XMPPConnection mXmppConnection = **null**;

//连接服务器

**private** **void** connectServer(){

//指定连接到服务器的参数：网站和端口

ConnectionConfiguration mConnectionConfiguration = **new** ConnectionConfiguration("192.168.0.178",5222);

//初始化XMPPConnection连接

mXmppConnection = **new** XMPPConnection(mConnectionConfiguration);

//连接上XMPP服务器

**try** {

mXmppConnection.connect();

} **catch** (XMPPException e) {

e.printStackTrace();

}

}

//创建联系人

**private** **void** createNewUser(XMPPConnection mXmppConnection){

AccountManager mAccountManager = mXmppConnection.getAccountManager();

**try** {

//用指定的用户名和密码创建用户

mAccountManager.createAccount("NewUser", "111111");

} **catch** (XMPPException e) {

e.printStackTrace();

}

}

//登录代码

**private** **void** login(XMPPConnection mXMPPConnection){

//使用指定的用户名和密码登录

**try** {

mXMPPConnection.login("NewUser", "111111");

} **catch** (XMPPException e) {

e.printStackTrace();

}

}

//用户下线

**private** **void** logout(XMPPConnection mXMPPConnection){

mXMPPConnection.disconnect();

}

//隐身

**private** **void** dismisslogin(XMPPConnection mXmppConnection){

//Presence类可以设置用户的是否在线的类型属性

Presence mPresence = **new** Presence(Presence.Type.*unavailable*);

//向服务器提交该属性

mXmppConnection.sendPacket(mPresence);

}

//获取联系人列表

**private** **void** getContactsList(XMPPConnection mXmppConnection){

//新建Roster对象

Roster roster = mXmppConnection.getRoster();

//获取联系人列表的集合

Collection<RosterEntry> rg = roster.getEntries();

//遍历联系人列表

**for**(RosterEntry rosterEntry : rg){

//获取联系人列表

Log.*d*("TAG", rosterEntry.getUser());

}

}

//获取联系人状态

**private** **void** getContactsStatues(XMPPConnection mXmppConnection){

//新建Roster对象

Roster roster = mXmppConnection.getRoster();

//获取联系人列表的集合

Collection<RosterEntry> rg = roster.getEntries();

//遍历联系人列表

**for**(RosterEntry rosterEntry:rg){

//获取好友状态对象

Presence presence = roster.getPresence(rosterEntry.getUser());

//获取好友状态

Log.*d*("TAG", presence.getStatus());

}

}

//设置状态（个性签名）

**private** **void** setStatus(XMPPConnection mXmppConnection){

//新建状态为在线

Presence mPresence = **new** Presence(Presence.Type.*available*);

//设置状态信息

mPresence.setStatus("set my status here");

//发送配置包，设置在线状态

mXmppConnection.sendPacket(mPresence);

}

//添加联系人

**private** **void** addContacts(XMPPConnection mXmppConnection){

//新建Roster对象

Roster roster = mXmppConnection.getRoster();

**try** {

//添加联系人，3个参数分别为：用户名（JID）、昵称和分组

roster.createEntry("windows2@openfire", **null**, **new** String[]{"friends"});

} **catch** (XMPPException e) {

//抛出异常

e.printStackTrace();

}

}

//删除联系人

**private** **void** deleteContacts(XMPPConnection mXmppConnection){

//新建Roster对象

Roster roster = mXmppConnection.getRoster();

**try** {

//删除联系人

roster.removeEntry(roster.getEntry("windows1@openfire"));

} **catch** (XMPPException e) {

e.printStackTrace();

}

}

//监听联系人添加信息

/\*\*

\* 收到订阅请求时候的XMPP信息如下：

\* <presence id="rm1952\_24" to="newuser@openfire" from="windows1@openfire" type="subscribed"></presence>

\* 收到被删除订阅的时候的XMPP信息如下：

\* <presence to="newuser@openfire" from="windows1@openfire" type="unsubscribe"></presence>

\*/

**private** **static** **void** addListener(XMPPConnection mXmppConnection){

//新建消息过滤器

PacketFilter filterMessage = **new** PacketTypeFilter(Presence.**class**);

//创建监听器

PacketListener myListener = **new** PacketListener() {

@Override

**public** **void** processPacket(Packet packet) {

//消息包赋值给一个新的Presence

**final** Presence mPresence = (Presence) packet;

**if**(mPresence.getType().equals(Presence.Type.*subscribe*)){

//收到请求订阅的信息，可以在该处处理

}**else** **if**(mPresence.getType().equals(Presence.Type.*unsubscribe*)){

//接收到取消订阅信息，可以在该处处理

}

//packet.toXML()可以将接收到的信息转化为XML字符串格式

}

};

//在XMPP连接上注册该监听

mXmppConnection.addPacketListener(myListener, filterMessage);

}

//接收消息

**private** **static** **void** addMessageListener(XMPPConnection mXmppConnection){

//添加消息监听

mXmppConnection.getChatManager().addChatListener(**new** ChatManagerListener() {

@Override

//创建消息

**public** **void** chatCreated(Chat chat, **boolean** createdLocally) {

//检测消息来源,是否为本地发出的

//这里为接收到的消息

**if**(!createdLocally){

//监听接收到的消息

chat.addMessageListener(**new** MessageListener() {

@Override

**public** **void** processMessage(Chat chat, Message message) {

//打印接收到的消息

Log.*d*("TAG", "接收到的消息为："+message.getBody());

}

});

}

}

});

}

//发送消息

**private** **void** sendMessage(XMPPConnection mXmppConnection){

//新建一个消息线程

Chat chat = mXmppConnection.getChatManager()

//指定接收人，并添加消息监听

.createChat("windows1@openfire", **new** MessageListener() {

@Override

//处理消息

**public** **void** processMessage(Chat chat, Message message) {

//打印接收到的消息

Log.*d*("TAG", "Received message:"+message);

}

});

//创建新的消息

Message newMessage = **new** Message();

//设置消息内容

newMessage.setBody("Hello!");

**try** {

//发送消息

chat.sendMessage(newMessage);

} **catch** (XMPPException e) {

e.printStackTrace();

}

}