本次实验中，我们实现了如下功能：

①优化代码，实现了RabbitMQ的消息可靠性

②实现了实时翻译的功能

1.RabbitMQ的消息可靠性：

对于生产者来说：

为了保证消息被正确投递到消息中间件，RabbitMQ提供了如下两个配置来保证消息投递的可靠性:

1.在发送消息的时候我们可以设置Mandatory属性。如果设置了Mandatory属性则当消息不能被正确路由到队列中去时将会触发Return Method，这样我们可以在Return Method中进行相关业务处理，如果Mandatory没有设置则当消息不能正确路由到队列中去的时候，Broker将会丢弃该消息。

2.RabbitMQ还提供了消息确认机制(Publisher Confirm)。生产者将Channel设置成Confirm模式，当设置Confirm模式后所有在该信道上面发布的消息都会被指派一个唯一的ID(从1开始，ID在同个Channel范围是唯一的)，一旦消息被投递到所有匹配的队列之后Broker就会发送一个确认给生产者(包含消息的唯一ID)，这就使得生产者知道消息已经正确到达目的队列了

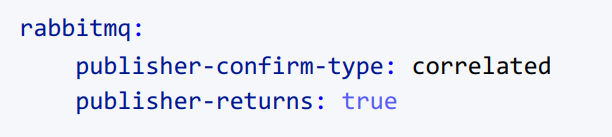
对于消费者来说：

RabbitMQ提供了Consumer Acknowledgements机制，使用Consumer Acknowledgements能在消费者消费消息后给Broker进行反馈，Broker根据反馈对消息进行处理。

除此之外，为了保证在服务器断开后，RabbitMQ中的消息仍然可以被保存，因此我们需要设置持久化。对于Exchange和Channel，我们需要指定durable属性为true，对于信息，我们需要制定delivery\_mode=2，即投递信息的模式为disc，这种情况下我们可以从AMQP服务器中恢复消息（需要牺牲部分的效率）。

代码呈现如下：

①



Publisher-confirm-type一共有三个值：NONE,CORRELATED,SIMPLE。

在选择为correlated的情况下，生产者发布消息成功到exchange后会触发回调方法。

在选择为simple的情况下，发布消息成功后使用rabbitTemplate调用waitForConfirms或waitForConfirmsOrDie方法等待broker节点返回发送结果，根据返回结果来判定下一步的逻辑，要注意的点是waitForConfirmsOrDie方法如果返回false则会关闭channel，则接下来无法发送消息到broker;在这种情况下消息具有阻塞性，然而生产者在聊天室中消息广播，不需要使用该方法，因此不使用simple字段。

Publisher-returns设置消息没有被路由到指定的queue时将消息返回，而不是丢弃，在这种情况下，Channel中才会添加监听回调函数。

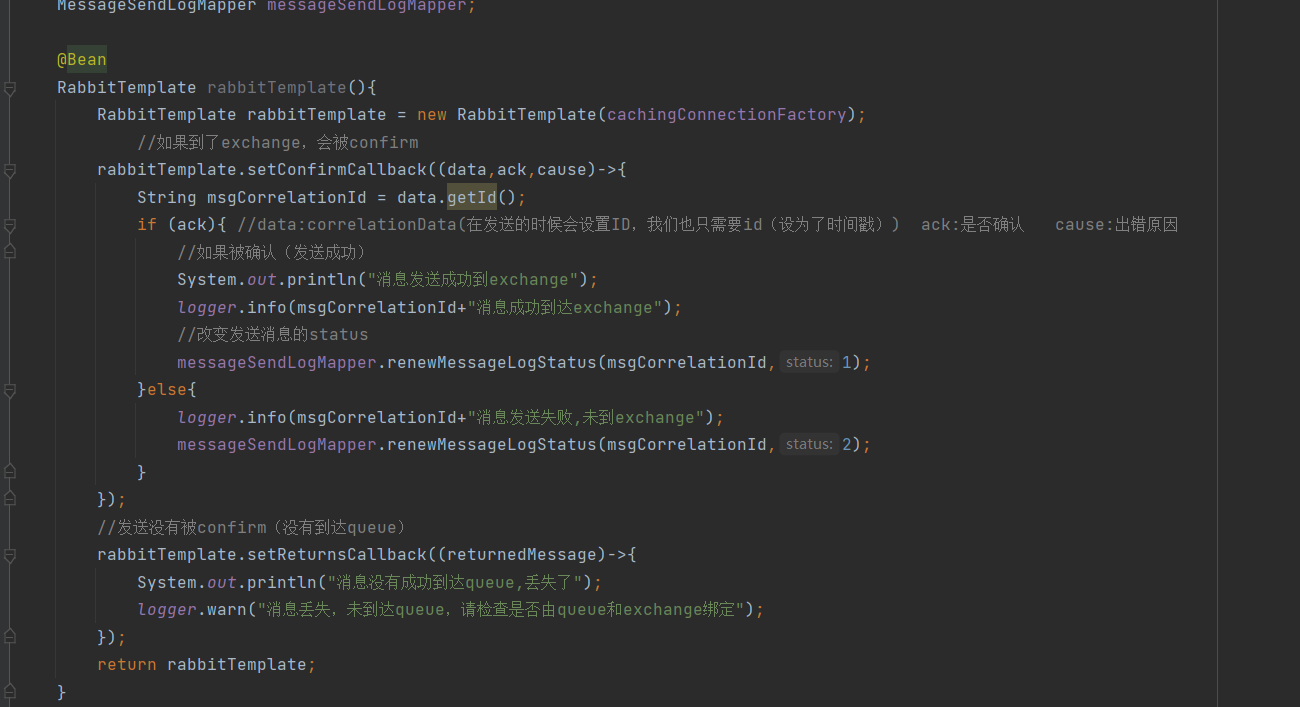
②每次发送都会创建一个MessageSendLog表，表的格式如下：



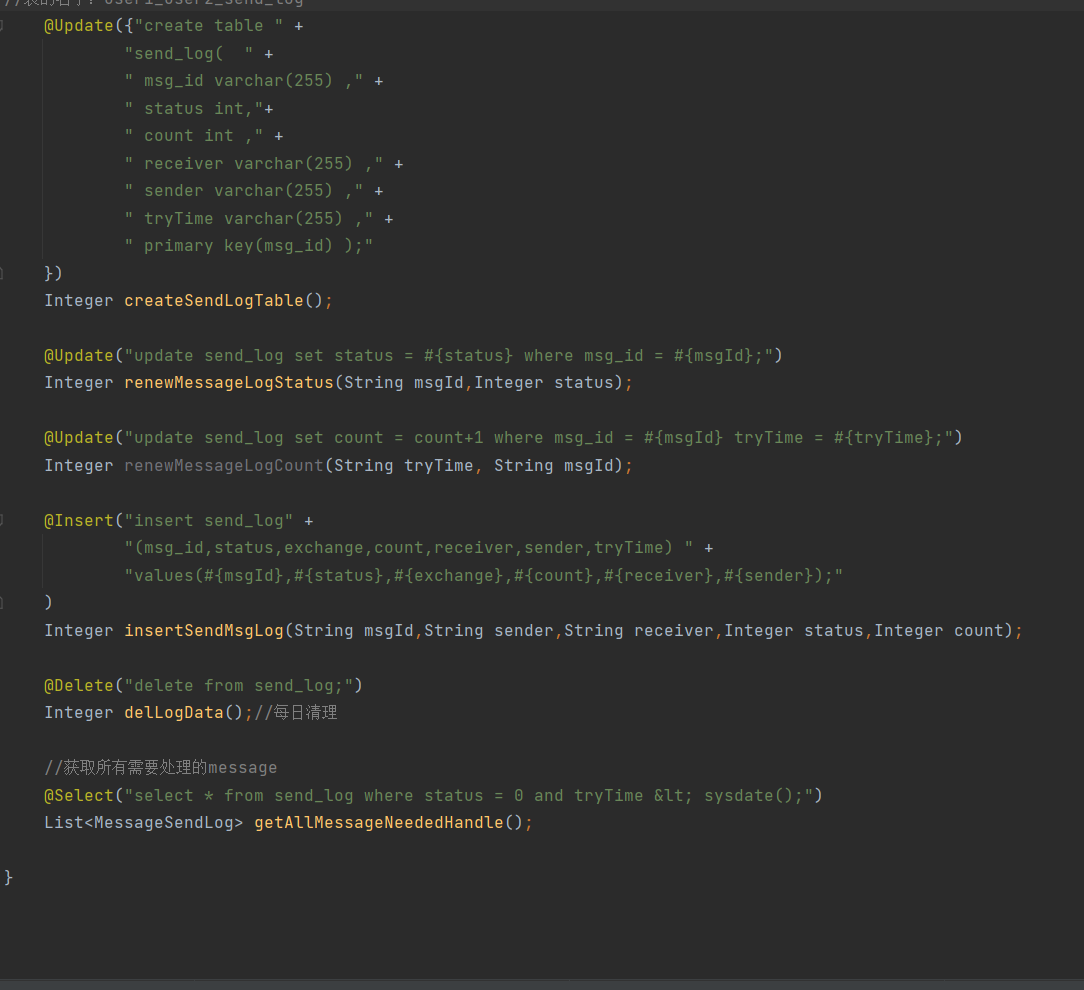
③



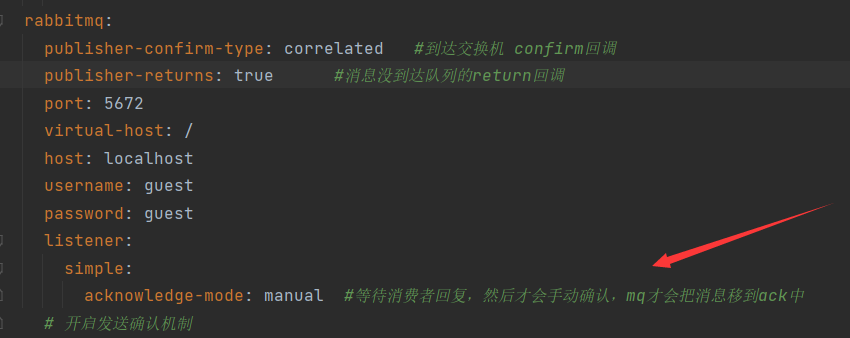
在收到客户端消息后，使用CorrelationData结构发送消息，确保消息的可靠性。CorrelationData对象内部只有一个 id 属性，用来表示当前消息唯一性。在RabbitConfig的消息回调函数中，设置CorrelationData的id值为消息的id值。



④MessageSendLog表的生成



⑤确保消息被消费



在配置文件中设置acknowledge-mode为manual。

在acknowledge-mode为默认值None的情况下，消息丢失的风险较大：

当BlockingQueue<Runnable>堆满时（BlockingQueue<Delivery>一定会先满），server端推送消息会失败，然后断开connection。消费端从Socket读取Frame将会抛出SocketException，触发异常处理，shutdown掉connection和所有的channel，channel shutdown后WorkPool中的channel信息（包括channel inProgress,channel ready以及Map）全部清空，所以BlockingQueue<Runnable>中的数据会全部丢失。

在rabbitmq server，connection断掉后就没有消费者去消费这个queue，因此在server端会看到消息堆积的现象。

设置acknowledge-mode的值为MANUAL后，server端会暂存没有收到ack的消息，等消费端ack后才会丢掉；如果收到消费端的nack（消费失败的标识）或connection断开没收到反馈，会将消息放回到原队列头部。在这种情况下，消息不会丢失。

⑥



使用basicAck、asicNack传入Channel参数。

basicAck(long deliveryTag, boolean multiple)

deliveryTag为消息的id，multiple代表是否批量处理。选择false表示本次只处理当前消息，从而确保消息安全性。

basicNack(long deliveryTag, boolean multiple, boolean requeue)

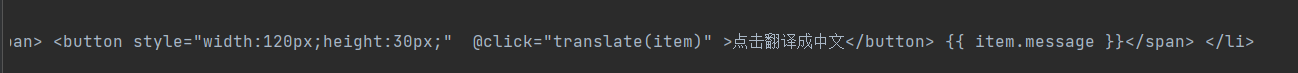
requeue表示被拒绝的是否重新加入队列，设置为true表示拒绝后重新添加在队列末尾而不是加入死信队列。

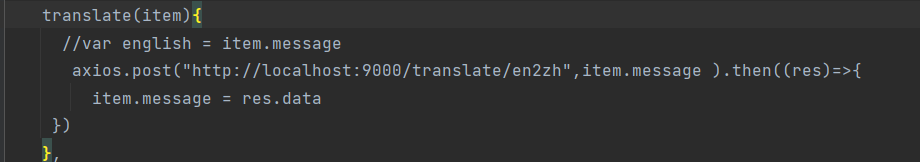
2.中间件实现实时翻译

我们通过调用百度翻译的API来实现实时翻译。

API使用文档：http://api.fanyi.baidu.com/doc/21

①实现前后端的交互逻辑

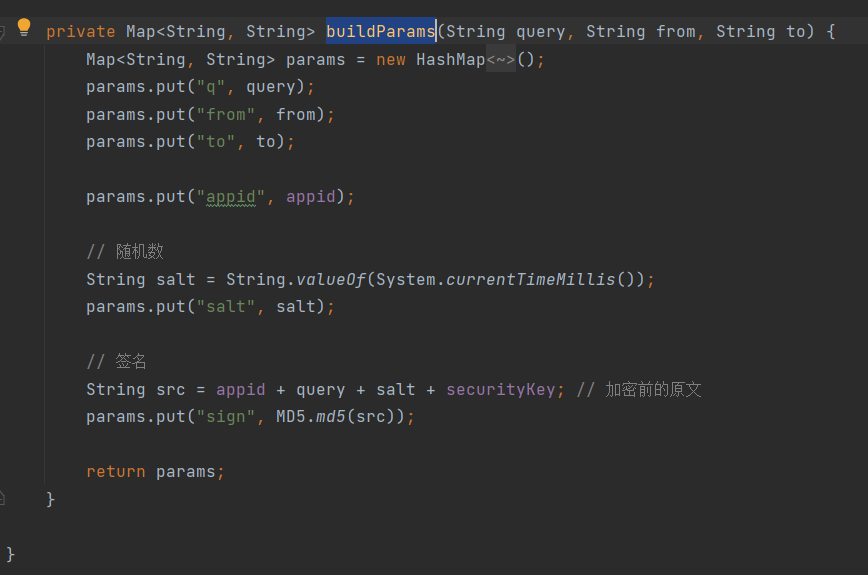




②编写翻译API

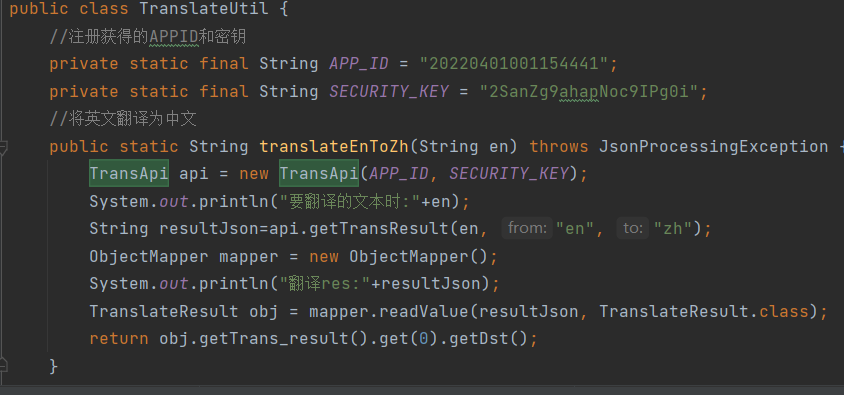
对应官方文档的格式，输入param并生成签名





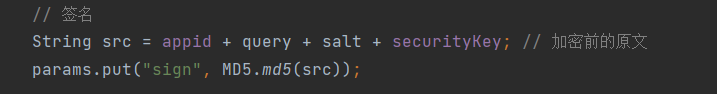
③实现API调用

在Util层调用翻译API

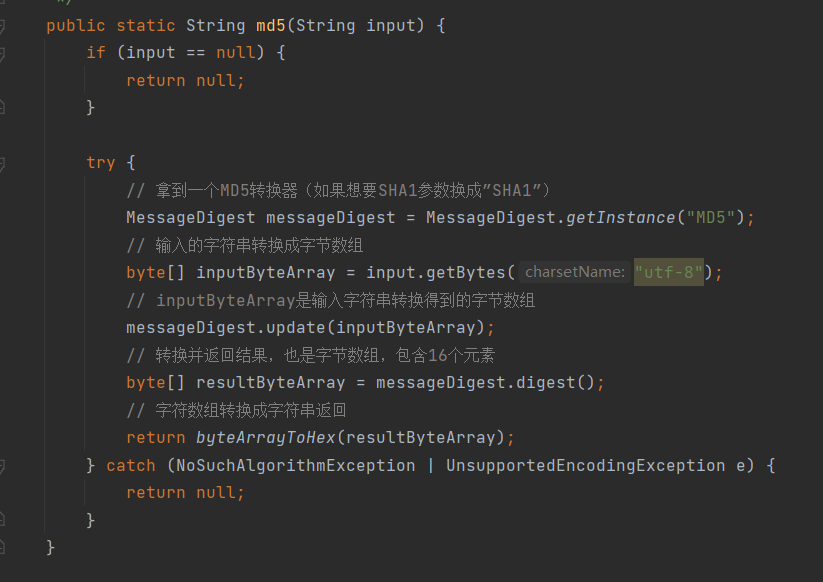


④保证消息安全性

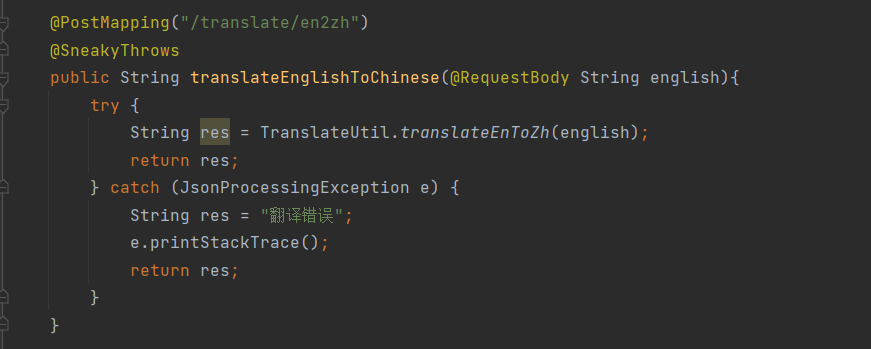
调用翻译API时，为保证翻译过程中消息的安全性，使用md5对签名进行加密：



编写专门的MD5类：



⑤Controller层进行接口调用



并添加了翻译错误的处理逻辑。

⑥使用翻译的界面呈现：

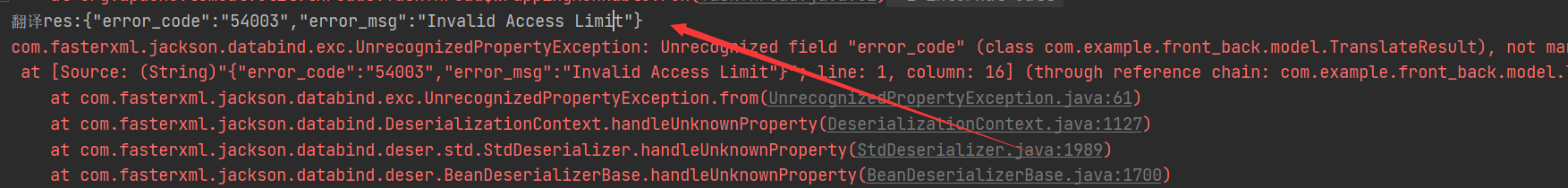
登录聊天室：



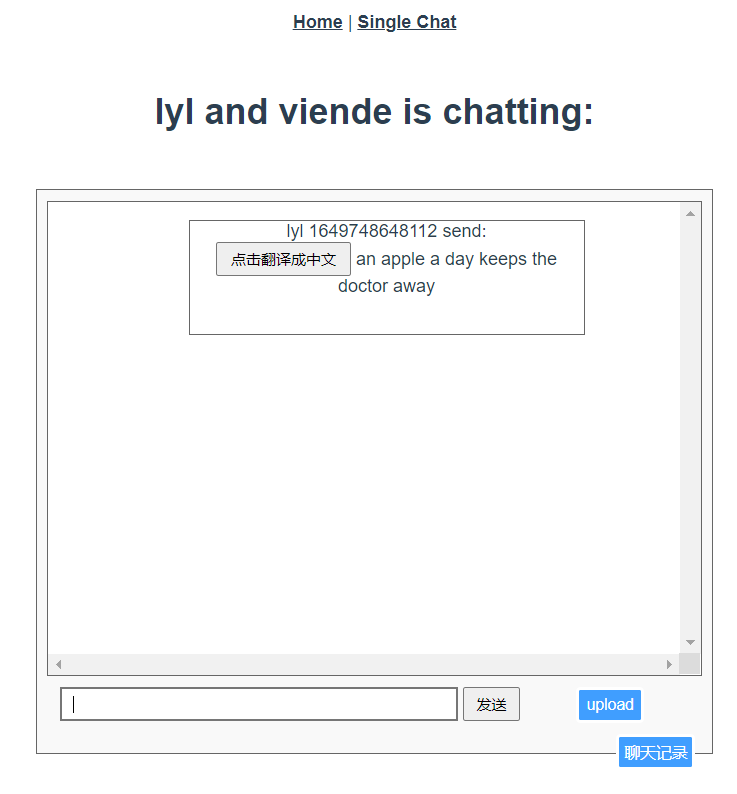
点击翻译按钮，输入的文字被翻译成中文。



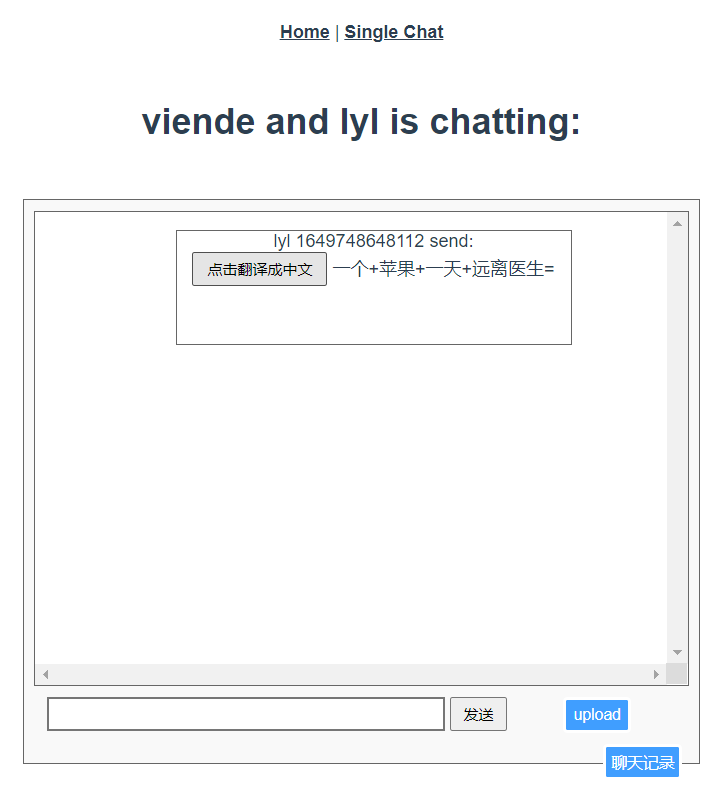
如果出现错误，会弹出错误提示：



点击人员，选择一位用户进行私聊：



点击翻译按钮：



查看数据库，此时该消息成功存入数据库，status code=1，表示已经成功收到。

