**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**3-Mavzu:** Pythonda tarmoq dasturlashga kirish.

**5-mashg‘ulot.** PyQt5 paketidan foydalanib zamonaviy chat dasturini tuzish.

**O‘quv savollari:**

1. TCP dasturini client qismini GUI ko’rinishda ishlab chiqish.

**1. TCP dasturini client qismini GUI ko‘rinishda ishlab chiqish.**

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

"""

Created on Tue Jul 24 13:05:46 2018

@author: JC

"""

import socket

import sys

import threading

import time

import functools

from PyQt5 import QtCore, QtGui

from PyQt5 import QtWidgets

from PyQt5.QtWidgets import QMainWindow, QApplication, QWidget, QPushButton

from PyQt5.QtWidgets import QVBoxLayout, QHBoxLayout, QMessageBox, QTabWidget

from PyQt5.QtWidgets import QGridLayout, QScrollArea, QLabel, QListView

from PyQt5.QtWidgets import QLineEdit, QComboBox, QGroupBox, QAction

from PyQt5.QtGui import QStandardItemModel, QStandardItem, QFont

class MyTableWidget(QWidget):

def \_\_init\_\_(self, parent):

super(QWidget, self).\_\_init\_\_(parent)

#connecion

self.conn = socket.socket()

self.connected = False

#tab UI

self.layout = QVBoxLayout(self)

self.tabs = QTabWidget()

self.tabs.resize(300,200)

self.tab1 = QWidget()

self.tab2 = QWidget()

self.tabs.addTab(self.tab1, "Home")

self.tabs.addTab(self.tab2, "Chat Room")

self.tabs.setTabEnabled(1,False)

#<Home>

gridHome = QGridLayout()

self.tab1.setLayout(gridHome)

self.IPBox = QGroupBox("IP")

self.IPLineEdit = QLineEdit()

self.IPLineEdit.setText("127.0.0.1")

IPBoxLayout = QVBoxLayout()

IPBoxLayout.addWidget(self.IPLineEdit)

self.IPBox.setLayout(IPBoxLayout)

self.portBox = QGroupBox("port")

self.portLineEdit = QLineEdit()

self.portLineEdit.setText("33002")

portBoxLayout = QVBoxLayout()

portBoxLayout.addWidget(self.portLineEdit)

self.portBox.setLayout(portBoxLayout)

self.nameBox = QGroupBox("Name")

self.nameLineEdit = QtWidgets.QLineEdit()

nameBoxLayout = QVBoxLayout()

nameBoxLayout.addWidget(self.nameLineEdit)

self.nameBox.setLayout(nameBoxLayout)

self.connStatus = QLabel("Status", self)

font = QFont()

font.setPointSize(16)

self.connStatus.setFont(font)

self.connBtn = QPushButton("Connect")

self.connBtn.clicked.connect(self.connect\_server)

self.disconnBtn = QPushButton("Disconnect")

self.disconnBtn.clicked.connect(self.disconnect\_server)

gridHome.addWidget(self.IPBox,0,0,1,1)

gridHome.addWidget(self.portBox,0,1,1,1)

gridHome.addWidget(self.nameBox,1,0,1,1)

gridHome.addWidget(self.connStatus,1,1,1,1)

gridHome.addWidget(self.connBtn,2,0,1,1)

gridHome.addWidget(self.disconnBtn,2,1,1,1)

gridHome.setColumnStretch(0, 1)

gridHome.setColumnStretch(1, 1)

gridHome.setRowStretch(0, 0)

gridHome.setRowStretch(1, 0)

gridHome.setRowStretch(2, 9)

#</Home>

#<Chat Room>

gridChatRoom = QGridLayout()

self.tab2.setLayout(gridChatRoom)

self.messageRecords = QLabel("<font color=\"#000000\">Welcome to chat room</font>", self)

self.messageRecords.setStyleSheet("background-color: white;");

self.messageRecords.setAlignment(QtCore.Qt.AlignTop)

self.messageRecords.setAutoFillBackground(True);

self.scrollRecords = QScrollArea()

self.scrollRecords.setWidget(self.messageRecords)

self.scrollRecords.setWidgetResizable(True)

self.sendTo = "ALL"

self.sendChoice = QLabel("Send to :ALL", self)

self.sendComboBox = QComboBox(self)

self.sendComboBox.addItem("ALL")

self.sendComboBox.activated[str].connect(self.send\_choice)

self.lineEdit = QLineEdit()

self.lineEnterBtn = QPushButton("Enter")

self.lineEnterBtn.clicked.connect(self.enter\_line)

self.lineEdit.returnPressed.connect(self.enter\_line)

self.friendList = QListView()

self.friendList.setWindowTitle('Room List')

self.model = QStandardItemModel(self.friendList)

self.friendList.setModel(self.model)

self.emojiBox = QGroupBox("Emoji")

self.emojiBtn1 = QPushButton("ก็ʕ•͡ᴥ•ʔ ก้")

self.emojiBtn1.clicked.connect(functools.partial(self.send\_emoji, "ก็ʕ•͡ᴥ•ʔ ก้"))

self.emojiBtn2 = QPushButton("(｡◕∀◕｡)")

self.emojiBtn2.clicked.connect(functools.partial(self.send\_emoji, "(｡◕∀◕｡)"))

self.emojiBtn3 = QPushButton("( ˘･з･)")

self.emojiBtn3.clicked.connect(functools.partial(self.send\_emoji, "( ˘･з･)"))

self.emojiBtn4 = QPushButton("ᕦ(ò\_óˇ)ᕤ")

self.emojiBtn4.clicked.connect(functools.partial(self.send\_emoji, "ᕦ(ò\_óˇ)ᕤ"))

emojiLayout = QHBoxLayout()

emojiLayout.addWidget(self.emojiBtn1)

emojiLayout.addWidget(self.emojiBtn2)

emojiLayout.addWidget(self.emojiBtn3)

emojiLayout.addWidget(self.emojiBtn4)

self.emojiBox.setLayout(emojiLayout)

gridChatRoom.addWidget(self.scrollRecords,0,0,1,3)

gridChatRoom.addWidget(self.friendList,0,3,1,1)

gridChatRoom.addWidget(self.sendComboBox,1,0,1,1)

gridChatRoom.addWidget(self.sendChoice,1,2,1,1)

gridChatRoom.addWidget(self.lineEdit,2,0,1,3)

gridChatRoom.addWidget(self.lineEnterBtn,2,3,1,1)

gridChatRoom.addWidget(self.emojiBox,3,0,1,4)

gridChatRoom.setColumnStretch(0, 9)

gridChatRoom.setColumnStretch(1, 9)

gridChatRoom.setColumnStretch(2, 9)

gridChatRoom.setColumnStretch(3, 1)

gridChatRoom.setRowStretch(0, 9)

#</Chat Room>

#Initialization

self.layout.addWidget(self.tabs)

self.setLayout(self.layout)

def enter\_line(self):

#assure the person still in rooom before send out

if self.sendTo != self.sendComboBox.currentText():

self.message\_display\_append("The person left. Private message not delivered")

self.lineEdit.clear()

return

line = self.lineEdit.text()

if line == "":#prevent empty message

return

if self.sendTo != "ALL":#private message, send to myself first

#this is a trick leverage the server sending back a copy to myself

send\_msg = bytes("{"+self.userName+"}"+line, "utf-8")

self.conn.send(send\_msg)

time.sleep(0.1) #this is important for not overlapping two sending

send\_msg = bytes("{"+self.sendTo+"}"+line, "utf-8")

self.conn.send(send\_msg)

self.lineEdit.clear()

self.scrollRecords.verticalScrollBar().setValue(self.scrollRecords.verticalScrollBar().maximum())

def send\_emoji(self, emoji):

#assure the person still in rooom before send out

if self.sendTo != self.sendComboBox.currentText():

self.message\_display\_append("The person left. Private message not delivered")

return

if self.sendTo != "ALL":#private message, send to myself first

#this is a trick leverage the server sending back a copy to myself

send\_msg = bytes("{"+self.userName+"}"+emoji, "utf-8")

self.conn.send(send\_msg)

time.sleep(0.1) #this is important for not overlapping two sending

send\_msg = bytes("{"+self.sendTo+"}"+emoji, "utf-8")

self.conn.send(send\_msg)

def message\_display\_append(self, newMessage, textColor = "#000000"):

oldText = self.messageRecords.text()

appendText = oldText+"<br /><font color=\""+textColor+"\">"+newMessage+"</font><font color=\"#000000\"></font>"

self.messageRecords.setText(appendText)

time.sleep(0.2) #this helps the bar set to bottom, after all message already appended

self.scrollRecords.verticalScrollBar().setValue(self.scrollRecords.verticalScrollBar().maximum())

def updateRoom(self):

while self.connected:

data = self.conn.recv(1024)

data = data.decode("utf-8")

print(data)

if data != "":

if "{CLIENTS}" in data:

welcome = data.split("{CLIENTS}")

self.update\_send\_to\_list(welcome[1])

self.update\_room\_list(welcome[1])

if not welcome[0][5:] == "":

self.message\_display\_append(welcome[0][5:])

self.scrollRecords.verticalScrollBar().setValue(self.scrollRecords.verticalScrollBar().maximum())

elif data[:5] == "{MSG}": #{MSG} includes broadcast and server msg

self.message\_display\_append(data[5:], "#006600")

self.scrollRecords.verticalScrollBar().setValue(self.scrollRecords.verticalScrollBar().maximum())

else: #private messgage is NONE format

self.message\_display\_append("{private}"+data, "#cc33cc")

self.scrollRecords.verticalScrollBar().setValue(self.scrollRecords.verticalScrollBar().maximum())

time.sleep(0.1) #this is for saving thread cycle time

def connect\_server(self):

if self.connected == True:

return

name = self.nameLineEdit.text()

if name == "":

self.connStatus.setText("Status :"+"Please enter your name")

return

self.userName = name

IP = self.IPLineEdit.text()

if IP == "":

IP = "127.0.0.1"

port = self.portLineEdit.text()

if port == "" or not port.isnumeric():

self.portLineEdit.setText("33002")

self.connStatus.setText("Status :"+"Port format invalid")

return

else:

port = int(port)

try:

self.conn.connect((IP, port))

except:

self.connStatus.setText("Status :"+" Refused")

self.conn = socket.socket()

return

send\_msg = bytes("{REGISTER}"+name, "utf-8")

self.conn.send(send\_msg)

self.connected = True

self.connStatus.setText("Status :"+" Connected")

self.nameLineEdit.setReadOnly(True) #This setting is not functional well

self.tabs.setTabEnabled(1,True)

self.rT = threading.Thread(target= self.updateRoom)

self.rT.start()

def disconnect\_server(self):

if self.connected == False:

return

send\_msg = bytes("{QUIT}", "utf-8")

self.conn.send(send\_msg)

self.connStatus.setText("Status :"+" Disconnected")

self.nameLineEdit.setReadOnly(False)

self.nameLineEdit.clear()

self.tabs.setTabEnabled(1,False)

self.connected = False

self.rT.join()

self.conn.close()

self.conn = socket.socket()

def update\_room\_list(self, strList):

L = strList.split("|")

self.model.clear()

for person in L:

item = QStandardItem(person)

item.setCheckable(False)

self.model.appendRow(item)

def update\_send\_to\_list(self, strList):

L = strList.split("|")

self.sendComboBox.clear()

self.sendComboBox.addItem("ALL")

for person in L:

if person != self.userName:

self.sendComboBox.addItem(person)

previous = self.sendTo

index = self.sendComboBox.findText(previous)

print("previous choice:",index)

if index != -1:

self.sendComboBox.setCurrentIndex(index) #updating, maintain receiver

else:

self.sendComboBox.setCurrentIndex(0) #updating, the receiver left, deafault to "ALL"

def send\_choice(self,text):

self.sendTo = text

print(self.sendTo)

self.sendChoice.setText("Send to: "+text)

class Window(QMainWindow):

def \_\_init\_\_(self):

super(Window, self).\_\_init\_\_()

self.setGeometry(50, 50, 500, 300)

self.setWindowTitle("Chat-Client")

self.table\_widget = MyTableWidget(self)

self.setCentralWidget(self.table\_widget)

self.show()

def closeEvent(self, event):

close = QMessageBox()

close.setText("You sure?")

close.setStandardButtons(QMessageBox.Yes | QMessageBox.Cancel)

close = close.exec()

if close == QMessageBox.Yes:

self.table\_widget.disconnect\_server() #disconnect to server before exit

event.accept()

else:

event.ignore()

def run():

app = QApplication(sys.argv)

GUI = Window()

sys.exit(app.exec\_())

if \_\_name\_\_ == "\_\_main\_\_":

run()