软件复用课程设计

client-server 应用程序

程序文档

小组： Team10

成员： 谢志杰 1352975

计鹏玥 1352914

王思尧 1352896

谢明玥 1352937

徐锦程 1353012

文档变更记录

|  |  |  |
| --- | --- | --- |
| 更改人 | 日期 | 更改内容 |
| 王思尧 | 2016年3月28日 | 创建文档 |
| 计鹏玥，谢明玥 | 2016年4月9日 | 完善文档 |
|  |  |  |
|  |  |  |

**1** 开发规划

**1.1** 开发人员

谢志杰 1352975

计鹏玥 1352914

王思尧 1352896

谢明玥 1352937

徐锦程 1353012

**1.2** 开发计划

|  |  |  |
| --- | --- | --- |
| 起始时间 | 结束时间 | 计划 |
| 2016年3月16日 | 2016年3月30日 | 实现计划初制定的全部需求，并完成各个阶段的测试。 |
| 2016年3月23日 | 2016年4月6日 | 继续完成应用程序的代码开发和文档撰写，并标识出所有的可复用构件。 |
| 2016年3月30日 | 2016年4月13日 | 完成配置管理（CM）、性能管理（PM）、许可证（License/Load Regulation）等可复用构件的开发，生成JAR包，并继续撰写文档。 |
| 2016年4月6日 | 2016年4月20日 | 讨论构件的定义和接口，完成其他的可复用构件的代码和文档编写，并将已经开发完善的构件存储为复用库，以及将选择的其他组的可复用构件标识到文档中。 |

**1.3** 开发环境和工具

GitHub － 版本管理工具

Java － 开发语言

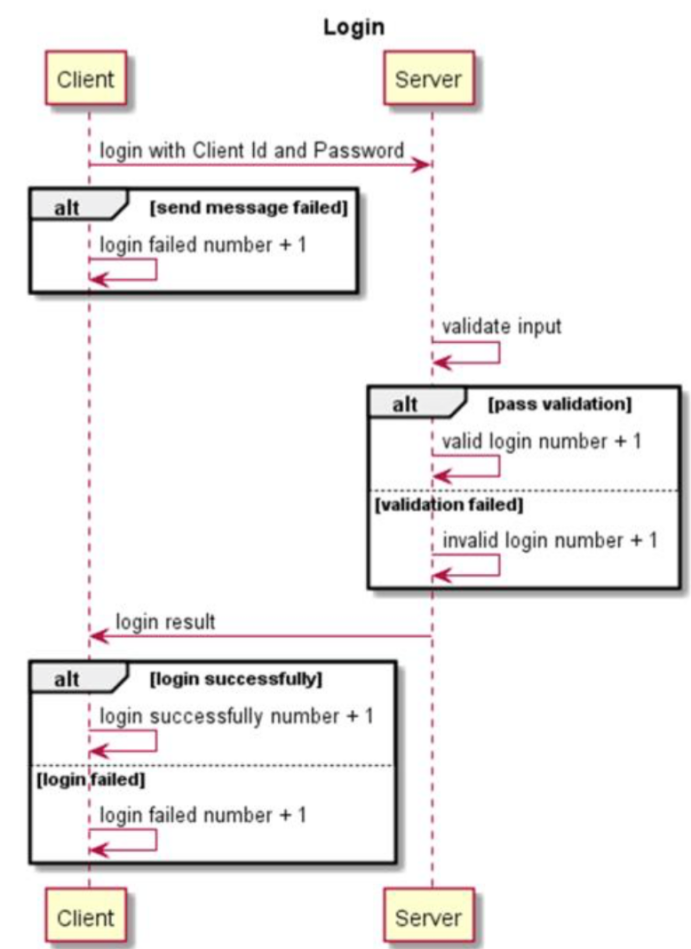
JUnit － 单元测试框架

RabbitMQ - 中间件

**2** 总体设计

**2.1** 基本设计要求

程序时序图如下所示：



基本要求：

1.客户端登录服务器：

1.1.需要提供：用户id，用户密码，可配置的服务器地址

1.2.基本要求：登录之后保持连接

记录以下数据，每隔一分钟写入文件：

1.客户端：登录成功、登录失败

2.服务端：有效登录、无效登录

2.登录成功后客户端向服务端发送消息：

2.1.服务端验证接受的消息并发出应答信息：

2.1.1.忽略客户端登录之前发送的消息

2.1.2.不允许客户端一秒内发送超过5条消息（该条件需可配置）

－超过的消息可以直接忽略无需做出应答

2.1.3.不允许客户端每次登录发送超过100条消息（该条件需可配置）

－在客户端发送第100条消息后服务端要给出重新登录的应答信息

－客户端发送100条消息后，在客户端重新登录前忽略全部消息

－与客户端保持连接

－重新登录后，上一次登录发送的超过100条的消息不计入本次计数

2.1.4.验证通过需要作出OK应答

2.2.客户端收到重新登录应答后需要自动重新登录

2.3.记录以下数据，每隔一分钟写入文件：

1.客户端：发送消息的数量

2.服务端：接受／忽略消息的数量

3.服务端通知其他登录的客户端接收到的消息：

3.1.服务端向其他登录状态的客户端转发收到的有效消息

3.2.客户端告知服务端已经收到消息

3.3.记录以下数据，每隔一分钟写入文件：

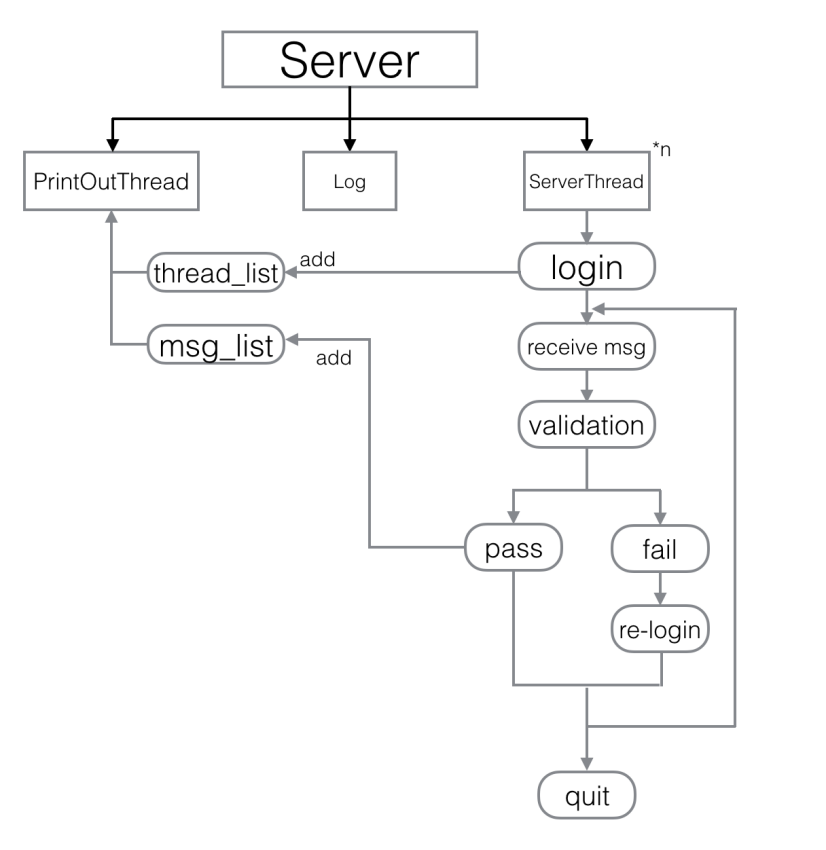
1.服务端：转发消息的数量，对于每一个客户端每一条消息＋1

2.客户端：收到的消息的数量

**2.2** 基本设计描述

考虑到上述提到的功能需求，该应用程序主要分成三个模块，即用户端、客户端以及工具类模块，客户端主要完成上述功能需求中有关客户端的部分，包括登录、发送消息、接受消息以及客户端日志记录等功能。服务端主要完成上述功能需求中有关服务端的部分，包括用户登录验证、消息确认及转发、消息应答以及服务端日志记录等功能。工具类包涵应用程序中需要用到的方法类，包括IO读写操作、消息生成与解析（JSON）等。

服务端基本设计如下所示：



**2.3** 主要界面流程描述

**//**待定

**2.4** 模块列表

－客户端

－服务端

－工具类

－**Authentication**

－CM(Configuration Management)

－DataSource

－**License**

－MessageUtils

－PM(Property Management)

**3** 接口规范

**3.1 Authentication**

**3.1.1. Authentication.java**

**(1)** getUsername()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| username | String | 用户的登录名 |

(2) getPassword()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| password | String | 用户的登录密码 |

(3) authenticate(DataSource dataSource, String username, String password)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| dataSource | DataSource | 数据源 |
| username | String | 用户名 |
| password | String | 密码 |

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| true/false | Boolean | 身份验证是否通过 |

**3.1.2. Verification.java**

**(1)** getUsername()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| username | String | 用户的登录名 |

(2) getPassword()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| password | String | 用户的登录密码 |

1. login(BufferedReader in, PrintWriter out, DataSource dataSource, Logger logger, String valid\_login\_per\_min, String invalid\_login\_per\_min, long threadId)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| dataSource | DataSource | 数据源 |
| logger | Logger | 计数与写日志 |
| valid\_login\_per\_min | String | 每分钟合法登录的次数 |
| invalid\_login\_per\_min | String | 每分钟非法登录的次数 |
| threadId | long | 线程标识符 |

1. csLogin(BufferedReader in, PrintWriter out, DataSource dataSource, Logger logger, String valid\_login\_per\_min, String invalid\_login\_per\_min)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| dataSource | DataSource | 数据源 |
| logger | Logger | 计数与写日志 |
| valid\_login\_per\_min | String | 每分钟合法登录的次数 |
| invalid\_login\_per\_min | String | 每分钟非法登录的次数 |

**3.2** CM(Configuration Management)

**3.2.1.** GetConfiguration.java

**(1)** GetConfiguration()

用于构造GetConfiguration对象，无参数与返回值。

1. ReadJSONFile(String Path)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| Path | String | 配置文件的路径 |

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| laststr | String | 配置文件中的内容 |

1. writeJSONFile(String path,String[] key,String[] value)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| path | String | 准备写入的文件所在的路径 |
| key | String[] | 键值 |
| value | String[] | 键值所对应的值 |

1. getStringByKey(String key)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| key | String | 键值 |

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| value | String | 键值所对应的值 |

1. getIntByKey(String key)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| key | String | 键值 |

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| value | Int | 键值所对应的值 |

1. load()

动态加载配置信息。

1. loadData()

每30秒动态加载一次配置信息。

(8) getSERVER\_IP()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| SERVER\_IP | String | 服务器的IP地址 |

(9) getSERVER\_PORT()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| SERVER\_PORT | String | 服务器的端口号 |

(10) getMAX\_MESSAGE\_PER\_SECOND()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| MAX\_MESSAGE\_PER\_SECOND | Int | 每分钟可收发的最大消息数 |

(11) getMAX\_MESSAGE\_FOR\_TOTAL()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| MAX\_MESSAGE\_FOR\_TOTAL | Int | 可进行收发的最大消息总数 |

(12) getDBUSER()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| DBUSER | String | 数据库名称 |

(12) getDBPW()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| DBPW | String | 数据库密码 |

**3.3** DataSource

**3.3.1. DataSource.java**

1. DataSource(BufferedReader in, PrintWriter out)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| in | BufferedReader | 缓存阅读器 |
| out | PrintWriter | 向文本输出流打印对象的格式化表示形式 |

1. DataSource(String dbuser, String dbpw)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| dbuser | String | 数据库用户名 |
| dbpw | String | 数据库密码 |

1. getType()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| sourceType | String | 当前对象的资源类型 |

1. getPasswordDB(String username)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| username | String | 想要查找的用户名 |

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| password | String | 输入的用户名对应的密码 |

1. getPasswordResponse(String msg)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| msg | String | 想要答复的信息内容 |

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| res | String | 答复的内容 |

**3.4** License

**3.4.1 License.java**

(1) License(int max\_msg\_in\_second, int max\_num\_message, int firstTime, int period)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| max\_msg\_in\_second | int | 每分钟收发消息的最大数 |
| max\_num\_message | int | 总收发消息的最大数 |
| firstTime | int | 计时器开启的时间 |
| period | int | 计时器的周期 |

(2) setTime(int firstTime, int period)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| firstTime | int | 计时器开启时间 |
| period | int | 计时器的周期 |

(3) commence()

用于开启消息计数的计时器。

(4) cancel()

关闭计时器。

1. setMax(int a, int b)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| a | int | 每分钟收发消息最大数 |
| b | int | 总收发消息最大数 |

(6) increaseMsg()

将消息数进行加1。

(7) reset()

将消息数重置为0。

(8) checkMsgInSecond()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| true/false | boolean | 判断每分钟获取的消息数是否超过了max\_msg\_in\_second |

(9) checkTotalMsg()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| true/false | boolean | 判断获取的总消息数是否超过了max\_num\_message |

**3.4.2 MessageCount.java**

(1) MessageCount()

将每秒钟收发的消息数和总收发的消息数都初始化为0。

(2) getMsgInSecond()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| msg\_in\_second | Int | 每分钟收发的消息数 |

(3) getTotalMsg()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| num\_message | Int | 总收发的消息数 |

1. increaseMsg()()

该方法用于将消息计数加1，包括每分钟消息计数与总消息计数。

1. reset()

该方法用于将消息计数归零，包括每分钟消息计数与总消息计数。

1. run()

该方法继承于TimerTask，可以用在定时器中，主要功能为将每分钟消息计数归零。

**3.5** MessageUtils

**3.5.1 Message.java**

(1) Message(String msg, long ownerThread)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| msg | String | 消息的内容 |
| ownerThread | long | 消息所属线程的标识符 |

(2) getOwner()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| ownerThread | long | 消息所属线程的标识符 |

(3) getValue(String key)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| key | String | 键值 |

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| value | String | 键值对应的值 |

(4) setValue(String key, String value)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| key | String | 键值 |
| value | String | 键值对应的值 |

**(5)** setValue(String key, long value)

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| key | String | 键值 |
| value | long | 键值对应的值 |

1. reset(String msg)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| msg | String | JSON对象的内容 |

1. clear()

实例化一个空的JSON字符串。

(8) init(String queue\_name, String hostName)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| queue\_name | String | 消息队列的名称 |
| hostName | String | 建立连接的主机名 |

(9) bindTo(String exchangeName, String routerKey)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| exchangeName | String | 交换机的名称 |
| routerKey | String | 路由键的名称 |

(10) getQueueName()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| queueName | String | 消息队列的名称 |

1. getChannel()

获取信道对象。

(11) publishToOne(String exchangeName, String routerKey)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| exchangeName | String | 交换机名称 |
| routerKey | String | 路由键 |

(12) publishToAll(String exchangeName)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| exchangeName | String | 交换机名称 |

1. publishToOthers(String exchangeName, String routerKey)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| exchangeName | String | 交换机名称 |
| routerKey | String | 路由键 |

1. terminate()

关闭信道和连接。

(15) toString()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
|  | String | JSON对象字符串内容 |

**3.5.2 MessageDeparturer.java**

(1) MessageConsumer(Message msg, PrintWriter out, Logger logger, String logKey)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| msg | String | 消息的内容 |
| out | PrintWriter | 输出流 |
| logger | Logger | 登录计数器 |
| logKey | String | 登录关键字 |

(2) sendMessage(String msg)

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| msg | String | 消息的内容 |

(3) handleDelivery(String consumerTag, Envelope envelope, AMQP.BasicProperties properties, byte[] body)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| consumerTag | String | 消费者标签 |
| envelope | Envelope | 信封 |
| properties | AMQP.BasicProperties | AMQP基本属性 |
| body | byte[] | 内容 |

(4) MessageDeparturer(Message msg, PrintWriter out, Logger logger, String logKey)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| msg | Message | 消息内容 |
| out | PrintWriter | 输出流 |
| logger | Logger | 登录计数器 |
| logKey | String | 登录关键字 |

**3.6 P**M(Property Management)

**3.6.1** CheckCount.java

1. CheckCount(String logFilename)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| logFilename | String | 写入的日志的地址 |

(2) addCountType(String msg)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| msg | String | 计数字段 |

(3) addCount(String msg)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| msg | String | 需要增加的计数字段 |

(4) getLock(String msg)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| msg | String | 获取对象锁的计数字段 |

(5) run()

该方法用于将全部的计数字段以及各个计数字段所对应的计数结果存入在构造函数中指定的日志文件中，该方法实质为TimerTask中的run方法，可以用于定时器操作。

**3.6.2** IOLog.java

(1) IOLog(String logAddr, boolean bAppend)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| logAddr | String | 写文件地址 |
| bAppend | boolean | 是否在文件为写入 |

(2) IOWrite(String sWriten)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| sWriten | String | 要写入的字符串 |

**3.6.3** Logger.java

(1) Logger(String logFilename)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| logFilename | String | 将要创建日志的文件名 |

(2) setTime(int firstTime, int period)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| firstTime | int | 计时器的开始时间 |
| period | int | 计时器的周期 |

1. commence()

该方法用于开启登录计数的计时器。

1. terminate()

关闭计时器。

(5) addCountType(String msg)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| msg | String | 要计数的消息 |

(6) addCount(String msg)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| msg | String | 消息的内容 |

(7) getLock(String msg)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| msg | String | 要计数的消息 |