**Connection.h 文件**

#ifndef CONNECTION\_H

#define CONNECTION\_H

#include <QMessageBox>

#include <QSqlDatabase>

#include <QSqlQuery>

#include <QTime>

#include <QDateTime>

#include <QDebug>

static bool createConnection()

{ qDebug()<<"aa";

QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");

qDebug()<<"bb";

db.setDatabaseName("ss.db");

qDebug()<<"cc";

if (!db.open()) {

QMessageBox::critical(0, QObject::tr("无法打开数据库"),

"无法创建数据库连接！", QMessageBox::Cancel);

qDebug()<<"dd";

return false;

}

QSqlQuery query;

// 创建分类表

query.exec("create table type(data varchar(500),time varchar(500),wendu varchar(500), shidu varchar(500))");

return true;

}

#endif // CONNECTION\_H

**Clientwidget.h文件**

#include "clientwidget.h"

#include "ui\_clientwidget.h"

#include "qdebug.h"

clientWidget::clientWidget(QWidget \*parent) :

QWidget(parent),

ui(new Ui::clientWidget)

{

ui->setupUi(this);

socket = new QTcpSocket(this);

connect(ui->pushButton\_connect,SIGNAL(clicked()),this,SLOT(onConnect())); //点击相关按钮触发相关槽函数

connect(ui->pushButton\_disconnect,SIGNAL(clicked()),this,SLOT(onDisconnect()));

connect(ui->pushButton\_send,SIGNAL(clicked()),this,SLOT(onSend()));

connect(socket,SIGNAL(connected()),this,SLOT(imcoming())); //管道建立后触发相关函数

ui->pushButton\_send->setShortcut(tr("Ctrl+R")); //设置发送相关快捷键

}

clientWidget::~clientWidget()

{

delete socket;

delete ui;

sum=0;

}

/\*连接按钮点击后的函数\*/

void clientWidget::onConnect()

{

ui->pushButton\_connect->setDisabled(true); //设置相关按键使能与否

socket->connectToHost(QString(ui->lineEdit\_socket->text()),9090); //这里可以在lineEdit里写相应ip，如果只是测试只要输入localhost即可，且配置好端口

if(!socket->waitForConnected(3000)) //等待3s，如果管道没接通那么则行{}内容

{

ui->pushButton\_connect->setEnabled(true);

ui->textEdit->append("connect failed!");

return;

}

ui->pushButton\_disconnect->setEnabled(true); //设置相关按键使能与否

ui->pushButton\_send->setEnabled(true);

ui->textEdit->append("connected to host");

// connect(socket,SIGNAL(readyRead()),this,SLOT(onread()));

}

/\*点击Disconnect按钮后执行函数\*/

void clientWidget::onDisconnect()

{

socket->abort(); //强行断开管道

ui->textEdit->append("disconnected!"); //设置相关按键使能与否

ui->pushButton\_connect->setEnabled(true);

ui->pushButton\_disconnect->setDisabled(true);

ui->pushButton\_send->setDisabled(true);

// connect(socket,SIGNAL(readyRead()),this,SLOT(onread()));

}

/\*点击Send按钮后触发函数\*/

void clientWidget::onSend()

{

socket->write(ui->lineEdit->text().toUtf8()); //写信息进管道

// socket->write("Hello",strlen("Hello")); //写信息进管道

socket->flush(); //推信息

ui->lineEdit->clear();

}

/\*读信息函数\*/

void clientWidget::onread()

{

QSqlQuery query;

char bufclient[100]; //缓冲区

char w[5];

char s[5];

QString da;

QString tt;

memset(bufclient,0,sizeof(bufclient)); //清空缓冲区

int lengthc = socket->bytesAvailable(); //从管道中计算传送数据长度

if(lengthc>0)

{

socket->read(bufclient,lengthc); //把指定长度数据从管道中放入缓冲区buf

ui->textEdit->insertPlainText("\n");

ui->textEdit->insertPlainText("server : "); //显示相关发信息人，以及时间等

ui->textEdit->insertPlainText(QDateTime::currentDateTime().toString("yyyy-MM-dd "));

ui->textEdit->insertPlainText(QTime::currentTime().toString("hh:mm:ss"));//textEdit文本框显示当前时间

ui->textEdit->insertPlainText("\n");

ui->textEdit->append(bufclient); //显示信息

ui->textEdit->insertPlainText("\n");

da=QDateTime::currentDateTime().toString("yyyy-MM-dd");

tt=QTime::currentTime().toString("hh:mm:ss");

memset(w,0,sizeof(w));

memset(s,0,sizeof(s));

w[0]=bufclient[0];

w[1]=bufclient[1];

s[0]=bufclient[3];

s[1]=bufclient[4];

/\*bool rtn = query.exec(QString("insert into table values('%1', '%2', '%3', %4, %5, 0, %6)")

.arg(id).arg(brand).arg(type).arg(price).arg(num).arg(num));\*/

query.exec(QString("insert into type values('%1','%2','%3','%4')").arg(da).arg(tt).arg(w).arg(s));

}

//ui->textEdit->append(bufclient);

}

/\*管道建立后，触发的相关函数\*/

void clientWidget::imcoming()

{

ui->textEdit->append("connected.");

connect(socket,SIGNAL(readyRead()),this,SLOT(onread())); //准备，然后触发读信息槽

connect(socket,SIGNAL(disconnected()),this,SLOT(enServerButton()));//管道断开，触发相关槽

}

/\*断开后，各按键相关使能函数\*/

void clientWidget::enServerButton()

{

ui->pushButton\_connect->setEnabled(true); //各按键使能

ui->pushButton\_disconnect->setDisabled(true);

ui->pushButton\_send->setDisabled(true);

// socket->abort();

}

void clientWidget::on\_opensqlbutton\_clicked()

{

QSqlQuery query;

// int r=0;

query.exec("select \* from type");

while(query.next())

{sum++;}

/\* query.exec("select \* from type");

QStandardItemModel \*model=new QStandardItemModel(sum,4,this);

int k=0;

qDebug()<<query.value(k).toString();

while(query.next())

{

QStandardItem \*item=new QStandardItem(query.value(0).toString());

model->setItem(r,0,item);

QStandardItem \*item1=new QStandardItem(query.value(1).toString());

model->setItem(r,1,item1);

QStandardItem \*item2=new QStandardItem(query.value(2).toString());

model->setItem(r,2,item2);

QStandardItem \*item3=new QStandardItem(query.value(3).toString());

model->setItem(r,3,item3);

r++;

\*/

/\* qDebug()<<query.value(0).toString()<<query.value(1).toString()<<query.value(2).toString()<<query.value(3).toString();

ui->textEdit\_2->append(query.value(0).toString());

ui->textEdit\_2->append(query.value(1).toString());

ui->textEdit\_2->append("wendu:");

ui->textEdit\_2->append(query.value(2).toString());

ui->textEdit\_2->append("shidu:");

ui->textEdit\_2->append(query.value(3).toString());

ui->textEdit\_2->append("\n");

\*/

/\*

}

this->hide();

tableView=new QTableView;

tableView->setModel(model);

tableView->show();

this->show();

sum=0;

\*/

Dialog2 dialog;

dialog.exec();

}

void clientWidget::on\_cleanbutton\_clicked()

{

QSqlQuery query;

query.exec("delete from type");

sum=0;

}

void clientWidget::on\_pushBotton\_clear\_clicked()

{

ui->lineEdit\_socket->clear();

}