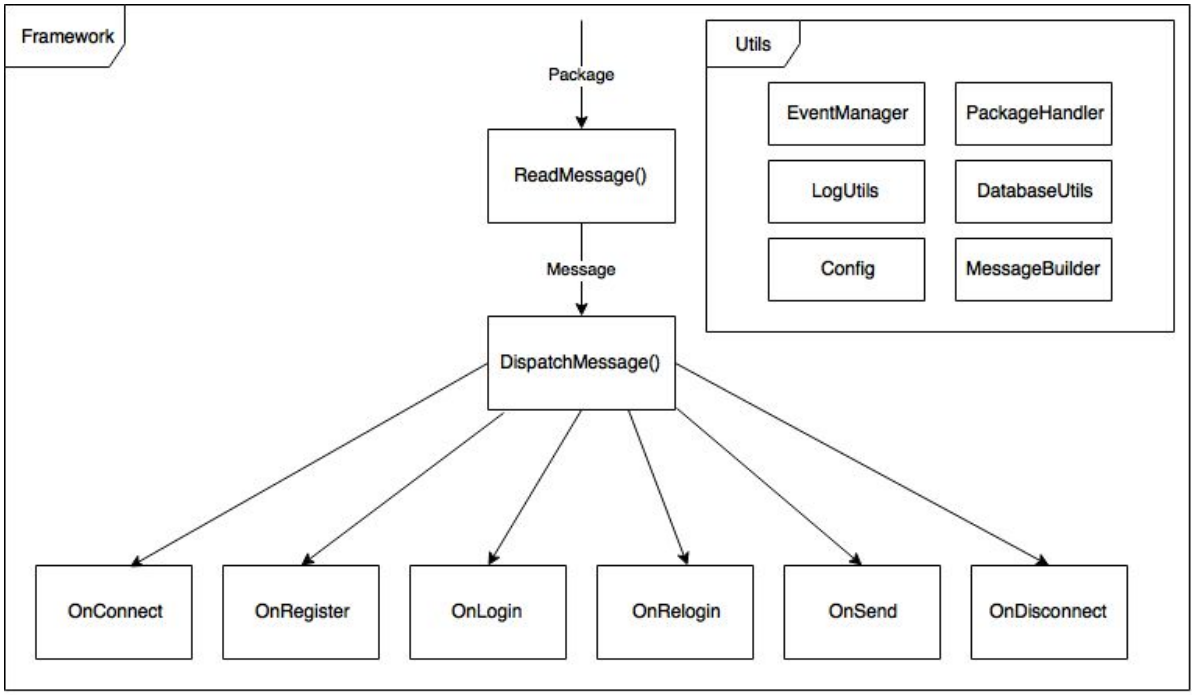
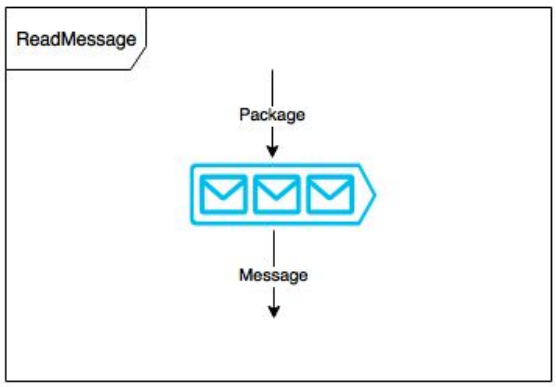
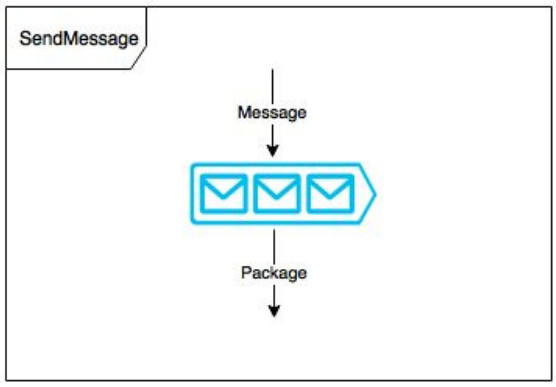
复用文档

1. 架构设计


Framework
Package
ReadMessage()
Message
DispatchMessage()
Utils
EventManager
PackageHandler
LogUtils
DatabaseUtils
Config
MessageBuilder
OnConnect
OnRegister
OnLogin
OnRelogin
OnSend
OnDisconnect


ReadMessage
Package
Message


SendMessage
Message
Package

根据本项目的需求，我们设计出了上图所示的架构，客户端与服务端可以直接复用这个架构。我们采用的是基于事件驱动的异步架构，首先，消息通过Java的异步Socket库NIO2读入进来，然后根据读进来的消息标签，触发对应的消息处理函数。

对于本需求，我们定义的事件有以下几种：连接事件(OnConnect)，注册事件(OnRegister)，登录事件(OnLogin)，重新登录事件(OnRelogin)，发送事件(OnSend)，断开事件(OnDisconnect)。连接事件(OnConnect)总是第一个发生，断开事件(OnDisconnect)总是最后一个发生，其他事件没有先后次序的限制。

对于Socket编程，由于存在粘包和半包的问题，我们封装了原生的NIO库，通过加入队列作为缓冲池，将读入的消息放入缓冲池，然后判断特殊分隔符(“\u0004”)来处理，取得了很好的效果。同样的对于发送消息，由于存在写Socket的独占问题（同一时间只能有一个对象对socket进行写操作），我们也加入了一个队列作为缓冲池处理独占问题。

最后对于一些通用的操作，比如数据库操作，记录操作，我们将他们封装到独立的模块里，以供下一个项目使用。

2. 可复用构件

2.1 事件模块

2.1.1 EventManager

函数名	功能说明	输入	输出
addEventListener	添加事件	String event, EventListener listener	无
triggerEvent	触发事件	String event	无

2.1.2 用例

```
EventManager mEventManager = new EventManager();
mEventManager.addEventListener("click", new EventListener() {  //添加click事件
        @Override
        public void run(HashMap<String, String> args) {
                //Do something
        }
});
mEventManager.triggerEvent("click");                            //触发click事件
```

2.2 分包模块

2.2.1 PackageHandler

函数名	功能说明	输入	输出

addPackage	添加数据包	ByteBuffer buf	无
hasPackage	判断是否有数据包	无	Boolean
getPackage	获取数据包	无	String message

2.2.2 用例

```
PackageHandler mPackageHandler = new PackageHanlder();
mPackageHandler.addPackage(buf);                          //添加数据包
while (mPackageHandler.hasPackage) {                      //判断是否还有数据包
       String message = mPackageHandler.getPackage();     //获取数据包
}
```

2.3 配置模块

2.3.1 Config

函数名	功能说明	输入	输出
getProperty	获取配置信息	String key	String
getProperty	获取配置信息，如果没有则使用默认值	String key, String default	String
setProperty	设置配置信息	String key, String value	无

2.3.2 用例

```
Config.getConfig().setProperty("host", "localhost");
String host = Config.getConfig().getProperty("host");      //host->localhost
```

2.4 数据库模块

2.4.1 DatabaseUtils

函数名	功能说明	输入	输出
isExisted	判断用户是否存在	String username	Boolean
isValid	判断用户是否合法	String username, String password	Boolean
createAccount	创建用户	String username, String password	Boolean

2.4.2 用例

```
DatabaseUtils.createAccount("test1", "test1test1");
```

```
Boolean exist = isExisted("test1");              //True
Boolean valid = isValid("test1");                //True
```

2.5 记录模块

2.5.1 LogUtils

函数名	功能说明	输入	输出
log	记录数据到文件中	LogType type, int… numbers	无

2.5.2 用例

```
LogUtils.log(LogUtils.LogType.CLIENT, 1, 2, 3, 4);
```

2.6 消息转换模块

2.6.1 MessageBuilder

函数名	功能说明	输入	输出
add	添加json的key,value	String key, String value	MessageBuilder
build	将添加好的key,value转换为String	无	String

2.6.2 用例

```
String msgToSend = new MessageBuilder()
        .add("event", "login")
        .add("username", "test")
        .add("password", "test)
        .build();                    //{"event":"login", "username":"test",
"password":"test"}
```

2.7 套接字处理模块

2.7.1 SocketUtils

函数名	功能说明	输入	输出	说明
readMessage	从socket中读消息	AsynchronousSocketChannel socket, CompletionHandler handler	无	handler为异步处理函数
sendMessage	写消息到socket中	AsynchronousSocketChannel socket, String message, CompletionHandler handler	无	handler为异步处理函数

2.7.2 用例

```
SocketUtils.readMessage(socketChannel, new GeneralHandler() {
        @Override
        public void completed(Integer result, Attachment attachment) {
                ByteBuffer byteBuffer = attachment.socketChannel;
                AsynchronousSocketChannel socketChannel = attachment.socketChannel;
                //Do something
        }

        @Override
        public void failed(Throwable throwable, Attachment attachment) {

        }
});
SocketUtils.sendMessage(socketWrapper, message, new GeneralHandler() {
        @Override
        public void completed(Integer result, Attachment attachment) {
                //Do something
        }

        @Override
        public void failed(Throwable throwable, Attachment attachment) {

        }
});
```

2.8 字符处理模块

2.8.1 StringUtils

函数名	功能说明	输入	输出
bufToString	将ByteBuffer转换为String	ByteBuffer buf	String
md5Hash	获取String的Hash值	String plainText	String

2.8.1 用例

```
ByteBuffer buf = new ByteBuffer.wrap(new String("Hello World"));
String message = ByteBuffer.bufToString(buf);    //message->"Hello World"
String md5 = ByteBuffer.md5Hash(message);    //md5->b10a8db164e0754105b7a99be72e3fe5
```