget
enabled
geometry
[(0, 0), 694 x 653]
X
Y
Width
694
Height
653
sizePolicy
[Preferred, Preferred, 0, 0]
Horizontal Policy
Preferred
Vertical Policy
Preferred
Horizontal Stretch
0
Vertical Stretch
0
minimumSize
0 x 0
Width
0
Height
0
maximumSize
16777215 x 16777215
Width
16777215
Height
16777215
sizeIncrement
0 x 0
baseSize
0 x 0
palette
Inherited
font
A [DejaVu Sans, 10]
Family
DejaVu Sans [Qt Embedded]
Point Size
10
Bold
Italic
Signal/Slot Editor
Sender
Signal
Receiver
Slot
actionPrev
triggered()
webView
back()
actionNext
triggered()
webView
forward()
actionReload
triggered()
webView
reload()
backButton
clicked()
webView
back()
reloadButton
clicked()
webView
reload()
forwardButton
clicked()
webView
forward()
Object Inspector
Object
Class
MainWindow
QMainWindow
centralwidget
QWidget
horizontalLayout
QHBoxLayout
backButton
QPushButton
forwardButton
QPushButton
lineEdit_2
QLineEdit
reloadButton
QPushButton
webView
QWebView
menubar
QMenuBar
menu_Location
QMenu

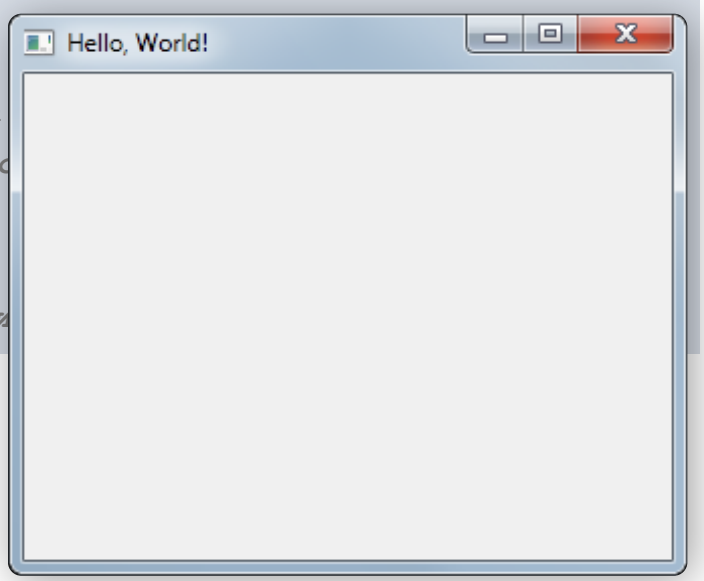
Приложения с оконным интерфейсом

- 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 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!") # устано
widget.show() # отобразить окно на экране

sys.exit(application.exec_()) # запуск основн. ци
```



“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 кнопочку

```
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!

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

Вызываем свой метод кнопочкой

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


class MyWidget(QWidget): # создаём на основе стандартного свой виджет с
                           блэкджеком 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 кнопочкой

```
        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_())
```

PyIstaller

- 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