Documentacao_Consumer

AUTHOR
Versão
11/12/2018

Sumário

Table of contents

Namespaces

Lista de Namespaces

Esta é a lista de todos os Namespaces com suas respectivas descrições:

Ui 7

Índice Hierárquico

Hierarquia de Classes

Esta lista de hierarquias está parcialmente ordenada (ordem alfabética):

QMainWindow

MainWindow.....Erro: Origem da referência não encontrada

QWidget

Plotter.....Erro: Origem da referência não encontrada

Índice dos Componentes

Lista de Classes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:

MainWindow .....Erro: Origem da referência não encontrada
Plotter .....Erro: Origem da referência não encontrada

Índice dos Arquivos

Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:

C:/Users/robso/Desktop/qtsupervisory-master/QtTcpClientConsumer/main.cpp 13
C:/Users/robso/Desktop/qtsupervisory-master/QtTcpClientConsumer/mainwindow.cpp 14
C:/Users/robso/Desktop/qtsupervisory-master/QtTcpClientConsumer/mainwindow.h 15
C:/Users/robso/Desktop/qtsupervisory-master/QtTcpClientConsumer/plotter.cpp 16
C:/Users/robso/Desktop/qtsupervisory-master/QtTcpClientConsumer/plotter.h 17

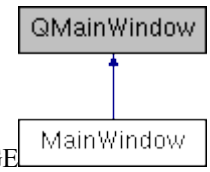
Namespace

Refência do Namespace Ui

Classes

Referência da Classe MainWindow

`#include <mainwindow.h>`
Diagrama de hierarquia para MainWindow:



IMAGE

Slots Públicos

void **tcpConnect** (void)
tcpConnect Conecta ao servidor

void **tcpDisconnect** (void)
tcpDisconnect Desconecta do servidor

void **getData** (void)
getData

void **stopData** (void)
stopData Para de colher dados do servidor

void **updateIP** (void)
updateIP Atualiza lista de clientes produtores conectados

void **timeEvent** (QTimerEvent *e)
timeEvent

Membros Públicos

MainWindow (QWidget *parent=0)
~MainWindow ()

Construtores e Destrutores

MainWindow::MainWindow (QWidget * *parent* = 0)[explicit]

```
13                                          :
14   QMainWindow(parent),
15   ui(new Ui::MainWindow)
16 {
17   ui->setupUi(this);
18   socket = new QTcpSocket(this);
19   idTime = 0;
20
21   connect(ui->pushButtonGet,
22           SIGNAL(clicked(bool)),
23           this,
24           SLOT(getData()));
25   connect(ui->pushButton_Update,
26           SIGNAL(clicked(bool)),
27           this,
28           SLOT(updateIP()));
29   connect(ui->pushButton_Stop,
30           SIGNAL(clicked(bool)),
31           this,
32           SLOT(tcpConnect()));
33   connect(ui->pushButton_Connect,
34           SIGNAL(clicked(bool)),
35           this,
36           SLOT(tcpConnect()));
```

```
37   connect(ui->pushButton_Disconnect,
38           SIGNAL(clicked(bool)),
39           this,
40           SLOT(tcpDisconnect()));
41
42 }
```

MainWindow::~MainWindow ()

```
188 {
189   delete socket;
190   delete ui;
191 }
```

Funções membros

void MainWindow::getData (void) `[slot]`

getData

```
58                          {
59   QString str;
60   QByteArray array;
61   QStringList list;
62   qint64 thetime;
63   qDebug() << "to get data...";
64   if(socket->state() == QAbstractSocket::ConnectedState){
65     if(socket->isOpen()){
66       qDebug() << "reading...";
67       socket->write("get 127.0.0.1 5\r\n");
68       socket->waitForBytesWritten();
69       socket->waitForReadyRead();
70       qDebug() << socket->bytesAvailable();
71       while(socket->bytesAvailable()){
72         str = socket->readLine().replace("\n","").replace("\r","");
73         list = str.split(" ");
74         if(list.size() == 2){
75           bool ok;
76           str = list.at(0);
77           thetime = str.toLongLong(&ok);
78           str = list.at(1);
79           qDebug() << thetime << ": " << str;
80         }
81       }
82     }
83   }
84 }
```

void MainWindow::stopData (void) `[slot]`

stopData Para de colher dados do servidor

```
181                              {
182     if(idTime){
183         killTimer(idTime);
184     }
185 }
```

void MainWindow::tcpConnect (void) `[slot]`

tcpConnect Conecta ao servidor

```
44                             {
45   socket->connectToHost("127.0.0.1",1234);
46   if(socket->waitForConnected(3000)){
47     qDebug() << "Connected";
48   }
49   else{
50     qDebug() << "Disconnected";
51   }
52 }
```

void MainWindow::tcpDisconnect (void) [slot]

tcpDisconnect Desconecta do servidor

```
   53                                {
   54     socket->disconnectFromHost();
   55 }
```

void MainWindow::timeEvent (QTimerEvent * *e*) [slot]

timeEvent

Parâmetros:

e	

```
   86                                          {
   87     QString str;
   88     QStringList list;
   89     qint64 thetime, num;
   90     double max_x, min_x, min_y, max_y;
   91     std::vector<double> time;
   92     std::vector<double> data;
   93     std::vector<double> temposnorm;
   94     std::vector<double> dadosnorm;
   95
   96     qDebug() << "to get data...";
   97     if(socket->state() == QAbstractSocket::ConnectedState){
   98         if(socket->isOpen()){
   99             qDebug() << "reading...";
  100             str = "get " + ui->listWidget_ListaDeClients->currentItem()-
>text() + " 30\r\n";
  101             socket->write(str.toStdString().c_str());
  102             socket->waitForBytesWritten();
  103             socket->waitForReadyRead();
  104             qDebug() << socket->bytesAvailable();
  105             time.clear();
  106             data.clear();
  107             while(socket->bytesAvailable()){
  108                 str = socket->readLine().replace("\n","").replace("\r","");
  109                 list = str.split(" ");
  110
  111                 if(list.size() == 2){
  112                     bool ok;
  113                     str = list.at(0);
  114                     thetime = str.toLongLong(&ok);
  115                     time.push_back(thetime);
  116
  117                     str = list.at(1);
  118                     num = str.toLongLong(&ok);
  119                     data.push_back(num);
  120                     qDebug() << thetime << ": " << str;
  121                 }
  122             }
  123         }
  124     }
  125
  126     qDebug()<<data.size()<<time.size();
  127     //achando valores maximos e minimos
  128     max_x = time[0], min_x = time[0];
  129     min_y = data[0], max_y = data[0];
  130
  131     for(int i = 1 ; i < 30; i++){
  132         if(time[i] < min_x){
  133             min_x = time[i];
  134         }
  135         else if(time[i] > max_x){
  136             max_x = time[i];
  137         }
  138         if(data[i] < min_y){
  139             min_y = data[i];
  140         }
  141         else if(data[i] > max_y){
  142             max_y = data[i];
```

```
143         }
144     }
145 
146     qDebug()<<max_x-min_x;
147 
148 
149     qDebug()<<max_y<<min_y;
150 
151 
152     //normalizando dados
153     temposnorm.clear();
154     dadosnorm.clear();
155     for(int i = 0; i<30; i++){
156         temposnorm.push_back((time[i] - min_x)/(max_x - min_x));
157         dadosnorm.push_back((data[i] - min_y)/(max_y - min_y));
158     }
159     qDebug()<<"passou";
160     ui->widget_Plotter->loadData(temposnorm,dadosnorm);
161 }
```

void MainWindow::updateIP (void) [slot]

updateIP Atualiza lista de clientes produtores conectados

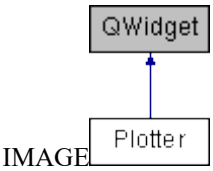
```
163                                 {
164     QString str;
165 
166     ui->listWidget_ListaDeClients->clear();
167     if(socket->state() == QAbstractSocket::ConnectedState){
168         socket->write("list\r\n");
169         socket->waitForBytesWritten();
170         socket->waitForReadyRead();
171         while(socket->bytesAvailable()){
172             str = socket->readLine().replace("\n","").replace("\r","");
173             ui->listWidget_ListaDeClients->addItem(str);
174         }
175     }
176     else{
177         ui->listWidget_ListaDeClients->addItem("Nao ha nada");
178     }
179 }
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

0 C:/Users/robso/Desktop/qtsupervisory-master/QtTcpClientConsumer/**mainwindow.h**
1 C:/Users/robso/Desktop/qtsupervisory-master/QtTcpClientConsumer/**mainwindow.cpp**

Referência da Classe Plotter

`#include <plotter.h>`
Diagrama de hierarquia para Plotter:



IMAGE

Membros Públicos

Plotter (QWidget *parent=nullptr)
Plotter seta valores.
void **paintEvent** (QPaintEvent *e)
paintEvent
void **loadData** (vector< double >, vector< double >)
loadData

Construtores e Destrutores

Plotter::Plotter (QWidget * *parent* = `nullptr`)`[explicit]`

Plotter seta valores.

Parâmetros:

parent	

```
 9                                   : QWidget(parent)
10 {
11     for(int i=0; i<30; i++){
12         tempos.push_back(i);
13         dados.push_back(i);
14     }
15 }
```

Funções membros

void Plotter::loadData (vector< double > *t*, vector< double > *d*)

loadData

```
53                                                          {
54     tempos = t;
55     dados = d;
56     repaint();
57 }
```

void Plotter::paintEvent (QPaintEvent * *e*)

paintEvent

Parâmetros:

e	

```
18 {
19     QPainter painter(this);
```

```
20      QBrush brush;
21      QPen pen;
22      double x1, x2, y1, y2;
23
24      painter.setRenderHint(QPainter::Antialiasing);
25
26      brush.setColor(QColor(255, 100, 100));
27      brush.setStyle(Qt::SolidPattern);
28
29      painter.setBrush(brush);
30      painter.setPen(pen);
31
32      //Desenha o fundo do Plotter
33      painter.drawRect(0, 0, width(), height());
34
35      pen.setBrush(QColor(0, 0, 255));
36      pen.setWidth(2);
37      pen.setStyle(Qt::SolidLine);
38      painter.setPen(pen);
39
40      //Plotando os graficos
41      x1 = tempos[0]*width();
42      y1 = dados[0]*(height() - dados[0]);
43
44      for(int i=1; i<30; i++){
45          x2 = tempos[i]*width();
46          y2 = dados[i]*(height() - dados[i]);
47          painter.drawLine(x1, y1, x2, y2);
48          x1 = x2;
49          y1 = y2;
50      }
51  }
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

2 C:/Users/robso/Desktop/qtsupervisory-master/QtTcpClientConsumer/**plotter.h**
3 C:/Users/robso/Desktop/qtsupervisory-master/QtTcpClientConsumer/**plotter.cpp**

Arquivos

Referência do Arquivo C:/Users/robso/Desktop/qtsupervisory-master/QtTcpClientConsumer/main.cpp

```
#include "mainwindow.h"
#include <QApplication>
```

Funções

int **main** (int argc, char *argv[])

Funções

int main (int *argc*, char * *argv*[])

```
 5 {
 6   QApplication a(argc, argv);
 7   MainWindow w;
 8   w.show();
 9
10   return a.exec();
11 }
```

Referência do Arquivo C:/Users/robso/Desktop/qtsupervisory-master/QtTcpClientConsumer/mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QDateTime>
#include <plotter.h>
#include <QLabel>
#include <QLineEdit>
#include <vector>
#include <QListWidget>
#include <QSlider>
#include <QWidget>
```

Referência do Arquivo C:/Users/robso/Desktop/qtsupervisory-master/QtTcpClientConsumer/mainwindow.h

```
#include <QMainWindow>
#include <QTcpSocket>
#include <QDebug>
```

Componentes

class **MainWindow**

Namespaces

Ui

Referência do Arquivo C:/Users/robso/Desktop/qtsupervisory-master/QtTcpClientConsumer/plotter.cpp

```
#include "plotter.h"
#include <QPainter>
#include <QBrush>
#include <QColor>
#include <cmath>
#include <QDebug>
```

Referência do Arquivo C:/Users/robso/Desktop/qtsupervisory-master/QtTcpClientConsumer/plotter.h

```
#include <QWidget>
#include <vector>
```

Componentes

class **Plotter**

Sumário

INDE