# 基础

## connect

### Connection

1、封装一个SocketChannel

2、重要字段

1. connectionDescription：连接描述符
2. socketChannel：套接字信道
3. selectors：该连接注册过的Selector集合
4. pipeLineTask：该连接所在的PipeListTask对象
5. messageReader：读取中间件
6. messageWriter：写入中间件
7. writeMessages：待写入的消息队列
8. status：状态，NORMAL正常，CANCEL不能接受新的消息
9. recentActiveTimeStamp：用于服务端超时下线功能

3、重要方法

1. offerMessage：向消息队列压入新消息
2. pollMessage：从消息队列取出消息
3. registerSelector：注册Selector，会调用关联的PipeLineTask对象
4. cancel：改变status状态，不再接受新消息
5. isActive：status是否等于NORMAL
6. activeNow：更新recentActiveTimeStamp

### ConnectionDescription

1、连接描述符，用于封装Connection的相关信息

2、重要字段

1. source：信源
2. destination：信宿
3. type：类型，目前只有一种文本类型，方便以后扩展(传输文件视屏等，需要开启新的连接，用于区分新源新宿相同，类型不同的连接)
4. sessionDescriptions：会话描述符集合

3、重要方法

1. addSessionDescription：增加会话描述符
2. removeSessionDescription：删除会话描述符
3. getSessionDescriptions：获取会话描述符集合

4、重写了hashCode和equals方法，可以作为Map的关键字

### SessionDescription

1、会话描述符

2、重要字段

1. fromUserName
2. toUserName
3. isGroupChat

## handler

### WindowHandler

1、回调接口

2、重要方法

1. onSuccessful
2. onFailure

## interceptor

### MessageInterceptor

1、拦截器接口

1. Object intercept(MessageInvocation messageInvocation) **throws** IOException;

### MessageInvocation

1、触发器接口

1. Object process() **throws** IOException;

### ProxyMethodInvocation

1、继承自MessageInvocation接口，增加了如下方法

1. Method getMethod();
2. Object getTarget();
3. Object[] getArguments();

### ReflectionMethodInvocation

1、ProxyMethodInvocation的实现类，用于封装一次反射调用中的所有信息

2、每次调用都会重新生成该对象!!!

## pipe

### PipeLineTask

1、任务接口，实现Runnable

1. /\*\*
2. \* 返回该任务绑定的线程
3. \*
4. \* @return
5. \*/
6. Thread getBindThread();
8. /\*\*
9. \* 返回读选择器
10. \*
11. \* @return
12. \*/
13. Selector getReadSelector();
15. /\*\*
16. \* 返回写选择器
17. \*
18. \* @return
19. \*/
20. Selector getWriteSelector();
22. /\*\*
23. \* 将Connection注册到当前PipeLineTask中
24. \*/
25. **void** registerConnection(Connection connection);
27. /\*\*
28. \* 让指定Connection断开连接
29. \*/
30. **void** offLine(Connection connection);
32. /\*\*
33. \* 移除Connection，与offLine不同，不会关闭Connection，仅仅将其移出当前PipeLine的管理
34. \*/
35. **void** removeConnection(Connection connection);
37. /\*\*
38. \* 返回当前PipeLineTask管理的Connection
39. \*
40. \* @return
41. \*/
42. Set<Connection> getConnections();
44. /\*\*
45. \* 返回当前PipeLine处理的Connection数量
46. \*
47. \* @return
48. \*/
49. **int** getConnectionNum();

### AbstractPipeLineTask

1、实现PipeLineTask，提供基本实现

2、增加start和offLinePostProcess抽象方法

## protocol

### Protocol

1、文本协议

2、定义了将信息分为三部分，Control、Header、Body

3、重要方法

1. String wrap(Message message)
2. Message parse(String messageString)
3. findEndIndexOfMessage(MessageBuffer messageBuffer)

### Message

1、含有Control、Header、Body三个部分

## reader

### MessageReader

1、消息读取中间件接口

1. /\*\*
2. \* 从信道中读取数据，如果已经有Message解析好了，则返回，允许抛出异常，交给外界处理
3. \* 必须返回所有，否则每次一有数据准备好，如果只取一个Message，那么会导致Message堆积在这里
4. \*
5. \* @param connection
6. \* @return
7. \* @throws IOException
8. \*/
9. List<Message> read(Connection connection) **throws** IOException;

### MessageBuffer

1、消息读取中间件的二级缓存，用于从ByteBuffer中读取数据，并缓存，当缓存的字节包含一个Message后，将其取出，存入Message消息列表

### MessageReaderImpl

1、消息读取中间件的基本实现

2、重要字段

1. messageQueue：已读取的消息
2. messageBuffer：MessageBuffer对象，每个消息读取中间件私有
3. byteBuffer：ByteBuffer对象，一级缓存，用于从信道读取数据
4. protocol：字节的解析必须依赖协议，Protocol对象

### MessageReaderFactory

1、工厂接口，用于生产MessageReader实例

### DefaultMessageReaderProxyFactory

1、工厂接口的实现类，利用JDK动态代理，配合拦截器技术，生产织入了增强逻辑的读取中间件

## writer

### MessageWriter

1、消息写入中间件接口

1. /\*\*
2. \* 将指定Message中的信息写入指定信道，允许抛出异常，交给外界处理
3. \*
4. \* @param message
5. \* @param connection
6. \* @throws IOException
7. \*/
8. **void** write(Message message, Connection connection) **throws** IOException;

### MessageWriterImpl

1、标准实现

2、重要字段

1. byteBuffer：一级缓存，用于与SocketChannel直接通信
2. protocol：协议，用于将Message封装成String，再由写入中间件序列化成字节

### MessageWriterFactory

1、工厂接口，用于生产消息写中间件

### DefaultMessageWriterProxyFactory

1、工厂接口的实现类，利用JDK动态代理，配合拦截器技术，生产织入了增强逻辑的写入中间件

# 服务端

## connection

1、继承自Connection

2、重要字段

1. isMainConnection：用于区分是客户端的主界面连接还是会话连接
2. isRefused：在监听到连接后，dispatcher方法进行分发，若超过服务器的限额，则会将该连接标记为refuse，但是只有主界面连接才会被真正拒绝

## interceptor

### AbstractServerMessageInterceptor

1、服务端读写拦截器的公共部分

2、重要字段

1. serverConnectionDispatcher：服务器单例对象
2. protocol：协议

### ServerMessageReaderInterceptor

1、服务器读拦截器，定义了消息读取后的处理流程

### ServerMessageWriterInterceptor

1、服务器写拦截器，定义了消息写入后的处理流程

## pipeline

### ServerConnectionListener

1、监听任务，继承自Runnalbe

2、利用ServerSocketChannel，以阻塞的方式监听新连接

### ServerSessionTask

1、继承自AbstractPipeLineTask，服务端的处理流程

## ui

## utils

### ServerUtils

1、定义一些常量以及生成Message的方法

### ServerGroupInfo

1、定义群聊的信息

2、重要字段

1. groupName：群聊名
2. groupSessionConnectionMap：会话连接映射，处于当前群聊中的会话连接

## ServerConnectionDispatcher

1、服务端单例对象，用于持有一些全局对象，便于多个线程取用

2、重要字段

1. pipeLineTasks：PipeLineTask任务集合
2. loadBalancingLock：负载均衡锁，保证负载均衡时，所有线程处于全局安全点
3. messageReaderFactory：读取工厂
4. messageWriterFactory：写入工厂
5. executorService：线程池，用于执行PipeLineTask
6. mainConnectionMap：主线程连接映射
7. sessionConnectionMap：会话连接映射
8. groupInfoMap：群聊信息
9. nextLoadBalancingTimeStamp：下一次负载均衡时刻

3、重要方法

1. dispatch：分发监听到的连接到PipeLineTask中
2. checkLoadBalancing：服务端线程负载均衡

## 流程

### 服务端启动



### 服务端终止



### 监听到新连接



### 服务器拒绝连接

### 读拦截器



#### processSystemMessage

#### processLoginInMessage

#### processLoginOutMessage

#### processOpenSessionMessage

#### processCloseSessionMessage

#### processNormalMessage

### 写拦截器



# 客户端

## connection

### ClientMainConnection

1、主线程连接，继承自Connection

2、重要字段

1. bindMainWindow：主界面连接关联的主界面对象

### ClientSessionConnection

1、会话连接，继承自Connection

2、重要字段

1. sessionWindowMap：当前会话连接的会话窗口
2. groupSessionWindowMap：当前会话连接的群聊会话窗口

## interceptor

### ClientMainTaskReaderInterceptor

1、主界面连接读拦截器

### ClientMainTaskWriterInterceptor

1、主界面连接写拦截器

### ClientSessionTaskReaderInterceptor

1、会话连接读拦截器

### ClientSessionTaskWriterInterceptor

1、会话连接写拦截器

## pipeline

### ClientSessionTask

1、继承自AbstractPipeLineTask，客户端处理流程

2、重要字段

1. clientConnectionDispatcher：客户端单例对象

## ui

### LoginWindow

1、登录界面

### MainWindow

1、主界面连接

2、重要字段

1. bindMainConnection：主界面关联的主界面连接，该主界面私有
2. sessionWindowMap：会话窗口映射
3. groupSessionWindowMap：群聊会话窗口映射
4. groupNames：群聊名称列表，仅用于判断重复，接收到服务器刷新消息后会更新

### SessionWindow

1、会话窗口

2、重要字段

1. header：消息头，用于封装此会话的fromUserName和toUserName
2. bindMainWindows：会话窗口关联的主界面，多个会话窗口共享
3. bindConnection：会话窗口关联的会话连接，多个会话窗口共享

### GroupSessionWindow

1、群聊会话窗口

2、重要字段

1. header：消息头，用于封装此会话的fromUserName和groupName
2. bindMainWindow：会话窗口关联的主界面，多个会话窗口共享
3. bindConnection：会话窗口关联的会话连接，多个会话窗口共享

## utils

### ClientUtils

1、定义一些常量以及生成Message的方法

## ClientConnectionDispatcher

1、客户端单例对象，用于持有一些共享变量，供多线程取用

2、重要字段

1. pipeLineTasks：PipeLineTask任务集合
2. loadBalancingLock：负载均衡锁
3. mainTaskMessageReaderFactory：主界面连接读中间件工厂
4. sessionTaskMessageReaderFactory：会话连接读中间件工厂
5. mainTaskMessageWriterFactory：主界面连接写中间件工厂
6. sessionTaskMessageWriterFactory：会话连接写中间件工厂
7. executorService：线程池
8. mainConnectionMap：主界面连接映射
9. sessionConnectionMap：会话连接映射
10. nextLoadBalancingTimeStamp：下一次负载均衡时刻

3、重要方法

1. createAndDispatch：新建连接并分发到PipeLineTask中
2. checkLoadBalancing：客户端线程负载均衡

# 流程

## LogIn



## LogOut



1、注意，在服务端，如果同时有主界面与会话界面，当主界面被关闭时，会通过主界面连接发送LogOutMessage，通过会话连接发送CloseSessionMessage

1. 当LogOutMessage先被处理，会话连接被online了

* 如果尚未收到CloseSessionMessage，注销结束
* 如果收到了CloseSessionMessage，于是又会执行一次online操作

1. 当CloseSessionMessage先被处理，会话连接依次删除所有的会话，然后被onLine，然后处理LogOutMessage，会再次online会话连接，只是此时，执行logOutNotify方法时，找不到会话连接了，因此不会发送离线通知

2、这个会话界面会被online两次，没有关系，都是Map和Set，允许空删，反正保证删除即可

## OpenSession



## CloseSession



## CreateGroupInfo



## SendNormalMessage

