# 安卓项目文档

# 一．《匿聊》功能架构

## （1）登入注册

连接云服务器，将用户信息存储到数据库上。

### 类DBUtil

实现数据库连接及相关操作

在阿里云服务器上部署mysql，通过JDBC连接，需要下载一个包存放在lib文件夹中

**public class** DBUtil {  
 *//接口服务器* **private static final** String ***databaseName*** = **"android\_test"**; *// 数据库名称* **private static final** String ***ip*** = **"rm-cn-pe339dysm0016r8o.rwlb.rds.aliyuncs.com"**; *// 主机名* **private static final** String ***port*** = **"3306"**; *// 端口号 例如 mysql 的默认端口号为 3306* **private static final** String ***user*** = **"testuser"**;*// 用户名* **private static final** String ***password*** = **"Qwe13681045826"**;*// 密码* **private static** String *url* = **"jdbc:mysql://"** + ***ip*** + **":"** + ***port*** + **"/"** + ***databaseName***;  
 *//注册msql驱动* **static**{  
 **try**{  
 Class.*forName*(**"com.mysql.jdbc.Driver"**);  
 Connection conn = DriverManager.*getConnection*(*url*,***user***,***password***);  
 }**catch** (ClassNotFoundException | SQLException e){  
 e.printStackTrace();  
 }  
 }

### 类User

定义用户表函数

**public class** User {  
 **private int id**;  
 **private** String **name**;  
 **private** String **password**;  
  
 **public int** getId() {  
 **return id**;  
 }  
  
 **public void** setId(**int** id) {  
 **this**.**id** = id;  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public void** setName(String name) {  
 **this**.**name** = name;  
 }  
  
 **public** String getPassword() {  
 **return password**;  
 }  
  
 **public void** setPassword(String password) {  
 **this**.**password** = password;  
 }  
}

### 类UserDao

实现相关数据库操作函数

**public static** User login (String name, String password) {  
 Connection conn = DBUtil.*getConnection*();  
 String sql = **"select \* from user where name=? and password=?"**;  
 User user = **null**;  
 PreparedStatement ps = **null**;  
 **try** {  
 ps = conn.prepareStatement(sql);  
 ps.setString(1,name);  
 ps.setString(2,password);  
 ResultSet rs = ps.executeQuery();  
 *// bean d导入* **if**(rs.next()){  
 user = **new** User();  
 user.setName(rs.getString(**"name"**));  
 user.setPassword(rs.getString(**"password"**));  
 }  
 DBUtil.*release*(rs, **null**, ps, conn);  
 }**catch** (Exception e){  
 e.printStackTrace();  
 }  
 **return** user;

### 类MainActivity

实现主活动

登入注册后跳转到聊天室页面

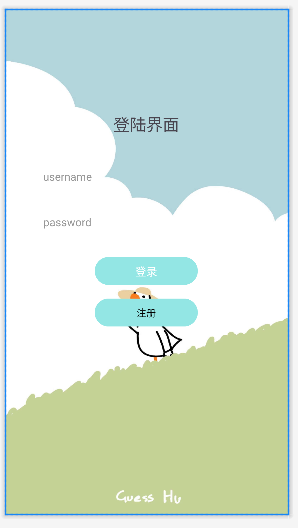
**public void** LoginTest(String name, String password){  
 **new** Thread(**new** Runnable() {  
 @Override  
 **public void** run() {  
 Intent intent = **null**;  
 Looper.*prepare*();  
 User user;  
 String ok = **"登陆成功!"**;  
 String fail = **"密码或账号错误，请重新登录！"**;  
 Connection con = DBUtil.*getConnection*();  
 **if**(con != **null**){  
 System.***out***.println(1);  
 }  
 user = UserDao.*login*(name, password);  
 **if**(user != **null**){  
 **if**(user.getName().equals(name) && user.getPassword().equals(password)){  
 intent = **new** Intent(MainActivity.**this**, FragmentActivity.**class**);  
 startActivity(intent);  
 *//Toast弹出消息* Toast.*makeText*(getApplicationContext(),ok,Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
 **else**{  
 *//Toast弹出不正确* Toast.*makeText*(getApplicationContext(),fail,Toast.***LENGTH\_SHORT***).show();  
 }  
 **try** {  
 DBUtil.*release*(**null**,**null**,**null**,con);  
 } **catch** (Exception e){  
 e.printStackTrace();  
 }  
  
 }  
 }).start();

### 类RegisterActivity

实现注册功能

**public void** RegisterTest(String name, String password){  
 **new** Thread(**new** Runnable() {  
 @Override  
 **public void** run() {  
 Looper.*prepare*();  
 Connection con = DBUtil.*getConnection*();  
 Message message = **handler**.obtainMessage();  
 User ExistsUser = UserDao.*findname*(name);  
 **boolean** IsSuccess;  
 **if**(ExistsUser != **null**){  
 message.**what** = 1;  
 message.**obj** = **"用户已经存在，请更换用户名重试！"**;  
 }**else**{  
 message.**what** = 2;  
 IsSuccess = UserDao.*register*(name, password);  
 **if**(IsSuccess) message.**obj** = **"注册成功！"**;  
 **else** message.**obj** = **"注册失败！"**;  
 }  
 **handler**.sendMessage(message);  
 **try** {  
 DBUtil.*release*(**null**,**null**,**null**,con);  
 } **catch** (Exception e){  
 e.printStackTrace();  
 }  
 }  
 }).start();

### activity\_main.xml



### activity\_register.xml



## （2）功能界面

通过Fragment实现底部导航栏，计划实现四个模块，聊天室，通讯录私聊，蓝牙聊天，个人信息修改。

#### 类FragmentActivity

操控导航栏

类MyFragment 核心界面的核心代码，四个模块的操作都在这里。

点击第一个导航项：

在线程中连接服务器，不断接收数据，并将数据通过消息机制传回主线程处理。

**new** Thread(**new** Runnable() {  
 @Override  
 **public void** run() {  
 **try** {  
*// socket = new Socket("127.0.0.1", 10000);  
// socket = new Socket("10.0.2.2", 9999);* **socket** = **new** Socket(**"10.0.2.2"**, 10001);  
*////// 申请连接* Log.*i*(**"提示"**,**"连接成功"**);  
*// 把这个连接移动到共享部分  
// socket = new Socket("10.0.2.2", 10001);  
//// 申请连接  
// Log.i("提示","连接成功");* InputStream inputStream = **socket**.getInputStream();  
 **byte**[] buffer = **new byte**[1024];  
 **int** len;  
 **while** ((len = inputStream.read(buffer)) != -1) {  
*// 循环不断从bufer中读取数据  
// 通过 inputStream.read(buffer) 方法返回实际读取的字节数。* String data = **new** String(buffer, 0, len);  
 *// 发到主线程中 收到的数据* Message message = Message.*obtain*();  
*// 实例化* message.**what** = 1;  
 message.**obj** = data;  
  
 all\_handler.sendMessage(message);  
*// 将消息发送给主线程进行处理和显示* }  
  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 }).start();

绑定按钮：用户输入信息点击，发送信息给服务端。设置发送信息类型*0群发1私聊2查询*

发送信息：outputStream.write((**socket**.getLocalPort() +**"//"** + data + **"//"** + df.format(**new** Date()) +**"//"** + **"0"**).getBytes(**"utf-8"**));

**all\_bt**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **final** String data = **all\_et**.getText().toString();  
*// client.send(0,data);* **new** Thread(**new** Runnable() {  
 @Override  
 **public void** run() {  
 **try** {  
  
 OutputStream outputStream = **socket**.getOutputStream();  
*// 0群发1私聊2查询* SimpleDateFormat df = **new** SimpleDateFormat(**"HH:mm:ss"**); *//设置日期格式* outputStream.write((**socket**.getLocalPort() +**"//"** + data + **"//"** + df.format(**new** Date()) +**"//"** + **"0"**).getBytes(**"utf-8"**));  
 outputStream.flush();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 }).start();  
 }

在Handle中接收消息

**else**{  
*// 聊天室* Log.*i*(**"提示"**,**"聊天室开始"**);  
 **if** (split[0].equals(localPort + **""**)) {  
 ChatMsg chatMsg = **new** ChatMsg(split[1], 0, split[2], **"我："**);  
 **all\_list**.add(chatMsg);  
 } **else** {  
 ChatMsg chatMsg = **new** ChatMsg(split[1], 1, split[2], (**"来自："** + split[0]));  
 **all\_list**.add(chatMsg);  
 }  
 Log.*i*(**"提示"**,**"输出信息ChatHandler:"**+**all\_list**.get(0).getMsg());  
  
 *// 向适配器set数据* **all\_adapter**.setData(**all\_list**);  
 **all\_rv**.setAdapter(**all\_adapter**);  
 LinearLayoutManager manager = **new** LinearLayoutManager(getActivity(), LinearLayoutManager.***VERTICAL***, **false**);  
 **all\_rv**.setLayoutManager(manager);  
*// 创建一个 LinearLayoutManager 对象，并将其设置为 RecyclerView 的布局管理器，用于控制列表项的显示方式。* }

点击第二个导航项：

点击刷新按钮，向服务器发送查询信息。

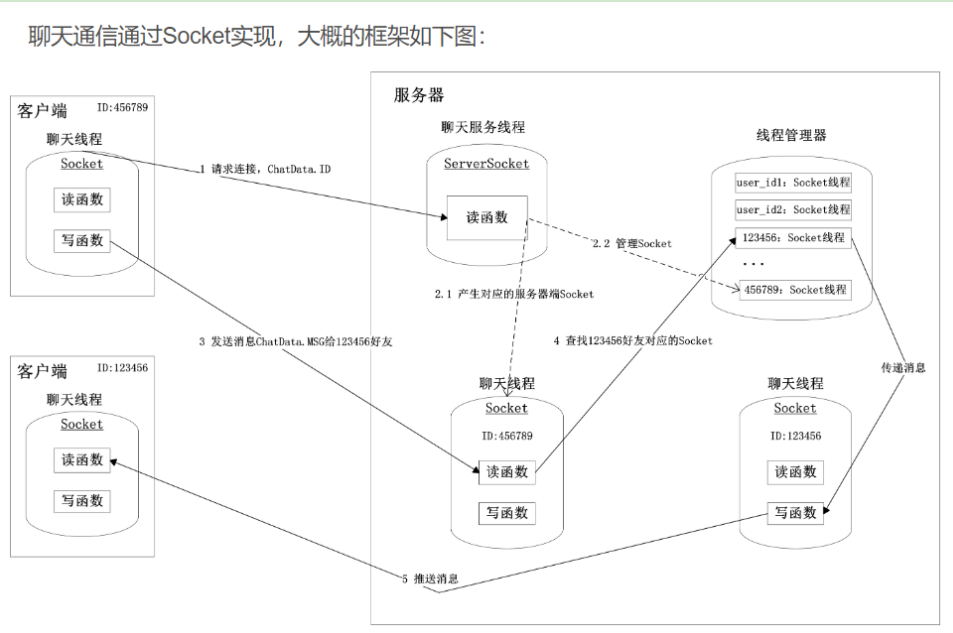
outputStream.write((**socket**.getLocalPort() + **"//"** + **"查询"** + **"//"** + df.format(**new** Date()) + **"//"** + **"2"**).getBytes(**"utf-8"**));

服务器返回目前连接的所有客户端ip。

在GlobalVariables中存储。

设置点击事件，点击在线用户后找到指定ip给服务器发送信息，要求服务器进行私聊。

**list\_bu**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Log.*i*(**"提示"**,**"刷新通讯录列表"**);  
  
 **new** Thread(**new** Runnable() {  
 @Override  
 **public void** run() {  
 **try** {  
 Log.*i*(**"提示"**,**"准备发出"**);  
*// 发送服务器查询请求2* OutputStream outputStream = **socket**.getOutputStream();  
 Log.*i*(**"提示"**,**"重新定义输出流"**);  
 SimpleDateFormat df = **new** SimpleDateFormat(**"HH:mm:ss"**); *//设置日期格式  
// 可以选择在日期的地方改为ip* outputStream.write((**socket**.getLocalPort() + **"//"** + **"查询"** + **"//"** + df.format(**new** Date()) + **"//"** + **"2"**).getBytes(**"utf-8"**));  
 outputStream.flush();  
 Log.*i*(**"提示"**,**"发出了查询"**);  
  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 }).start();

（这是张14年网图，看能不能自己画一下。）

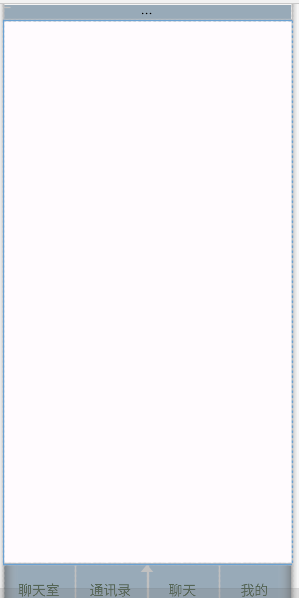
#### tab\_menu\_text.xml

设置导航项的选择器

<**selector xmlns:android="http://schemas.android.com/apk/res/android"**>  
 <**item android:color="#4CAF50" android:state\_selected="true"** />  
 <**item android:color="#546555"** />  
</**selector**>

#### activity\_fragment.xml

设置导航栏布局



## （3）服务器代码

技术要点：

解析收到信息

InputStream inputStream = socket.getInputStream();  
byte[] buffer = new byte[1024];  
int len;  
while ((len = inputStream.read(buffer)) != -1){  
 String text = new String(buffer,0,len);  
 System.*out*.println("收到的数据为：" + text);  
 //判断是否进行消息查询  
 String[] split = text.split("//");

群发

public static boolean sendMsgAll(String msg){  
 try {  
 for (Socket socket : *clientList*.values()) {  
// OutputStream outputStream = socket.getOutputStream();  
// outputStream.write(msg.getBytes("utf-8"));  
 synchronized (*lock*){  
 OutputStream outputStream = socket.getOutputStream();  
 outputStream.write(msg.getBytes("utf-8"));  
 }  
 }  
 return true;  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 return false;  
 }

私聊

else if (Objects.*equals*(split[3], "1")){  
// msg都不够用需要增加一个项来判断执行什么操作  
// 在时间那里改成id项  
// 拼接回去  
 StringBuilder sbText = new StringBuilder();  
 sbText.append(split[0]);  
 sbText.append("//");  
 sbText.append(split[1]);  
 sbText.append("//");  
 sbText.append(split[2]);  
 sbText.append("//");  
 sbText.append(split[3]);//选项  
  
 text = sbText.toString();  
  
 *sendMsg*(text,*clientList*.get(split[2]));  
 System.*out*.println(address+"和"+*clientList*.get(split[2])+"聊天");  
 }

// 私聊方法  
public static boolean sendMsg(String msg,Socket socket){  
 try {  
 synchronized (*lock*){  
 OutputStream outputStream = socket.getOutputStream();  
 outputStream.write(msg.getBytes("utf-8"));  
 }  
  
 return true;  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 return false;  
}

查询

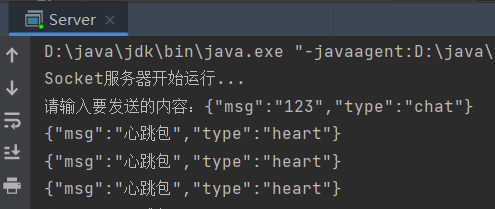
if(Objects.*equals*(split[3], "2")){  
 StringBuilder addressStringBuilder = new StringBuilder();  
 for (String address : *clientList*.keySet()) {  
 addressStringBuilder.append(address).append("\_");  
 }  
 String allAddresses = addressStringBuilder.toString();  
 // 移除最后一个分隔符 "\_"  
 if (allAddresses.endsWith("\_")) {  
 allAddresses = allAddresses.substring(0, allAddresses.length() - 1);  
 }  
  
 split[1]=allAddresses;  
 split[2]=address;  
 System.*out*.println(address+"发起查询："+allAddresses);  
  
// 拼接回去  
 StringBuilder sbText = new StringBuilder();  
 sbText.append(split[0]);//端口  
 sbText.append("//");  
 sbText.append(split[1]);//alladdress/data  
 sbText.append("//");  
 sbText.append(split[2]);//时间/selfaddress  
 sbText.append("//");  
 sbText.append(split[3]);//选项  
  
  
  
 text = sbText.toString();  
 // 在这里群发消息  
 *sendMsgAll*(text);  
// 可以选择私聊的  
// sendMsg(text,clientList.get(split[2]));  
 }

## （4）客户端模块介绍

### 1.聊天室模块

在查找资料后发现通过socket来实现客户端与服务端间的通信。

学习socket后，实现了客户端与服务端之间的对话，服务器将客户端发送的消息返回给客户端。利用心跳包机制实现断线重连。



进一步研究后改进为多客户端在同一个聊天室中聊天。使用RecyclerView实现聊天内容列表，通过Adapter设置两个xml文件区分我发送的消息和他人发送的消息。

#### 类ChatMsg

*//聊天消息的数据定义***public class** ChatMsg {  
 **private** String **msg**;  
*// 消息内容* **private int number**;  
*// 消息类型0，1（发，收）2(查询)* **private** String **time**;  
 *// 消息时间* **private** String **name**;  
*// id名称*

#### 类AllAdapter

聊天室消息界面适配器

通过判断时我发送的消息还是他人发送的消息，来设置不同列表项，加载到RecyclerView列表中。

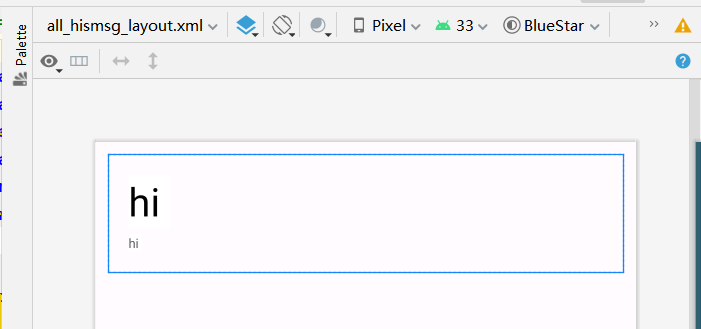
**public void** onBindViewHolder(@NonNull RecyclerView.ViewHolder holder, **int** position) {  
  
 **int** itemViewType = getItemViewType(position);  
 **if** (itemViewType==***T\_I***) {  
 T\_IViewHolder myViewHolder = (T\_IViewHolder) holder;  
 myViewHolder.**my\_tv**.setText(**Msg**.get(position).getMsg());  
 myViewHolder.**my\_tv2**.setText(**Msg**.get(position).getName());  
 }  
 **else**{  
 T\_HeViewHolder HisViewHolder = (T\_HeViewHolder) holder;  
 HisViewHolder.**his\_tv**.setText(**Msg**.get(position).getMsg());  
 HisViewHolder.**his\_tv2**.setText(**Msg**.get(position).getName());  
 }  
}

#### all\_chat.xml



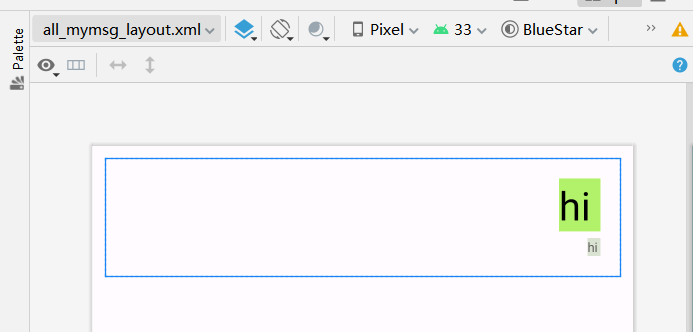
#### all\_hismsg\_layout.xml

聊天室中其他人发布的消息，显示发布人名称



#### all\_mymsg\_layout.xml

聊天室中我发布的消息，显示发布人名称：我



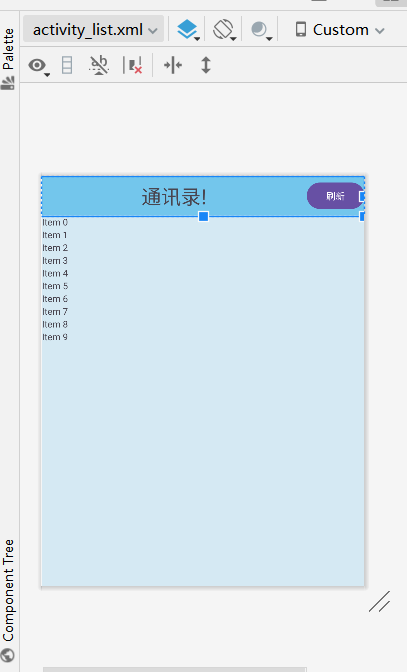
### 2.通讯录模块

#### 类GlobalVariables

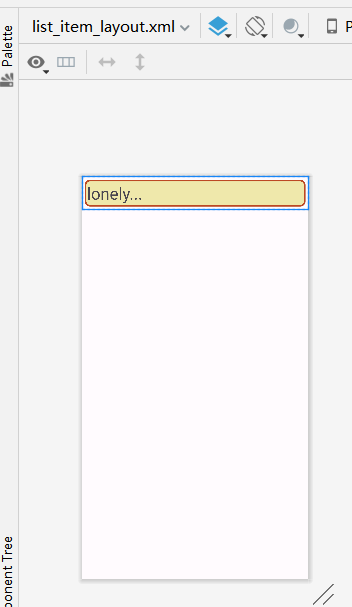
通过定义类设置全局变量，存储目前在线用户列表，socket对象信息，本机地址

**public class** GlobalVariables {  
 **private static** GlobalVariables *instance*;  
 **private int counter** = 0;  
 **private** String[] **Chatobject**=**null**;  
 **private** String **SelfAddress**=**null**;  
  
 **private** Client **client**=**null**;  
  
 **private** Socket **self\_socket**=**null**;

#### activity\_list.xml



list\_item\_layout.xml



boder.xml

设置边框样式



### 3.私聊模块

#### 类ListAdapter1

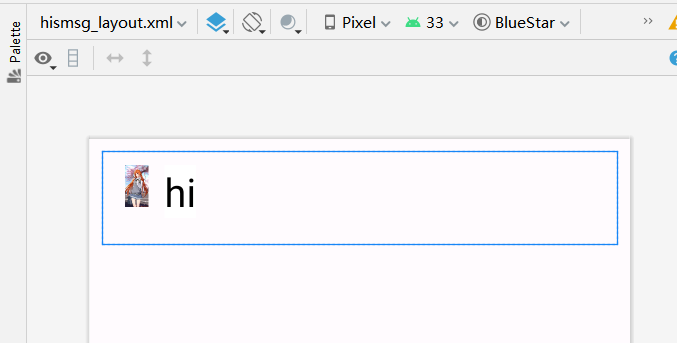
将在线用户列表信息加载到布局文件，然后自己定义接口实现点击事件，实现点击在线用户跳转到聊天界面。

**public interface** OnItemClickListener {  
 **void** onItemClick(**int** position);  
}

**public void** onBindViewHolder(@NonNull ItemViewHolder holder, **int** position) {  
 holder.**item\_tv**.setText(**object\_list**[position]);  
 holder.**itemView**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **int** clickedPosition = holder.getAdapterPosition();  
 **if** (**itemClickListener** != **null** && clickedPosition != RecyclerView.***NO\_POSITION***) {  
 **itemClickListener**.onItemClick(clickedPosition);  
 }  
 }  
 });  
}

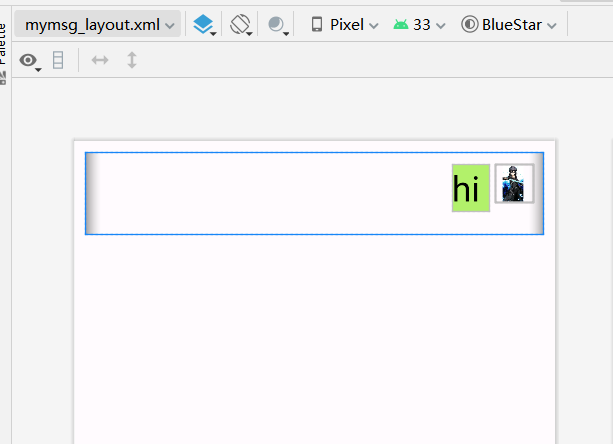
#### hismsg\_layout.xml

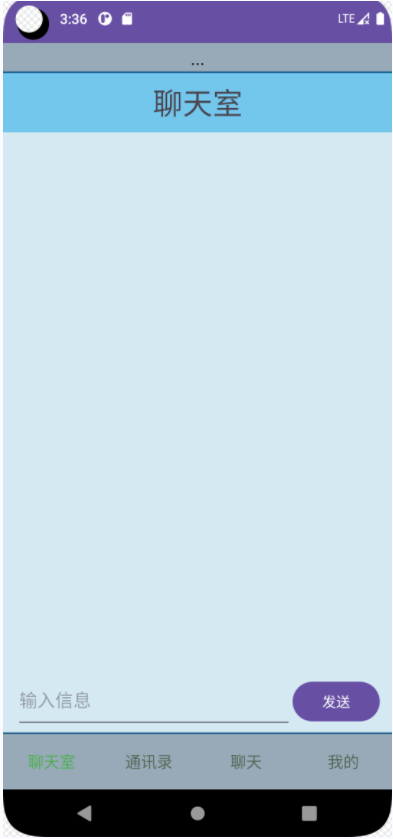
聊天中对方发的消息，居左，白色背景，显示头像



#### mymsg\_layout.xml

聊天中我发的消息，居右，绿色背景，显示头像





### 4.个人信息模块

#### 类UserInfo

展示个人信息，点击修改个人信息按钮跳转到修改信息界面

String name = "名字:";

String user\_say = "个人简介:";

// 用UserDao中写好的函数获取名字和个人简介append到name和usersay中

nameTextView.setText(name);

bioTextView.setText(user\_say);

editButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

// 跳转到修改信息界面

Intent intent = new Intent(UserInfo.this, EditActivity.class);

startActivity(intent);

}

});

#### 类EditActivity

编辑个人信息界面

编辑用户名和个人简介：

连接云数据库查找对应信息然后修改名字

public static void updateUserName(int userId, String newName) {

Connection conn = DBUtil.getConnection();

String sql = "UPDATE user SET name = ? WHERE id = ?";

PreparedStatement ps = null;

try {

ps = conn.prepareStatement(sql);

ps.setString(1, newName);

ps.setInt(2, userId);

ps.executeUpdate();

DBUtil.release(null,null,ps,conn);

} catch (SQLException e) {

e.printStackTrace();

} finally {

DBUtil.release(null, null, ps, conn);

}

}

public static void updateUserBio(int userId, String newBio) {

Connection conn = DBUtil.getConnection();

String sql = "UPDATE user SET user\_say = ? WHERE id = ?";

PreparedStatement ps = null;

try {

ps = conn.prepareStatement(sql);

ps.setString(1, newBio);

ps.setInt(2, userId);

ps.executeUpdate();

DBUtil.release(null,null,ps,conn);

} catch (SQLException e) {

e.printStackTrace();

} finally {

DBUtil.release(null, null, ps, conn);

}

}

修改完个人信息后跳转回个人信息界面：

Toast.makeText(EditActivity.this, "个人信息已保存", Toast.LENGTH\_SHORT).show();

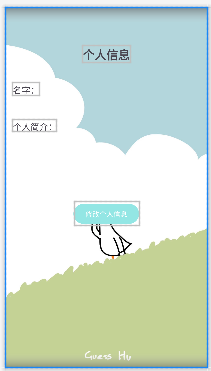
Intent intent = new Intent(EditActivity.this, UserInfo.class);

startActivity(intent);

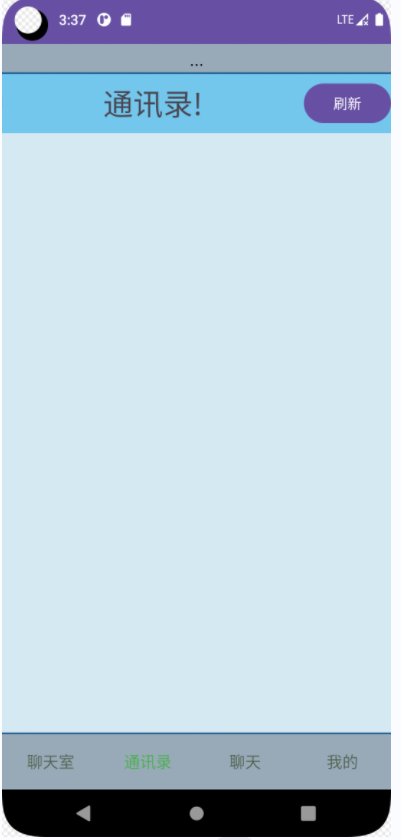
#### activity\_editinfo.xml

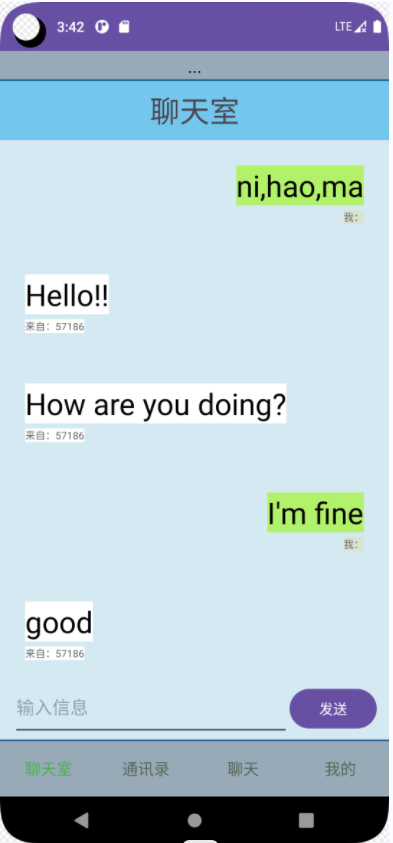


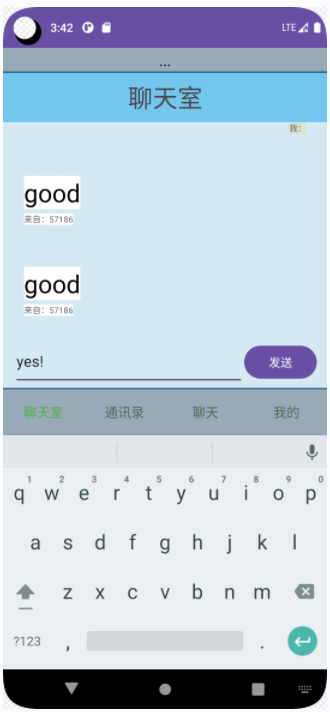
#### activity\_userinfo.xml



## （5）展示







# 二．开发问题记录

（1）[TCP/IP网络通信之Socket编程入门\_哔哩哔哩\_bilibili](https://www.bilibili.com/video/BV1eg411G7pW/?spm_id_from=333.788.recommend_more_video.3&vd_source=c9c5685ee953095f29282c9f6f382078)

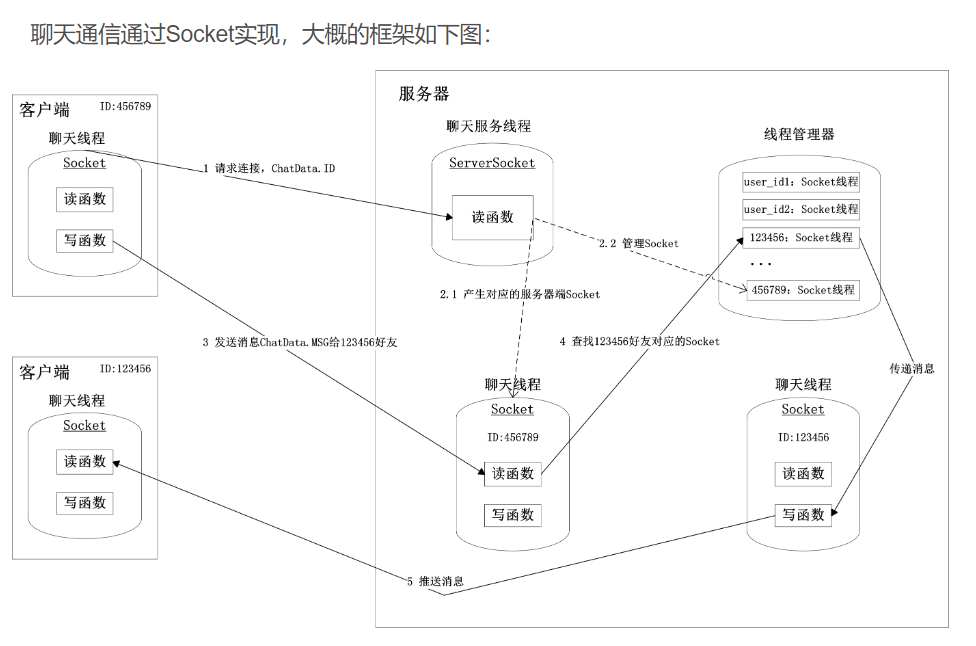
（2）使用 LinearLayout 布局将控件置于界面底部：

（3）Java类里的public static void main(String[] args)如何执行run

（4）Android打印信息使用Log吗

（5）Android的UI线程和后台线程的分离规则

（6）[(112条消息) Android聊天软件的开发(七)--聊天通信\_vaintwyt的博客-CSDN博客](https://blog.csdn.net/vaintwyt/article/details/32725771)



（7）import java.net.ServerSocket;这个和 import java.net.Socket;

（8）再详细解释一下accept()方法

（9）synchronized (this)

实现同步，防止其它线程访问

（10）socket对象中究竟有什么

属性：

InetAddress: 表示与远程主机通信的IP地址和主机名。

InputStream: 用于从套接字接收数据的输入流。

OutputStream: 用于向套接字发送数据的输出流。

LocalSocketAddress: 本地套接字地址。

RemoteSocketAddress: 远程套接字地址。

Port: 套接字连接的端口号。

isConnected(): 检查套接字是否已连接。

方法：

connect(): 连接到指定的远程主机和端口。

close(): 关闭套接字连接。

getInputStream(): 获取与套接字关联的输入流。

getOutputStream(): 获取与套接字关联的输出流。

getInetAddress(): 获取远程主机的IP地址。

getPort(): 获取远程主机的端口号。

getLocalAddress(): 获取本地主机的IP地址。

getLocalPort(): 获取本地主机的端口号。

isClosed(): 检查套接字是否已关闭。

Socket 对象的属性和方法提供了一种简单的方式来建立与远程主机的通信连接，并进行数据的发送和接收。它使得在网络编程中能够方便地处理套接字连接和数据交换的操作。

1. 用recyclerview编写聊天界面

[(112条消息) Android Studio新版本导入Recyclerview库的依赖\_添加recyclerview1.2.0\_tran\_sient的博客-CSDN博客](https://blog.csdn.net/tran_sient/article/details/101919083)

[(112条消息) RecyclerView 的使用（androidx）\_androidx recyclerview\_csdn1225987336的博客-CSDN博客](https://blog.csdn.net/csdn1225987336/article/details/109016013)

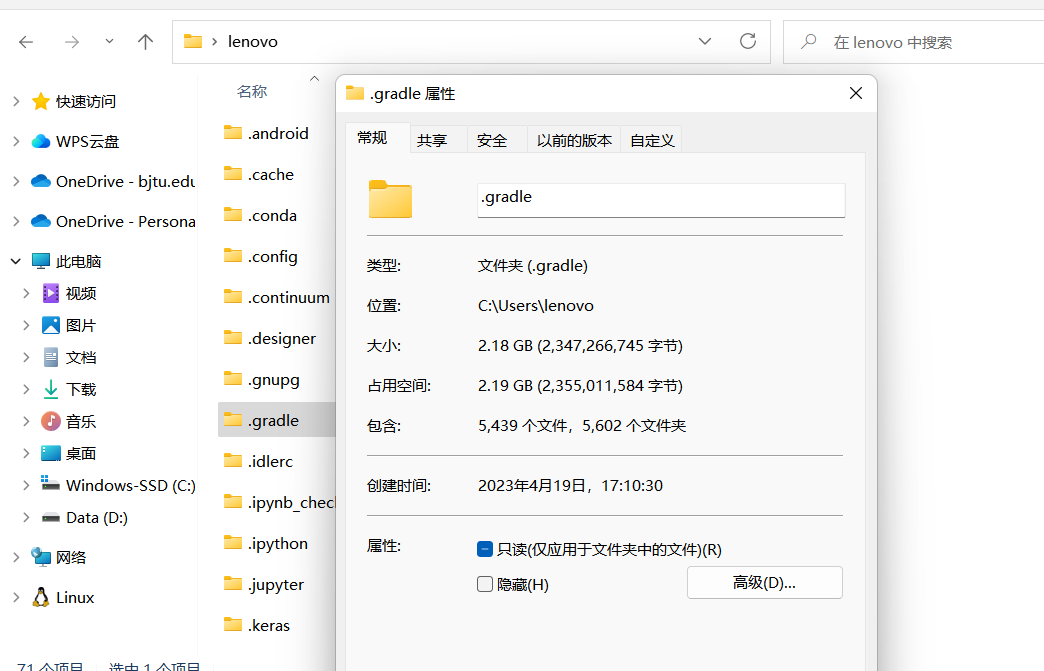
（12）用ImageView导入头像图片从相册中加载

[Android控件之ImageView(一) - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/78498723)

(13)Frament

[(112条消息) Fragment实现Android各个选项界面的切换\_胖胖的懒羊羊的博客-CSDN博客](https://blog.csdn.net/qq_44867340/article/details/105188532)

1. 一个名为 MyHandler 的内部类，它继承自 Handler 类，用于处理消息并更新 UI。



1. 进程一定要start

**new** Thread(**new** Runnable() {  
 @Override  
 **public void** run() {  
 **try** {  
 Socket socket = **new** Socket(**"localhost"**, 9999);  
 Log.*i*(**"提示"**,**"连接成功"**);  
*// 申请连接* }**catch** (Exception e){  
 e.printStackTrace();}  
 }  
  
 }).start();;

1. 安卓找本机ip

**socket** = **new** Socket(**"10.0.2.2"**, 10001);

待办：

1. 网络连接不稳定，心跳包机制，断线重连
2. 蓝牙连接
3. 指定联系人聊天
4. 通讯录

连接服务器数据库查找，设置本地数据库存储，

1. 语音聊天，图片传输
2. 朋友圈
3. 群聊设置
4. 获取通讯录/微信通讯录
5. Fragment功能

[5.2.1 Fragment实例精讲——底部导航栏的实现(方法1) | 菜鸟教程 (runoob.com)](https://www.runoob.com/w3cnote/android-tutorial-fragment-demo1.html)

如果在使用 <selector> 文件时出现报错，可能是由于以下几个原因：文件位置错误：请确保选择器文件的位置正确，通常放在 res/drawable 或 res/color 目录下。

推荐使用 FragmentManager 的替代类 FragmentMaangerX

1. 登入注册界面，信息部署到服务器数据库。
2. 个人信息修改界面
3. 学习一个侧滑界面
4. 图标库

[(112条消息) Android Studio使用IconFont阿里矢量图标库\_android 矢量图标ttf\_沧海龙腾LV的博客-CSDN博客](https://blog.csdn.net/xialong_927/article/details/107221892)

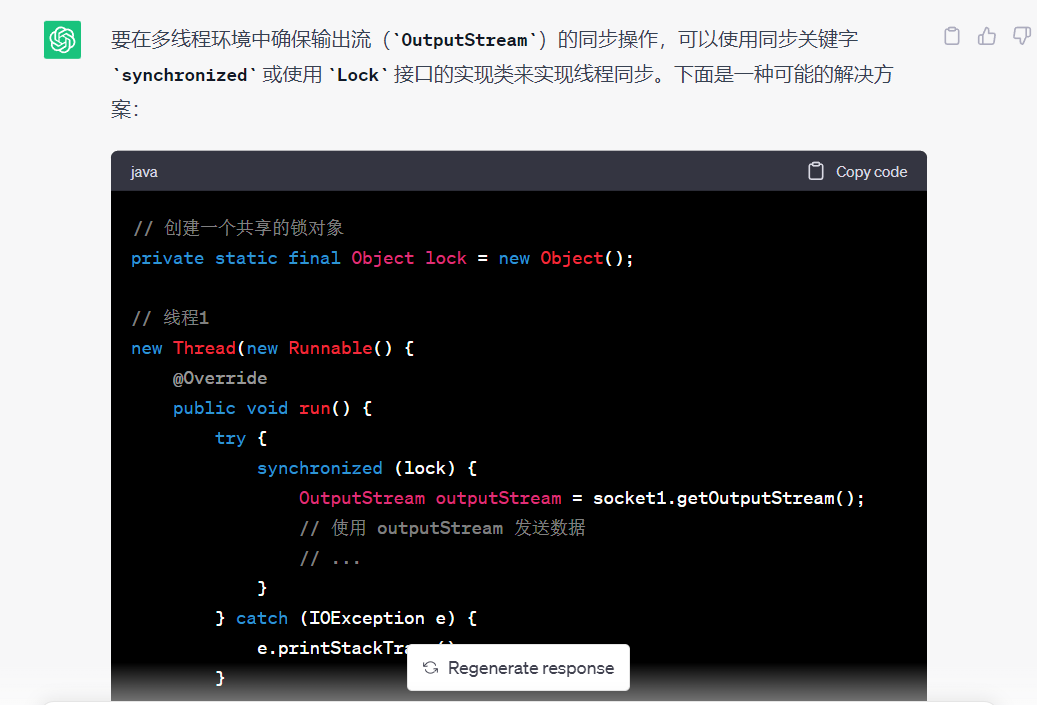
1. 数据库

用户id，用户名，用户ip，密码，（头像），好友，

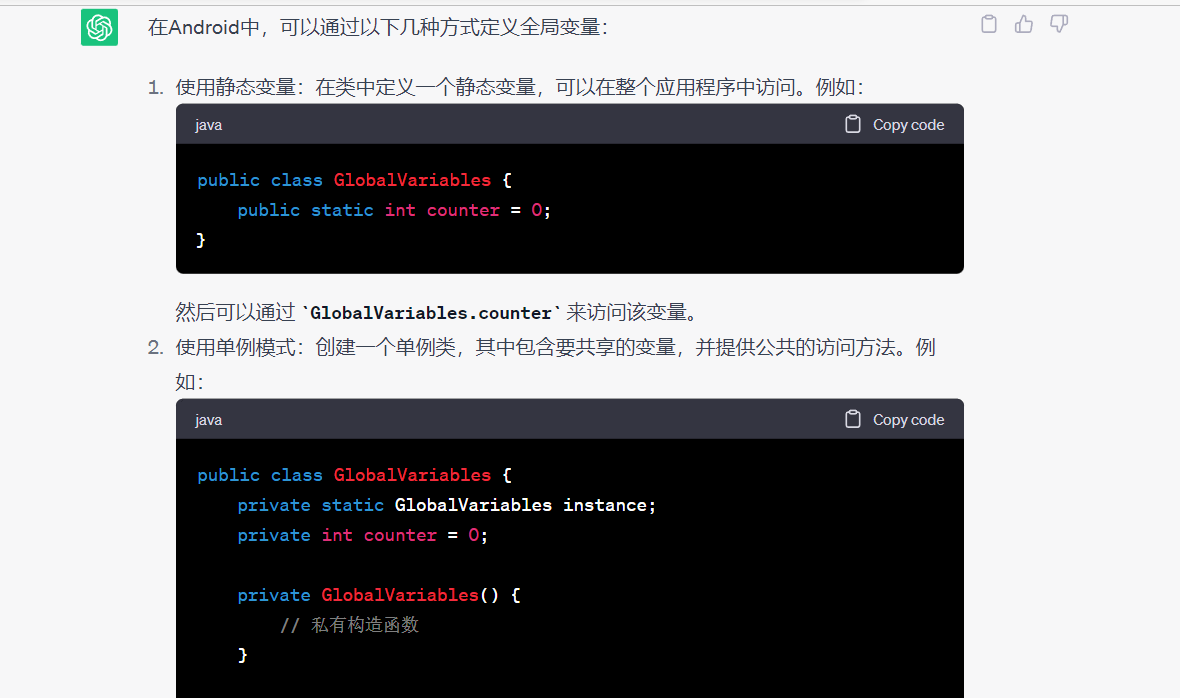
1. 端口号



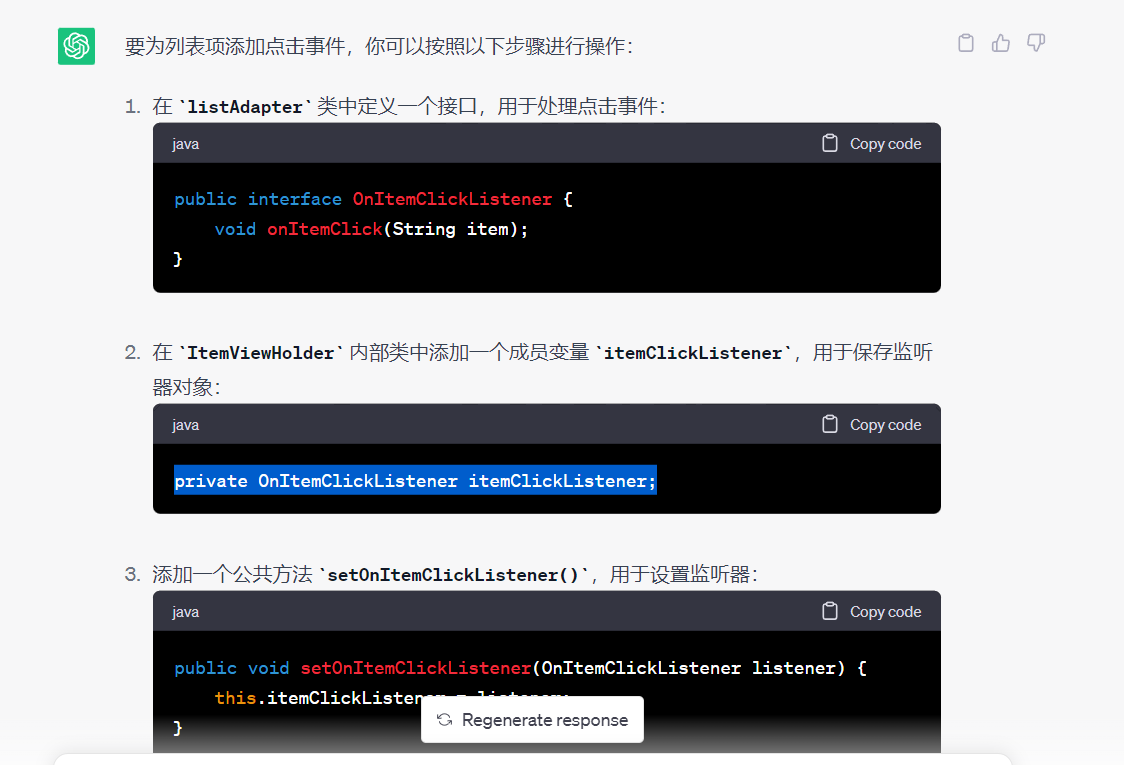
多线程访问同一个输出流出错



1. 定义全局变量



1. 给列表增加点击事件





提出的问题：

socket在Fragment的不同if条件句中如何传输？

如何在Fragment的一个条件句中跳到另一个条件语句？

1. 给列表增加点击事件

（40）那这种情况下，第一个socket连接会中断吗，因为socket好像是最开始定义的一个值

（41） 而且有一个问题没有考虑过，我切换到其它导航栏，聊天室socket好像是在一直接收信息吗？

（42）还有一个问题InputStream这个东西是一个代码里只能有一个吗

（43） 还有一个问题我这样再次建立一个连接那服务器那里不就重复又保存了一个连接吗

未解决问题

socket在Fragment的不同if条件句中如何传输？

导致通讯录功能最终失败。

在Fragment的第一个if条件句即聊天室模块中申请socket连接线程并监听输入，但是在Fragment的第二个if条件句即通讯录模块中无法使用全局变量中的socket，显示为空，通过单例模式全局类存储也失败。

如果在Fragment的第二个if条件句即通讯录模块中再次申请socket连接线程并监听输入。可以实现，但是我是在本机安卓中模拟多个手机进程，ip相同端口值不同，服务端代码会将每次连接都存储到一个数组中，点击通讯录模块和聊天室模块后一个手机用户，就会在服务端拥有两个连接，而查询后返回在线用户列表，返回给客户端后，客户端无法区分哪个是自己的那两个。也就是这个解决方法失败。

想出另外一个解决方法，额外写一个client类，在登入注册后调用client类建立socket连接，不断监听输入流，将信息传给主线程处理不同的UI界面，这里没有想清楚。然后在client类中写一个Handler类，写根据不同参数0、1、2选择查询、群发、私聊给服务器，但在发送给服务器查询信息，服务器返回目前在线用户数组后，客户端突然中断连接，无法接受信息。

如何在Fragment的一个条件句中跳到另一个条件语句？

导致私聊功能也失败，裁撤。

# 三．开发总结

不能指望着去学习现成项目，应该学会自己解决，规划设计思路，遇到问题去解决问题

开发就是不断降低自己的预期的过程

紧紧围绕着需求去完成任务。