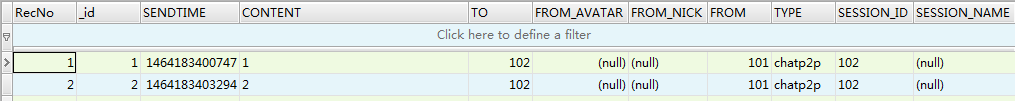
用户发送与接收的历史消息通过GreenDao框架保存在本地数据库中，在每次进入聊天界面时，从数据库中取出历史消息记录，并显示在界面上

1. 保存消息
2. 在监听到收到好友发来的消息后，将消息保存到数据库中，消息的javaBean中除了需要记录的消息内容、发送时间等属性，需要增加一个sessionID属性用于保存好友的id，以便于每次取出数据时根据该id即可取出，取出后再根据消息的类型（是发送还是接受到的消息）显示在聊天界面的两侧

imApp.getMsgDao().insert(msg);



1. 在当前用户发送消息给好友的时候，将发送的消息对象保存到数据库中

@OnClick(R.id.***send\_msg***)

**private** **void** sendMsg(View view){

imApp.getMsgDao().insert(msg);

}

二、每次打开和好友的对话界面时，则从数据库中获取历史消息记录，保存到一个集合中，然后通过adapter更新到界面

List<QQMessage> chatRecordList = imApp.getMsgDao().queryRaw("where SESSION\_ID = ? order by SENDTIME ASC", toAccount+"");

1. 好友每次发送一条最新消息给自己时，需要去数据库重新获取消息记录，如何确定在什么时候去数据库重新获取消息记录？

解决方案： 通过内容观察者ContentObserver去监听数据库数据的变化，如果监听到数据库数据发生了变化，则去数据库刷新数据到界面

1）创建一个内容观察者对象，在该对象的onChange方法（监听到数据变化时执行的回调方法）中去执行获取数据库数据的操作

/\*\*

\* 创建一个内容监听者用于监听数据库内容的变化

\*/

**private** ContentObserver observer = **new** ContentObserver(**new** Handler()) {

// ContentOberver监听到数据更新后执行的回调方法

// 2.2版本执行的回调

@Override

**public** **void** onChange(**boolean** selfChange) { **super**.onChange(selfChange);

System.***out***.println("低版本模拟器接收到消息数据变化");

msgList.clear();

updateMsgListFromDb();

msgAdapter.notifyDataSetChanged();

displayLatestMsg();

}

// 4.0版本执行的回调

@SuppressLint("NewApi")

@Override

**public** **void** onChange(**boolean** selfChange, Uri uri) {

**super**.onChange(selfChange, uri);

System.***out***.println("高版本模拟器接收到消息数据变化");

}

};

1. 在监听好友发来消息的监听器中，通过ContentResolver通知所有的ContentObserver数据库中的内容发生了改变

Uri uri = Uri.*parse*("content://"+QQMessageDao.**class**.getSimpleName());

getContentResolver().notifyChange(uri, **null**);

1. 在打开与好友对话的Activity界面时，通过ContentResolver设置ContentObserver监听对象

Uri uri = Uri.*parse*("content://"+QQMessageDao.**class**.getSimpleName());

// 注册数据库内容变化的监听

getContentResolver().registerContentObserver(uri, **true**, observer);

1. 在关闭该Activity时，取消监听

@Override

**protected** **void** onDestroy() {

**super**.onDestroy();

// 在关闭和好友进行聊天的界面时，取消监听数据库数据变化的ContentObserver

getContentResolver().unregisterContentObserver(observer);

}

即可实现实时通讯并显示界面的效果：