目录

[AcountManage 2](#_Toc38215599)

[DataBase 2](#_Toc38215600)

[Ihandler 2](#_Toc38215601)

[MainWindow 2](#_Toc38215602)

[NewUserVector 2](#_Toc38215603)

[Protocol 2](#_Toc38215604)

[Room 2](#_Toc38215605)

[RoomManage 2](#_Toc38215606)

[RoomVector 2](#_Toc38215607)

[SocketHandle 2](#_Toc38215608)

[StrategyHandle 2](#_Toc38215609)

[User 2](#_Toc38215610)

[UserHandle 2](#_Toc38215611)

# AcountManage

# DataBase

# Ihandler

virtual void *Handle*(QTcpSocket \*socket ,const Protocol &p) = 0;//纯虚函数

被AcountManage和RoomManage两个类继承

# MainWindow

负责连接Tcp

将所有的登录账户压入一个容器里面QVector<SocketHandle\*> sockets;（//保存所有客户端套接字信息的动态数组）

# NewUserVector

这是个单例，只有Tcp程序不退出运行，就源源不断地将所有账户压入该类的私有变量QVector<SocketHandle\*> sockets;中

这个类存在的意义：当主播创建直播间或者主播退出直播间时候，所有登录进来的账户，在客厅左边都要刷新已存在的主播房间名

# Protocol

# Room

在创建主播房间时候，总是：

Room host\_room(hostname,name,socket,ip,camera\_port,audio\_port);

New 一个主播房间实例

然后压入QVector<Room>动态数组里面

# RoomManage

# RoomVector

将QVector<Room> chat\_room;定义成私有成员

在主播创建房间时，通过

bool RoomVector::CheckRoom(const Room &room) {

QVector<Room>::iterator it;

for(it = chat\_room.begin(); it!=chat\_room.end(); it++) {

//确保一个客户端只能创建一个直播间,并且不能和已存在的房间重名

if(room.GetHostRoomName() == it->GetHostRoomName() || room.GetSocket() == it->GetSocket()) {//判断条件就两条

return true;

}

}

return false;//此处与其他不同，当为假的时候才能创建房间

}

确保一个主播只能创建一个直播间，并且还不能与已存在的直播间重名

在关闭直播间时候，判断是否是主播关闭的：

当判断是主播退出房间之后

要让在这个直播间所有的观众返回到大厅

当主播退出该直播间后，将该直播间从Qvec<Room>容器里面删除

所有已登录账户的大厅刷新现存在的直播间

//====关闭直播账户====//

void RoomManage::CloseHostRoom(QTcpSocket \*socket, const Protocol &p) {

QString host\_name = p["host\_name"].toString();//房间名

QString user\_name = p["user\_name"].toString();//账户名

Protocol pRet(Protocol::CloseRoom);//------------------------------

Protocol pUse(Protocol::UserLset);

qDebug() <<"服务器接收到房间名/账户名:" <<host\_name<<user\_name;

RoomVector \*rv = RoomVector::GetInstance();

QVector<Room>& chatrooms = rv->GetAllChat();

for(QVector<Room>::iterator it = chatrooms.begin(); it!=chatrooms.end(); it++) {

if(it->GetHostRoomName() == host\_name) {//找到主播房间名

QVector<Account\_t>& per = it->GetAudience();//找到这个主播房间下对应的所有观众（放在容器里面）

//判断是否为主播关闭窗口：QVector<Room>存了所有关于主播房间的类，找到这个含有房间名/账户名的类，与传过来的user\_name 对比

if(it->GetHostPreson() == user\_name) {

for(QVector<Account\_t>::iterator ip = per.begin(); ip!= per.end(); ip++) {

//下面两行的意思是(要关闭的主播房间)里面所的观众窗口全部退出观看窗口，返回到大厅

pRet["result"] = "CloseHostRoomTrue";

ip->socket->write(pRet.pack());//遍历，所有游客关闭窗口

this->AccountRefresh(ip->socket,ip->name);

}

//防止直播间只有主播自己的情况，无法进入上述for循环，则直接读到如下语句

pRet["result"] = "CloseHostRoomTrue";

it->GetSocket()->write(pRet.pack());

this->AccountRefresh(it->GetSocket(),it->GetHostPreson());

//当主播退出该直播房间后，将该直播房间从QVector<Room>容器中删除

rv->EraseRoom(\*it);

//主播房间关闭成功，直播间大厅自动刷新已存在的主播间

NewUserVector\* instance = NewUserVector::GetInstance();

QVector<SocketHandle\*>& sh = instance->GetAllSockets();

for(QVector<SocketHandle\*>::iterator it = sh.begin(); it != sh.end(); it++) {

QTcpSocket\* socketTmp = (\*it)->GetSocket();

this->RoomListHandle(socketTmp,p);

}

break;

} else {

}

}

}

return ;

}

# SocketHandle

# StrategyHandle

# User

# UserHandle