在QT中向API网关发送http请求并显示相关代码：

**#include "httpthread.h"#include <QDebug>#include <QBuffer>#include <QFileDialog>#include <QMessageBox>#include <QDebug>**

HttpThread::HttpThread()

{

manager = NULL;

SetApiType = FACE\_RECOGNITION;

requestUrls.insert(FACE\_RECOGNITION,"http://192.168.50.100:2280/face-recognition");

requestUrls.insert(IMAGE\_RECOGNITION,"http://192.168.50.100:2280/image-recognition");

requestUrls.insert(FACE\_IMAGE\_RECOGNITION,"http://192.168.50.100:2280/face-image-recognition");

**this**->init\_post\_datas();

exitFreeImage = false;

}

HttpThread::~HttpThread()

{

}

QByteArray HttpThread::QVariantMapToQByteArray(QVariantMap post\_data)

{

QByteArray json;

json.append('{');

**int** i =0;

foreach( QVariant tmp, post\_data )

{

if (i != 0)

{

json.append(",");

}

json.append("\""+post\_data.keys().at(i)+"\""+":");

qDebug()<<"typename:"<<tmp.typeName();

if (strcmp(tmp.typeName(), "QString") == 0)

json.append("\""+tmp.toString()+"\"");

else

json.append(tmp.toString());

i++;

}

json.append('}');

return json;

}

QString HttpThread::QImageToBase64(**const** QImage& img){

QByteArray ba;

QBuffer buffer(&ba);

buffer.open(QIODevice::WriteOnly);

img.save(&buffer, "jpg");

QString imgData = QString(ba.toBase64());

buffer.close();

return imgData;

}

**void** HttpThread::init\_post\_datas(){

QVariantMap post\_data;

*//face-recognition*

post\_data.insert("match\_threshold", 0.6);

post\_datas.insert(FACE\_RECOGNITION,post\_data);

*//image-recognition*

post\_data.clear();

post\_data.insert("threshold", 0.8);

post\_datas.insert(IMAGE\_RECOGNITION,post\_data);

*//face-image-recognition*

post\_data.clear();

post\_data.insert("match\_threshold", 0.6);

post\_data.insert("threshold", 0.8);

post\_datas.insert(FACE\_IMAGE\_RECOGNITION,post\_data);

}

**void** HttpThread::sendHttpRequest()

{

if (manager != NULL)

{

disconnect(manager, SIGNAL(finished(QNetworkReply\*)), **this**, SLOT(replyFinished(QNetworkReply\*)));

**delete** manager;

manager = NULL;

qDebug() << "manager is not NULL" ;

}

manager = **new** QNetworkAccessManager(**this**);

connect(manager, SIGNAL(finished(QNetworkReply\*)), **this**, SLOT(replyFinished(QNetworkReply\*)));

*//设置url*

QUrl url(requestUrls.value(**this**->SetApiType));

QNetworkRequest request(url);

*//设置Header*

request.setHeader(QNetworkRequest::ContentTypeHeader, QVariant("application/json"));

*//set post data*

QVariantMap post\_data = post\_datas.value(**this**->SetApiType);

QString imgData = **this**->QImageToBase64(videoimage);

post\_data.insert("image", imgData);

QByteArray json = **this**->QVariantMapToQByteArray(post\_data);

*//发送请求*

qDebug() << "sendHttpRequest post Begin";

manager->post(request, json);

qDebug() << "sendHttpRequest post End";

}

**void** HttpThread::run()

{

while (1)

{

msleep(1);

}

qDebug() << "the http thread quit" ;

}

**void** HttpThread::replyFinished(QNetworkReply \*reply)

{

**int** statusCode = reply->attribute(QNetworkRequest::HttpStatusCodeAttribute).toInt();

if(200 != statusCode)

{

qDebug() << "Status code is error:" << statusCode;

reply->deleteLater();

exitFreeImage = false;

return;

}

replyData = reply->readAll();

qDebug()<<"reply data is:"<<QString(replyData);

emit api\_result(videoimage, QString(replyData), **this**->SetApiType);

reply->deleteLater();

exitFreeImage = false;

}

**void** HttpThread::SetImage(**const** QImage &image)

{

if (exitFreeImage == false)

{

if(image.height()>0)

{

videoimage = image;

exitFreeImage = true;

sendHttpRequest();

}

else

{

}

}

else

{

}

}

**void** HttpThread::setSetApiType(**const** QString &value)

{

SetApiType = value;

}

*#include "mainwindow.h"#include "ui\_mainwindow.h"#include <QPainter>#include <QPaintEvent>#include <QMessageBox>#include <QDebug>#include "cJSON.h"*

MainWindow::MainWindow(QWidget \***parent**) :

QMainWindow(**parent**),

ui(**new** Ui::MainWindow)

{

ui->setupUi(this);

ui->mainToolBar->hide();

ui->recordBut->setText(QString::fromLocal8Bit("录制"));

recordFlag = **true**;*//true表示可录制，false表示可停止*

ffmpeg = **NULL**;

rtspthread = **NULL**;

httpthread = **new** HttpThread();

connect(httpthread, SIGNAL(api\_result(**const** QImage &, QString, QString)),

this, SLOT(show\_result(**const** QImage &, QString, QString)));

httpthread->start();

}

MainWindow::~MainWindow()

{

**if**(ffmpeg)

{

disconnect(ffmpeg,SIGNAL(GetImage(QImage)),this,SLOT(SetImage(QImage)));

disconnect(ffmpeg,SIGNAL(GetImage(QImage)),httpthread,SLOT(SetImage(QImage)));

disconnect(httpthread, SIGNAL(api\_result(**const** QImage &, QString, QString)),

this, SLOT(show\_result(**const** QImage &, QString, QString)));

ffmpeg->flag = **false**;

*// ffmpeg->deleteLater();*

*//rtspthread->deleteLater();*

}

delete ui;

}

void MainWindow::paintEvent(QPaintEvent \*event)

{

**if**(!videoimage.isNull())

{

QPainter painter(this);

*//QPixmap pix = QPixmap::fromImage(videoimage.scaled(ui->widget->width(),ui->widget->height(),Qt::KeepAspectRatio));*

ui->widget->resize(videoimage.size());

*/\**

*double widget\_W,widget\_h;*

*widget\_h = ui->widget->height();*

*widget\_W = ui->widget->width();*

*QImage temp = videoimage.scaled(ui->widget->width(),ui->widget->height(),Qt::KeepAspectRatio);*

*\*/*

painter.translate(0,0);

painter.drawImage(0,0,videoimage);

*//painter.drawImage(ui->widget->rect(),videoimage.scaled(ui->widget->width(),ui->widget->height(),Qt::KeepAspectRatio));*

*//painter.drawPixmap(ui->widget->rect(),pix);*

*//画矩形框*

QPainter paint;

paint.begin(this);

paint.translate(0,0);

paint.setPen(QPen(Qt::red, 1, Qt::SolidLine, Qt::RoundCap));

**int** i=0;

**foreach**(QRect rect,user\_rects)

{

paint.drawRect(rect);

paint.drawText(rect.topLeft(),user\_names.at(i));

i++;

}

i = 0;

**foreach**(QRect rect,object\_rects)

{

paint.drawRect(rect);

paint.drawText(rect.topLeft(),object\_names.at(i));

i++;

}

paint.end();

}

}

void MainWindow::on\_playBtn\_clicked()

{

*//ffmpegDecode \*ffmpeg = new ffmpegDecode(this);*

ffmpeg = **new** ffmpegDecode(this);

connect(ffmpeg,SIGNAL(GetImage(QImage)),this,SLOT(SetImage(QImage)));

connect(ffmpeg,SIGNAL(GetImage(QImage)),httpthread,SLOT(SetImage(QImage)));

ffmpeg->SetUrl(ui->lineEdit->text());

qDebug() << "url : " << ui->lineEdit->text();

**if**(ffmpeg->init())

{

*//RtspThread \*rtspthread = new RtspThread();*

rtspthread = **new** RtspThread();

rtspthread->setFFmpeg(ffmpeg);

rtspthread->start();

}**else**

{

QMessageBox::warning(this,QString::fromLocal8Bit("警告"),QString::fromLocal8Bit("初始化解码器错误！"),QMessageBox::Ok);

}

}

void MainWindow::SetImage(**const** QImage &image)

{

**if**(image.height()>0)

{

*//ui->video\_label->resize(image.width(),image.height());*

*// QPixmap pix = QPixmap::fromImage(image.scaled(ui->video\_label->width(),ui->video\_label->height(),Qt::KeepAspectRatio));*

videoimage = image;

update();

*//QPixmap pix = QPixmap::fromImage(image);// ui->video\_label->setAlignment(Qt::AlignCenter);// ui->video\_label->setPixmap(pix);*

}

}

void MainWindow::on\_recordBut\_clicked()

{

**if**(recordFlag)

{

**bool** flag = ffmpeg->initRecord();

ffmpeg->setRecordState(**true**);

recordFlag = **false**;

ui->recordBut->setText(QString::fromLocal8Bit("停止"));

}**else**

{

ffmpeg->setRecordState(**false**);

ffmpeg->wFileTrailer();

ffmpeg->delRecord();

recordFlag = **true**;

ui->recordBut->setText(QString::fromLocal8Bit("录制"));

}

}

void MainWindow::show\_result(**const** QImage &image, QString result, QString ApiType)

{

user\_rects.clear();

user\_names.clear();

object\_rects.clear();

object\_names.clear();

cJSON \*root = cJSON\_Parse(result.toStdString().c\_str());

**if** (root == **NULL**)

{

printf("json error!");

cJSON\_Delete(root);

**return**;

}

**try** {

**if**(ApiType == FACE\_RECOGNITION)

{

**const** cJSON \*face\_list = cJSON\_GetObjectItemCaseSensitive(root, "face\_list");

**const** cJSON \*face = **NULL**;

cJSON\_ArrayForEach(face, face\_list)

{

**const** cJSON \*face\_location = cJSON\_GetObjectItemCaseSensitive(face, "location");

**const** cJSON \*face\_dot = **NULL**;

**const** cJSON \*face\_userlist = cJSON\_GetObjectItemCaseSensitive(face, "user\_list");

**const** cJSON \*user\_info = **NULL**;

**int** temp\_rect[4];

**int** i=0;

cJSON\_ArrayForEach(face\_dot, face\_location)

{

temp\_rect[i] = face\_dot->valueint;

i++;

**if**(i>3) **break**;

}

QRect face\_rect(temp\_rect[0],temp\_rect[1],temp\_rect[2]-temp\_rect[0],temp\_rect[3]-temp\_rect[1]);

QString user\_name = "unknow";

cJSON\_ArrayForEach(user\_info, face\_userlist)

{

user\_name = cJSON\_GetObjectItemCaseSensitive(user\_info, "user\_info")->valuestring;

**break**;

}

user\_rects.append(face\_rect);

user\_names.append(user\_name);

}

}

**else** **if**(ApiType == IMAGE\_RECOGNITION)

{

**const** cJSON \*object\_list = cJSON\_GetObjectItemCaseSensitive(root, "result");

**const** cJSON \*object = **NULL**;

cJSON\_ArrayForEach(object, object\_list)

{

**const** cJSON \*object\_location = cJSON\_GetObjectItemCaseSensitive(object, "location");

**const** cJSON \*object\_dot = **NULL**;

**int** temp\_rect[4];

**int** i=0;

cJSON\_ArrayForEach(object\_dot, object\_location)

{

temp\_rect[i] = object\_dot->valueint;

i++;

**if**(i>3) **break**;

}

QRect object\_rect(temp\_rect[0],temp\_rect[1],temp\_rect[2]-temp\_rect[0],temp\_rect[3]-temp\_rect[1]);

QString object\_name = cJSON\_GetObjectItemCaseSensitive(object, "keyword")->valuestring;

object\_rects.append(object\_rect);

object\_names.append(object\_name);

}

}

**else** **if**(ApiType == FACE\_IMAGE\_RECOGNITION)

{

**const** cJSON \*all\_face = cJSON\_GetObjectItemCaseSensitive(root, "face");

**const** cJSON \*all\_object = cJSON\_GetObjectItemCaseSensitive(root, "object");

**const** cJSON \*face\_list = cJSON\_GetObjectItemCaseSensitive(all\_face, "face\_list");

**const** cJSON \*object\_list = cJSON\_GetObjectItemCaseSensitive(all\_object, "result");

**const** cJSON \*face = **NULL**;

**const** cJSON \*object = **NULL**;

cJSON\_ArrayForEach(face, face\_list)

{

**const** cJSON \*face\_location = cJSON\_GetObjectItemCaseSensitive(face, "location");

**const** cJSON \*face\_dot = **NULL**;

**const** cJSON \*face\_userlist = cJSON\_GetObjectItemCaseSensitive(face, "user\_list");

**const** cJSON \*user\_info = **NULL**;

**int** temp\_rect[4];

**int** i=0;

cJSON\_ArrayForEach(face\_dot, face\_location)

{

temp\_rect[i] = face\_dot->valueint;

i++;

**if**(i>3) **break**;

}

QRect face\_rect(temp\_rect[0],temp\_rect[1],temp\_rect[2]-temp\_rect[0],temp\_rect[3]-temp\_rect[1]);

QString user\_name = "unknow";

cJSON\_ArrayForEach(user\_info, face\_userlist)

{

user\_name = cJSON\_GetObjectItemCaseSensitive(user\_info, "user\_info")->valuestring;

**break**;

}

user\_rects.append(face\_rect);

user\_names.append(user\_name);

}

cJSON\_ArrayForEach(object, object\_list)

{

**const** cJSON \*object\_location = cJSON\_GetObjectItemCaseSensitive(object, "location");

**const** cJSON \*object\_dot = **NULL**;

**int** temp\_rect[4];

**int** i=0;

cJSON\_ArrayForEach(object\_dot, object\_location)

{

temp\_rect[i] = object\_dot->valueint;

i++;

**if**(i>3) **break**;

}

QRect object\_rect(temp\_rect[0],temp\_rect[1],temp\_rect[2]-temp\_rect[0],temp\_rect[3]-temp\_rect[1]);

QString object\_name = cJSON\_GetObjectItemCaseSensitive(object, "keyword")->valuestring;

object\_rects.append(object\_rect);

object\_names.append(object\_name);

}

}

} **catch** (QString **exception**){

user\_rects.clear();

user\_names.clear();

object\_rects.clear();

object\_names.clear();

printf("json error!");

}

**if** (root) cJSON\_Delete(root);

}

void MainWindow::on\_comboBox\_currentIndexChanged(**int** index)

{

qDebug()<<"iiiiiiiiiiiiii:"<<index;

**switch** (index) {

**case** 0:

this->httpthread->setSetApiType(FACE\_RECOGNITION);

**break**;

**case** 1:

this->httpthread->setSetApiType(IMAGE\_RECOGNITION);

**break**;

**case** 2:

this->httpthread->setSetApiType(FACE\_IMAGE\_RECOGNITION);

**break**;

**default**:

this->httpthread->setSetApiType(FACE\_RECOGNITION);

**break**;

}

}