Python. Подготовка к собеседованию

Урок 7

Фреймворк PyQt

Особенности использования фреймворка PyQt

Цели урока

Повторить особенности использования фреймворков PyQt и Kivy.

Назначение и установка PyQt, способы создания приложений.

Преимущества PyQt.

Основные PyQt-классы.

Различия между PyQt4 и PyQt5.

Назначение обработчика для сигнала, создание сигнала.

Передача данных в обработчик.

Взаимодействие с базами данных из PyQt.

PyQt, ее назначение и установка

Для Windows: Загружаем установщик с ресурса https://riverbankcomputing.com

Для Unix:

```
sudo apt-get install python3-pyqt4 pyqt4-dev-tools
```

или

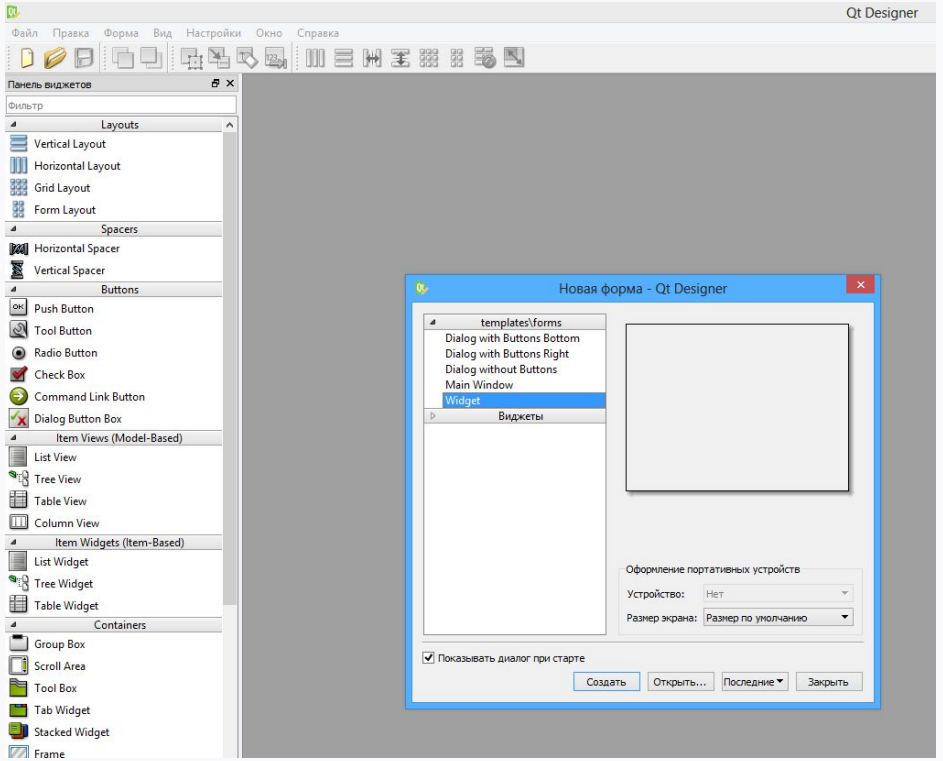
```
sudo apt-get install python3-pyqt5 pyqt5-dev-tools
```

Способы создания приложений на PyQt

Вручную:

```
from PyQt5 import QtCore, QtGui, QtWidgets
import sys

# Импортируем наш модуль интерфейса my.py
from my import *


class MyWin(QtWidgets.QMainWindow):
    def __init__(self, parent=None):
        QtWidgets.QWidget.__init__(self, parent)
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)

        # Здесь прописываем событие нажатия на кнопку
        self.ui.pushButton.clicked.connect(self.buttonpress)

    # Пока пустая функция которая выполняется
    # при нажатии на кнопку
    def buttonpress(self):
        # Здесь должны быть действия по нажатию кнопки
        # но пока здесь заглушка pass
        pass

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

С помощью Qt Designer:



Преимущества PyQt

- Средство реализации графических оболочек и механизмов взаимодействия с базами данных.
- Мощная библиотека классов-компонентов.
- Наличие хорошо структурированной документации.
- Наличие встроенной среды разработки Qt Designer.

Основные PyQt-классы

QtCore

QtGui

QtSql

QtNetwork

QtMultimedia

uic

QtWidgets

QtTest

QtPrintSupport

Различия между PyQt4 и PyQt5

- Проведена реорганизация модулей, часть модулей исключена.
- Реализован новый стиль обработки сигналов и слотов.
- Реализована поддержка нескольких новых модулей.
- Прекращена поддержка Python 2.6.

Назначение обработчика для сигнала

Обработчик-функция:	имя_компонента.сигнал.connect(имя_обработчика)
Обработчик-метод класса:	имя_компонента.сигнал.connect(экземпляр_класса.имя_метода_объекта_класса)
Обработчик-класс:	имя_компонента.сигнал.connect(имя_класса())
Обработчик-lambda-функция:	имя_компонента.сигнал.connect(lambda: имя_класса())

Создание пользовательского сигнала

Создание сигнала:

```
имя_сигнала = QtCore.pyqtSignal(имя_параметра, ...)
```

Привязка обработчика к сигналу:

```
имя_сигнала.connect(self.имя_обработчика_сигнала)
```

Обработчик сигнала:

```
def имя_обработчика_сигнала(self, имя_параметра, ...):
    # код обработчика
```

Генерация сигнала:

```
имя_сигнала.emit(имя_параметра, ...)
```

Передача данных в обработчик

Создание обертки в виде lambda-функции:

```
self.имя_компонента.сигнал.connect(
    lambda: self.имя_обработчика(передаваемый_параметр)
)
```

Передача ссылки на экземпляр класса с методом call():

```
self.имя_компонента.checked.connect(
    имя_класса(передаваемый_параметр)
)
```

Передача обработчиков и параметров функцию partial:

```
from functools import partial
self.chck_box.checked.connect(
    partial(self.имя_обработчика, передаваемый_параметр)
)
```

Взаимодействие с СУБД средствами PyQt (часть 1)

Класс QSqlDatabase

QMYSQL

QODBC

QPSQL

QSQLITE

Взаимодействие с СУБД средствами PyQt (часть 2)

Класс QSqlQuery - конструктор SQL-запроса

Класс QSqlQueryModel - конструктор модели на основе SQL-запроса

Класс QSqlTableModel - конструктор модели-таблицы

	id	firstname	lastname	archive
1	101	Danny	Young	true
2	102	Christine	Holand	false
3	103	Lars	Young	true
4	104	Roberto	Robitaille	false
5	105	Maria	Papadopoulos	true
6	106	Luisa	Papadopoulos	true
7	107	Luis	Papadopoulos	true

Практическое задание

Практическое задание

В рамках выполнения практического задания к данному уроку слушателям курса предлагается продолжить работу над настольным приложением с графическим интерфейсом пользователя для ведения простого складского учета.

1) Создать главное окно программы, для которого реализовать меню с шестью пунктами, верхний и центральный виджет.
2) Создать верхний виджет программы с фреймом, в который расположить три виджета: надпись (путь к базе данных), текстовое поле для отображения пути к базе данных (сделать его недоступным для редактирования), кнопке для открытия диалога выбора файла sqlite-базы данных.
3) В центральном виджете программы реализовать виджет с табличным компонентом-представлением (QTableView) и двумя кнопками (для добавления и удаления записи таблицы базы данных). В компоненте-представлении будут отображаться модели данных, соответствующие каждой из таблиц. Пользователь сможет добавлять и удалять записи.
4) К кнопке открытия диалога выбора файла sqlite-базы данных привязать обработчик нажатия, который должен открывать окно диалога для выбора файла базы данных (класс QFileDialog).
5) Для каждого из пунктов меню реализовать обработчик нажатия. Код обработчиков реализовать в отдельном модуле, который должен импортироваться в главное окно программы. К каждой из кнопок (добавления и удаления записей) привязать обработчики соответствующих событий. В компоненте-представлении QTableView должны отображаться изменения (т.е. таблицы с добавленными или без удаленных записей).

Дополнительные материалы

1. https://www.linux.org.ru/forum/development/10373013.
2. http://pyqt.sourceforge.net/Docs/PyQt4/qtsql.html.
3. http://doc.qt.io/qt-5/index.html.