

# 廈門大學



## 软件学院

### 《中间件技术》实验报告

题 目 中间件技术实验二

姓 名 陈澄

学 号 32420212202930

班 级 软工三班

实验时间 2024/03/14

2024 年 03 月 14 日

## 1 实验目的

- 1) 了解消息型中间件的基本概念
- 2) 理解存储转发的概念。
- 3) 理解消息 queue 和 topic 的概念。

## 2 实验环境

操作系统：Windows11

编译环境：IntelliJ IDEA

平台技术：JMS（JavaEE 开发平台）

## 3 实验题目

选择 1：实现存储转发功能。可以基于 socket，也可以基于 jms。

选择 2：模拟实现消息 topic 的一对多发送和接收。

选择 3：实现文件的（语音或视频更好）的收发和转发。

## 4 代码展示

实现思路已经体现在注释中

```
m pom.xml (Project3)  Main.java x
6 import javax.jms.*;
7 import java.io.BufferedReader;
8 import java.io.File;
9 import java.io.IOException;
10 import java.io.InputStreamReader;
11
12 public class Main {
13     //ActiveMq服务所在的地址和端口
14     //1个用法
15     private static final String BROKER_URL = "tcp://localhost:61616";
16     //用户订阅的TOPIC, 只有相同的TOPIC才能收到相同的消息(即群聊)
17     //1个用法
18     private static final String CHAT_TOPIC = "chatTopic";
19     //用户的用户名, 用于唯一标识一个用户, 后续也会作为ClientID使用
20     //3个用法
21     private static final String USER_NAME = "user1";
22
23     public static void main(String[] args) {
24         try {
25             //与ActiveMq建立连接
26             ConnectionFactory connectionFactory = new ActiveMQConnectionFactory(BROKER_URL);
27             Connection connection = connectionFactory.createConnection();
28             connection.setClientID(USER_NAME);
29             connection.start();
30
31             //设定消息确认模式(此处是自动确认)和TOPIC
32             Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
33             Topic topic = session.createTopic(CHAT_TOPIC);
34
35             //创建消息监听事件, 便于收到其他用户消息
36             MessageConsumer consumer = session.createDurableSubscriber(topic, USER_NAME);
37             consumer.setMessageListener(message -> {
38                 if (message instanceof TextMessage) {
39                     //如果是TextMessage即为普通聊天消息
40                     try {
41                         String receivedMessage = ((TextMessage) message).getText();
42                         System.out.println("Received: " + receivedMessage);
43                     } catch (JMSException e) {
44                         e.printStackTrace();
45                     }
46                 } else if (message instanceof BytesMessage) {
47                     //如果是BytesMessage即为文本消息
48                     try {
49                         BytesMessage bytesMessage = (BytesMessage) message;
50                         byte[] fileBytes = new byte[(int) bytesMessage.getBodyLength()];
51                         bytesMessage.readBytes(fileBytes);
52
53                         //指定保存文件的路径, 此处以文本文件为例
54                         String fileName = message.getStringProperty("fileName");
55                         String filePath = "C:\\Users\\CC507\\Desktop\\output\\" + fileName;
56                         FileUtils.writeByteArrayToFile(new File(filePath), fileBytes);
57
58                         System.out.println("File received and saved: " + filePath);
59                     } catch (JMSException | IOException e) {
60                         e.printStackTrace();
61                     }
62                 }
63             });
64
65             //创建了MessageProducer对象, 用于发送消息到主题
66         } catch (Exception e) {
67             e.printStackTrace();
68         }
69     }
70 }
```

```
m pom.xml (Project3)  Main.java x
61
62 //创建了MessageProducer对象，用于发送消息到主题
63 MessageProducer producer = session.createProducer(topic);
64
65 // 简单的命令行输入输出
66 BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
67 while (true) {
68     String messageText = reader.readLine();
69
70     if (messageText.startsWith("/file")) {
71         //如果以/file开头即为发送文件，后跟文件路径即可
72         String filePath = messageText.substring(6); // 提取文件路径
73
74         File file = new File(filePath);
75         String fileName = file.getName();
76         if (!file.exists() || !file.isFile()) {
77             System.out.println("Invalid file path!");
78             continue; // 如果文件不存在或者不是一个普通文件，则继续下一次循环
79         }
80
81         byte[] fileBytes = FileUtils.readFileToByteArray(file); // 读取文件内容
82
83         BytesMessage bytesMessage = session.createBytesMessage();
84         bytesMessage.setStringProperty("s: fileName", fileName);
85         bytesMessage.writeBytes(fileBytes);
86         producer.send(bytesMessage); // 发送字节消息
87     } else {
88         //发送普通消息
89         TextMessage textMessage = session.createTextMessage("s: USER_NAME + ": " + messageText);
90         producer.send(textMessage); // 发送文本消息
91     }
92 }
93
94 } catch (JMSException | IOException e) {
95     e.printStackTrace();
96 }
97
98 }
99
```

相关依赖

```

<dependencies>
  <!-- ActiveMQ Core Library -->
  <dependency>
    <groupId>org.apache.activemq</groupId>
    <artifactId>activemq-client</artifactId>
    <version>5.15.9</version> <!-- 替换为您需要的版本 -->
  </dependency>

  <!--<dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-classic</artifactId>
    <version>1.2.3</version>
  </dependency>-->

  <dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.11.0</version>
  </dependency>

</dependencies>

```

## 5 实验结果

### 选择 1：存储转发

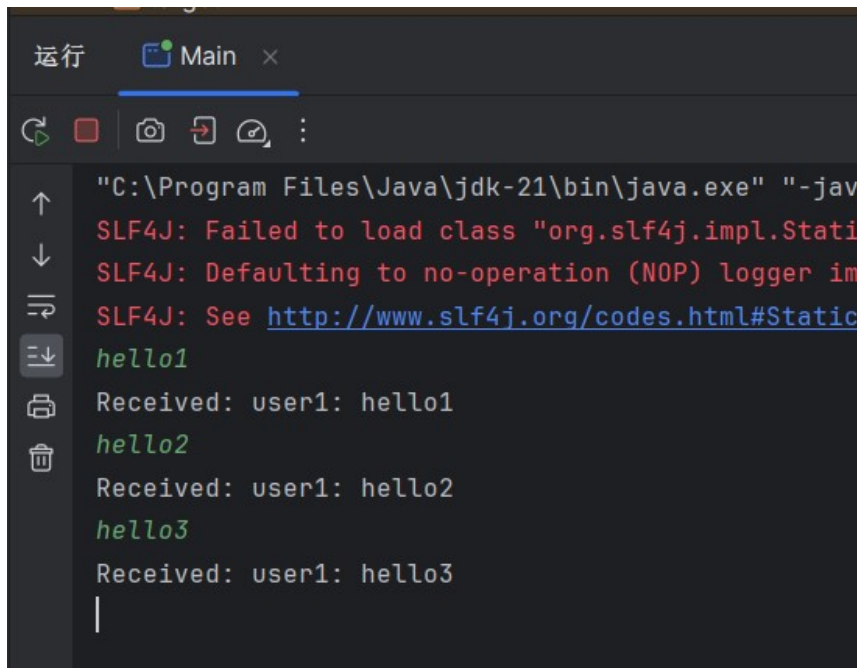
创建消息消费者对象时，将其初始化为持久订阅者，消息会被持久化存储并进行转发，即使消费者不在线也能接收到消息

```

//创建消息监听事件，便于收到其他用户消息
MessageConsumer consumer = session.createDurableSubscriber(topic, USER_NAME);

```

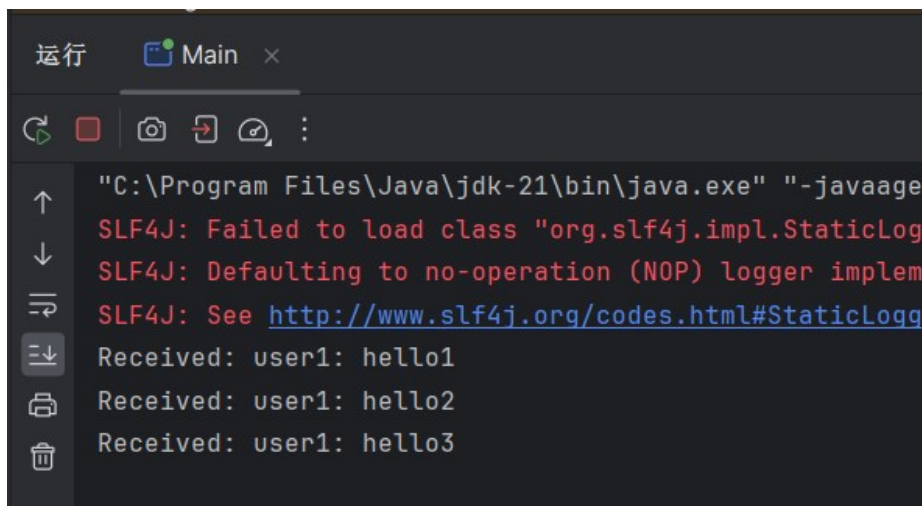
使用 user1 发送三条消息



```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-jav
SLF4J: Failed to load class "org.slf4j.impl.Stati
SLF4J: Defaulting to no-operation (NOP) logger im
SLF4J: See http://www.slf4j.org/codes.html#Static
hello1
Received: user1: hello1
hello2
Received: user1: hello2
hello3
Received: user1: hello3
|
```

关闭程序切换为 user2 并重新运行

收到了上述三条消息，说明消息在没有被消费的时候一直被持久化存储



```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaage
SLF4J: Failed to load class "org.slf4j.impl.StaticLog
SLF4J: Defaulting to no-operation (NOP) logger implem
SLF4J: See http://www.slf4j.org/codes.html#StaticLogg
Received: user1: hello1
Received: user1: hello2
Received: user1: hello3
```

## 选择 2：一对多收发

所有订阅了相同 TOPIC 的用户都将收到相同的消息

发送方：

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:D:\JetBrains\IntelliJ IDEA 2023
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
123456hello
Received: user1: 123456hello
```

接收方：（这里只以一个用户为例，实际上多个接受方收到的也相同）

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:D:\JetBrains\IntelliJ IDEA 2023.2.5\lib\
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Received: user1: 123456hello
|
```

### 选择 3：文件收发

发送文件时附带一个 fileName 字段存储文件名称，用于根据不同的文件类型解析文件

```
while (true) {
    String messageText = reader.readLine();

    if (messageText.startsWith("/file")) {
        //如果以/file开头即为发送文件，后跟文件路径即可
        String filePath = messageText.substring(beginIndex: 6); // 提取文件路径

        File file = new File(filePath);
        String fileName = file.getName();
        if (!file.exists() || !file.isFile()) {
            System.out.println("Invalid file path!");
            continue; // 如果文件不存在或者不是一个普通文件，则继续下一次循环
        }

        byte[] fileBytes = FileUtils.readFileToByteArray(file); // 读取文件内容

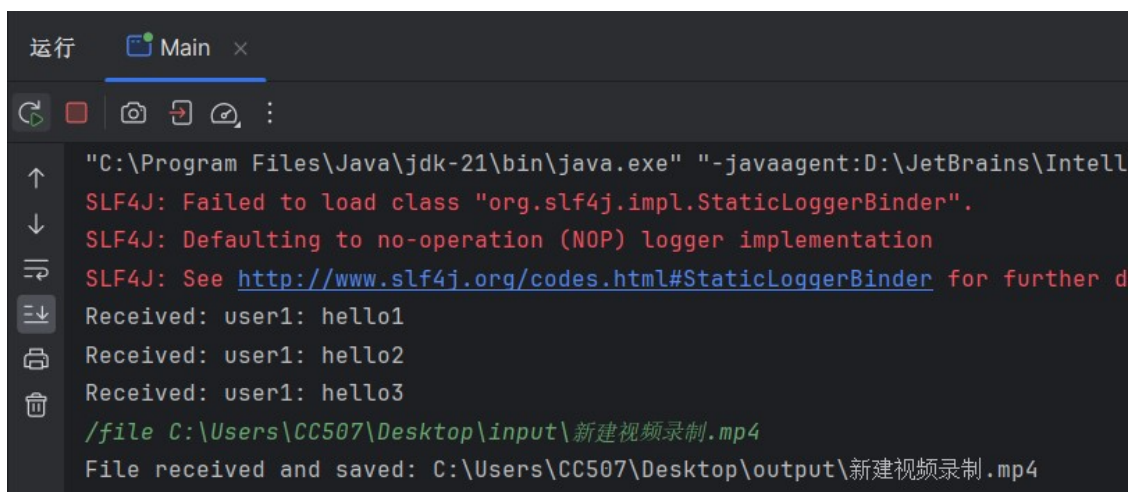
        BytesMessage bytesMessage = session.createBytesMessage();
        bytesMessage.setStringProperty("s: fileName", fileName);
        bytesMessage.writeBytes(fileBytes);
        producer.send(bytesMessage); // 发送字节消息
    }
}
```



同样接收文件时也需要解析该字段并存储

```
} else if (message instanceof BytesMessage) {  
    //如果是BytesMessage即为文本消息  
    try {  
        BytesMessage bytesMessage = (BytesMessage) message;  
        byte[] fileBytes = new byte[(int) bytesMessage.getBodyLength()];  
        bytesMessage.readBytes(fileBytes);  
  
        // 指定保存文件的路径, 此处以文本文件为例  
        String fileName = message.getStringProperty("s: fileName");  
        String filePath = "C:\\Users\\CC507\\Desktop\\output\\" + fileName;  
        FileUtils.writeByteArrayToFile(new File(filePath), fileBytes);  
  
        System.out.println("File received and saved: " + filePath);  
    } catch (JMSException | IOException e) {  
        e.printStackTrace();  
    }  
}
```

此处以发送视频文件为例



```
运行 Main x  
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:D:\JetBrains\IntelliJ  
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".  
SLF4J: Defaulting to no-operation (NOP) logger implementation  
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further d  
Received: user1: hello1  
Received: user1: hello2  
Received: user1: hello3  
/file C:\Users\CC507\Desktop\input\新建视频录制.mp4  
File received and saved: C:\Users\CC507\Desktop\output\新建视频录制.mp4
```

Output 文件夹中有正确的.mp4 文件，发送成功



