PMR3201 Computação para Automação
Aula de Laboratório 5

Construção de Interfaces em PyQt5

Newton Maruyama
Thiago de Castro Martins
Marcos S. G. Tsuzuki

17 de maio de 2017

PMR-EPUSP

Conteúdo

1. *Framework* Qt

2. Alguns exemplos

Framework Qt

Qt

- Qt é um *framework* multiplataforma para o desenvolvimento de interfaces gráficas em C++ desenvolvido pela empresa Trolltech.
- O *framework* tem versões para praticamente todas as plataformas como: Windows, Windows CE, X11, OSX, QNX, Android, iOS, etc.
- Qt é mantido pelo Qt Project, uma iniciativa de software livre envolvendo desenvolvedores individuais e provenientes de empresas como Nokia, Digia e outras.
- PyQt é um *wrapper* para interfacear a linguagem Python com o *framework* Qt desenvolvido pela empresa Riverbank Computing.
- A versão atual é a PyQT5.8.2.

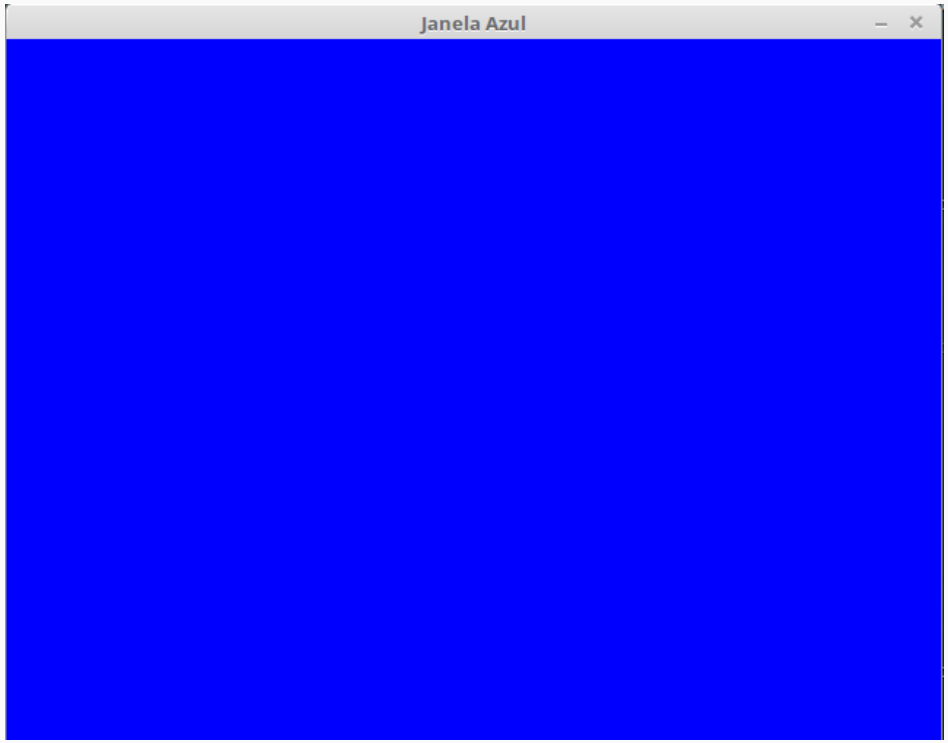
Framework

- Em linguagem mais simples o *Framework* Qt, ou no nosso caso, o PyQt permite criar uma GUI (*Graphical User Interface*).
- As funções da biblioteca "dialogam" com o Sistema Operacional enviando mensagens para abrir janelas e criar *Widgets* com funções específicas dentro das janelas.
- Podemos a partir disso criar aplicativos como Editores de Texto, Games, programas CAD, etc.

Alguns exemplos

Exemplo 1: Tela Azul - no it's not about Windows!

- Inicialmente vamos criar uma simples janela com fundo azul como abaixo:



- Essa janela não possui nenhuma função associada.
- O código se encontra no arquivo pyqt1.py.
- Carregue o programa no IDE Spyder e execute.

Exemplo 1: arquivo pyqt1.py

- O código é apresentado a seguir:

```
import sys
from PyQt5.QtGui import QPalette
from PyQt5.QtWidgets import QApplication, QMainWindow
from PyQt5.QtCore import Qt

def adeus(x):
    print("Adeus")

app = QApplication(sys.argv)
# Criando uma janela
window = QMainWindow()
window.setFixedSize(640,480)
window.setWindowTitle("Janela Azul")
# Setando as cores
cores = QPalette(app.palette(window))
cores.setColor(QPalette.Window, Qt.blue)
window.setPalette(cores)
# Tornando a janela visivel
window.show()
# Setando o que ocorre no fechamento da janela
window.closeEvent = adeus
# finaliza a interface
app.exec()
```

Algumas observações

- Inicialmente você deve carregar todos as classes que você está utilizando:

```
import sys
from PyQt5.QtGui import QPalette
from PyQt5.QtWidgets import QApplication, QMainWindow
from PyQt5.QtCore import Qt
```

- Toda aplicação em PyQt deve criar um objeto aplicação:

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

- Depois este deve ser finalizado:

```
    app.exec()
```

- Uma janela com o seu respectivo tamanho e label é criada da seguinte forma:

```
  window = QMainWindow()
  window.setFixedSize(640,480)
  window.setWindowTitle("Janela Azul")
```

Mais observações

- As cor de background da janela é feita dessa forma:

```
# Setando as cores
cores = QPalette(app.palette(window))
cores.setColor(QPalette.Window, Qt.blue)
window.setPalette(cores)
```

- Curiosamente a janela deve ser feita visível:

```
# Tornando a janela visivel
window.show()
```

- Uma ação pode ser definida quando do fechamento da janela. Nesse caso a função `adeus()` é executada.

```
# Setando o que ocorre no fechamento da janela
window.closeEvent = adeus
```

Teste com outras cores ?

- **Teste uma mudança da cor de *Background*.**
- As cores predefinidas são ilustradas a seguir:

Exemplo 2: Inserindo Botões na Janela Azul

- Criaremos agora um ToolBar e acrescentaremos botões nesse elemento.

- O código se encontra no arquivo pyqt2.py.
- Carregue o programa no IDE Spyder e execute.

Exemplo 2: arquivo pyqt2.py

- O código

```
from PyQt5.QtGui import QPalette
from PyQt5.QtWidgets import QApplication, QMainWindow, QToolBar, QWidget, QPu
from PyQt5.QtCore import Qt, QThread, QRunnable

def adeus(x):
    print("Adeus")

def trabalho():
    while True:
        QThread.msleep(500)
        print("Trabalhando...")

app = QApplication(sys.argv)
# define janela e cores
window = QMainWindow()
window.setFixedSize(640,480)
window.setWindowTitle("Janela Azul")
cores = QPalette(app.palette(window))
cores.setColor(QPalette.Window, Qt.blue)
```

Exemplo 2: arquivo pyqt2.py

- continuação ...

```
barra = window.addToolBar("")
window.setCentralWidget(QWidget())
window.centralWidget().setPalette(cores)
window.centralWidget().setAutoFillBackground(True)
# define uma thread nova
trabalhador = QThread()
trabalhador.run = trabalho
trabalhador.start()
# adiciona um botao
barra.addAction("Novo Botao")
barra.addAction("Outro Botao")
window.show()
window.closeEvent = adeus
app.exec()
```

Algumas observações

- Cria-se um ToolBar como abaixo:

```
barra = window.addToolBar("")
```

- Cria-se um objeto `QWidget` e associa-se ao objeto `window`:

```
window.setCentralWidget(QWidget())
```

- QWidget é a classe base para todo os objetos da interface de usuário.
- Eventos de mouse e teclados são gerenciados por QWidget.
- Cria-se uma nova thread denominada `trabalhador`.

```
trabalhador = QThread()
trabalhador.run = trabalho
trabalhador.start()
```

- Essa thread executa a função `trabalho()` que fica permanentemente imprimindo a mensagem "Trabalhando".

Mais observações

- Finalmente adiciona-se os botões da seguinte forma:

```
barra.addAction("Novo Botao")
barra.addAction("Outro Botao")
```

- **Insira vários botões adicionais e verifique o efeito na tela**.

Exemplo 3: Estabelecendo ações no botão

- Os botões agora realizam ações quando pressionados.

- O código se encontra no arquivo pyqt3.py.
- Carregue o programa no IDE Spyder e execute.

Exemplo 3: arquivo pyqt3.py

- O código é apresentado a seguir:

```
from PyQt5.QtGui import QPalette
from PyQt5.QtWidgets import QApplication, QMainWindow, QToolBar, QWidget, QPu
from PyQt5.QtCore import Qt, QThread, QRunnable

def adeus(x):
    print("Adeus")

def trabalho():
    while True:
        QThread.msleep(500)
        print("Trabalhando...")

def processa_acao(acao):
    if id(acao1)==id(acao):
        print("Acao 1")
    print("Botao pressionado:" + acao.text())

app = QApplication(sys.argv)
window = QMainWindow()
window.setFixedSize(640,480)
window.setWindowTitle("Janela Azul")
cores = QPalette(app.palette(window))
cores.setColor(QPalette.Window, Qt.blue)
window.setPalette(cores)
```

- Continuação

```
window.setCentralWidget(QWidget())
window.centralWidget().setPalette(cores)
window.centralWidget().setAutoFillBackground(True)
# Define uma thread nova que fica imprimindo
trabalhador = QThread()
trabalhador.run = trabalho
trabalhador.start()
# Adiciona a deteccao dos botoes
acao1 = barra.addAction("Novo Botao")
barra.addAction("Outro Botao")
barra.actionTriggered.connect(processa_acao)
window.show()

window.closeEvent = adeus
app.exec()
```

Algumas observações

- As ações relativas ao botão "Novo Botao" e "Outro Botao" são adicionados ao ToolBar `barra`

```
acao1 = barra.addAction("Novo Botao")
barra.addAction("Outro Botao")
```

- Quando o evento é detectado é acionado a função `processa_acao()`

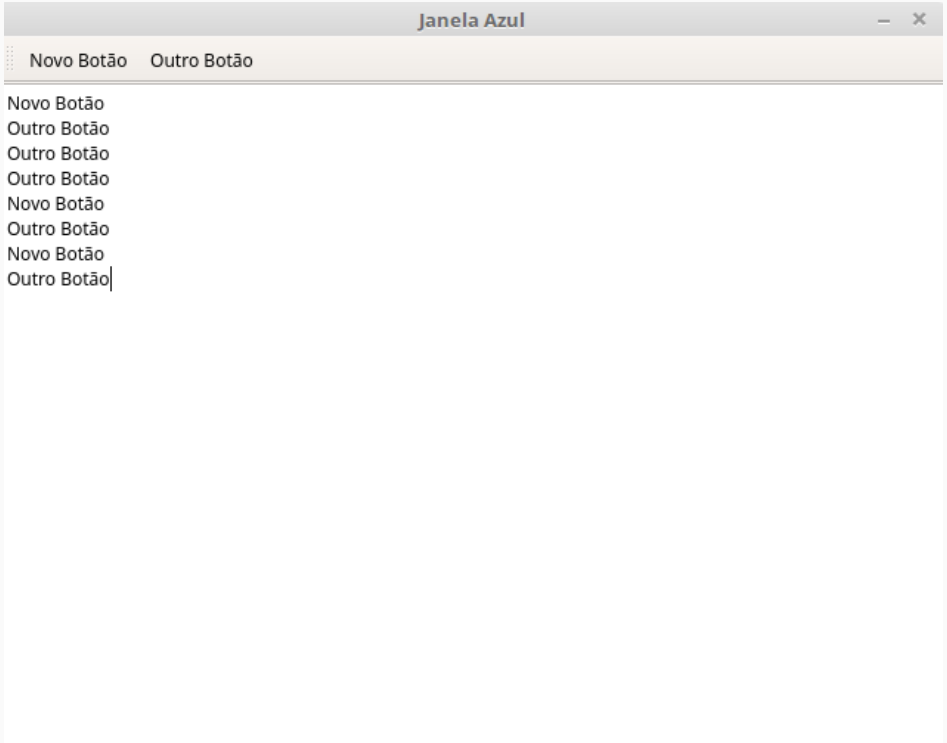
```
barra.actionTriggered.connect(processa_acao)
```

- A função `processa_acao()` trata de maneira distinta os botões:

```
def processa_acao(acao):
    if id(acao1)==id(acao):
        print("Acao 1")
    print("Botao pressionado:" + acao.text())
```

Exemplo 4: Colocando uma Caixa de Texto

- Agora quando são detectadas as ações sobre os botões existe uma mensagem correspondente na Caixa de Texto além da mensagem no console.



- O código se encontra no arquivo pyqt4.py.
- Carregue o programa no IDE Spyder e execute.

Exemplo 4: arquivo pyqt4.py

- O código é apresentado a seguir:

```
import sys
from PyQt5.QtGui import QPalette
from PyQt5.QtWidgets import QApplication, QMainWindow, QToolBar, QWidget, QPushButton, QLabel, QTextEdit
from PyQt5.QtCore import Qt, QThread, QRunnable


def adeus(x):
    print("Adeus")

def trabalho():
    while True:
        QThread.msleep(500)
        print("Trabalhando...")

def processa_acao(acao):
    if id(acao1)==id(acao):
        print("Acao 1")
    print("Botao pressionado:" + acao.text())

app = QApplication(sys.argv)
window = QMainWindow()
window.setFixedSize(640,480)
window.setWindowTitle("Janela Azul")
cores = QPalette(app.palette(window))
cores.setColor(QPalette.Window, Qt.blue)
```

- Continuação

```
barra = window.addToolBar("")
# cria o text box
window.setCentralWidget(QTextEdit())
window.centralWidget().setAutoFillBackground(True)

trabalhador = QThread()
trabalhador.run = trabalho
trabalhador.start()

acao1 = barra.addAction("Novo Botao")
barra.addAction("Outro Botao")
barra.actionTriggered.connect(processa_acao)
window.show()
# Insire a string relativa a acao atraves de append
barra.actionTriggered.connect(lambda acao : window.centralWidget().append(acao.text()))

window.closeEvent = adeus
app.exec()
```

Algumas observações

- Associa-se um objeto `QTextEdit` ao objeto `window`:

```
window.setCentralWidget(QTextEdit())
```

- Quando a ação é detectada insire-se a string definida por `acao.text()` através de uma ação `append`.

```
barra.actionTriggered.connect(lambda acao : window.centralWidget().append(acao.text()))
```