信息服务系统MQ配置

及规范说明书

|  |  |
| --- | --- |
| 当前版本: | V1.0 |
| 更新日期: | 2017/10/27 |
| 作者: |  |
| 创建日期: | 2017/10/27 |
| 审批人: |  |
| 审批日期: |  |

版本变更

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 更新日期 | 作者 | 变更描述 |
| V1 | 2017/10/27 | 许卫平 | 创建文档 |
|  |  |  |  |

审核历史

|  |  |  |
| --- | --- | --- |
| 审核日期 | 业务审查 | 质量审查 |
|  |  |  |

**目 录**

[1 信息服务系统中MQ配置说明 4](#_Toc496626988)

[2 消息体（格式） 1](#_Toc496626988)2

[3 配置变更（新增Q）时参考说明 14](#_Toc496626989)



1. **信息服务系统中MQ配置说明**
   1. ISM项目中对应MQ的配置（相当于信息服务系统的MQ的生产者）

在config/spring-common.xml中，配置MQ的说明如下

1. 首先配置MQ的连接工厂，如下图



图中brokerURL指定的是MQ服务器的ip和端口号，如不在同一台服务器需要指定ip。

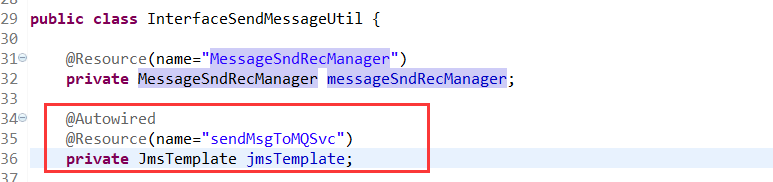
1. 配置生产者，如下图



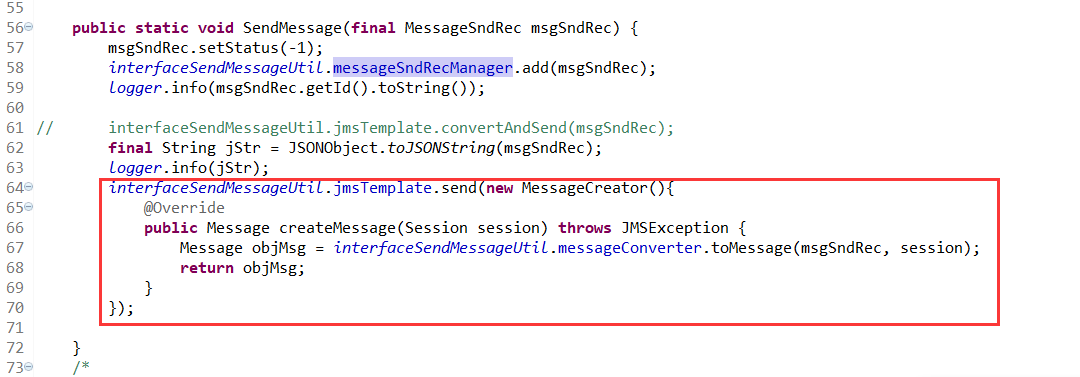
msgToMQSvcDestination中定义的唯一标识符“Message.ISM.To.MQSvc”指的是MQ中Q的名称，用于标识Q，接收端用此名称可接收Q上的消息；

sendMsgToMQSvc为spring中配置的JmsTemplate，配置好后下面3）中用到这个来发送具体消息；

1. 具体发送消息模块



在程序中配置上述JmsTemplate，用于发送消息；



发送消息代码如上，把消息体封装成JMS定义的Message，调用toMessage方法即完成消息发送到MQ中。

注：消息体可以是任意格式，信息服务系统中的格式为MessageSndRec，第二节中有此格式的说明。

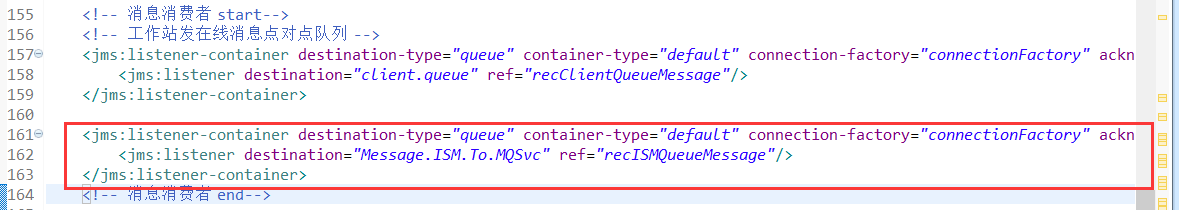
* 1. ISMMQSvc项目中对应MQ的配置（相当于信息服务系统的MQ的本身）

1. 配置MQ连接工厂



此配置虽如上面连接工厂的配置一样，需说明一下，因本ISMMQSvc一般都和MQ服务器在同一台机器上配置，所以brokerURL中ip一般配置localhost，相当于其他两个项目都来连接本MQ服务器。

1. 接收ISM端发来的消息的配置（即如上1.1的3发出的消息）



接收端==消费者，配置消费者监听器来接收MQ上的消息，destination为"Message.ISM.To.MQSvc"，就是指接收此MQ名字为标识符上的队列消息，也就是1.1的3）中的消息；

监听到消息了要由具体监听程序来处理，代码如下



此配置说明这个文件为监听程序，将用于处理接收消息的逻辑；具体逻辑处理如下说明

1. 根据业务逻辑（如钉钉、企信通等），再次发送到不同的Q中如何配置

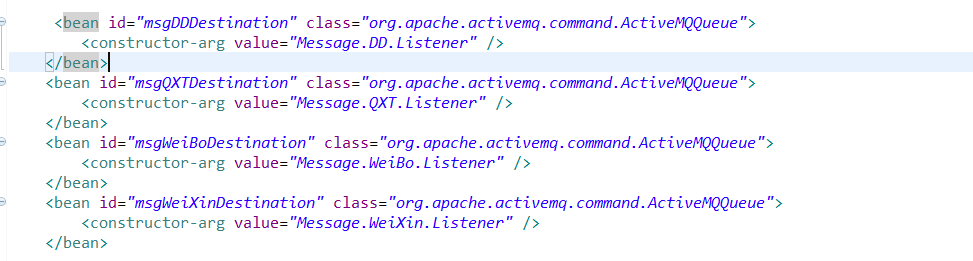
如有消息被监听到，将触发如下方法



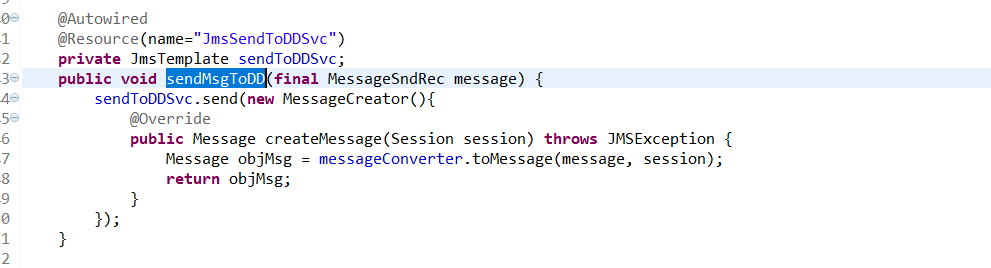
接收到消息时触发onMessage方法，此方法中首先把序列化的消息体转为具体消息体格式文件，然后根据逻辑判断再次把发送一条消息到各个消息通道Q中（钉钉、企信通等）。

1. 各个消息通道的消息发送

这里的消息发送如ISM中消息发送类似。先配置生产者，再在程序中把具体消息发送出去；







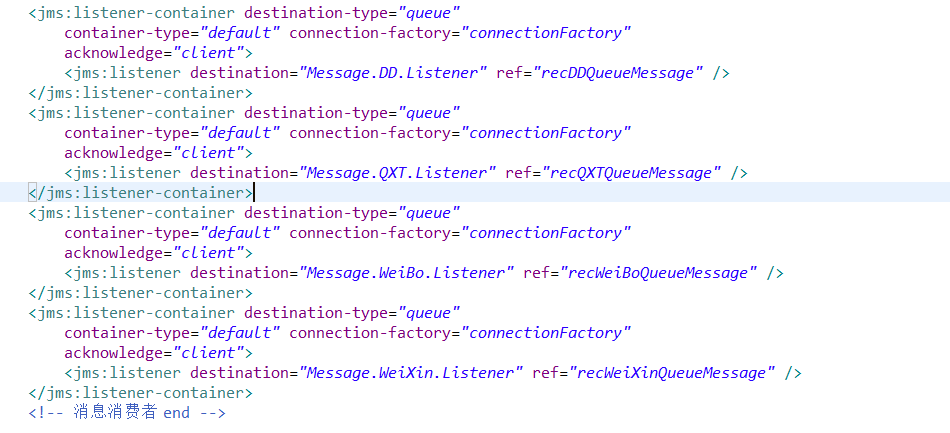
* 1. ISMMsgSvc项目中对应MQ的配置（相当于信息服务系统的MQ的消费者）

此处的消息接收也如上的消息接收

1. 配置连接工厂



1. 配置监听器







1. **消息体（格式）**

信息服务系统中的MQ的消息体为一张表的实体类，参照表格式如下

CREATE TABLE `message\_snd\_rec` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`destination` varchar(128) NOT NULL COMMENT '主题（01岗位、02钉钉、03企信通、04终端）',

`business` varchar(128) NOT NULL COMMENT '业务名（信息来源）01:应急信息,02:应急指引,03:行车交路,04:其他',

`send\_from` varchar(128) NOT NULL COMMENT '发送方（当前用户名/岗位）',

`send\_to` varchar(128) NOT NULL COMMENT '接收方（接收岗位、钉钉群组、企信通群组、终端用户名/岗位），每个不同类型对应一条记录',

`guide\_module\_id` int(11) NOT NULL COMMENT '指引消息模板code',

`message` text NOT NULL COMMENT '消息内容（文本类型消息）',

`message2` varchar(255) DEFAULT NULL COMMENT '消息内容（图片、声音等类型消息）',

`status` int(11) NOT NULL COMMENT '发送状态，0：排队中（MQ）；1：发送中（钉钉/企信通）；2：发送成功；9：发送失败',

`msg\_id` varchar(128) DEFAULT NULL COMMENT '消息id',

`read\_status` int(11) NOT NULL COMMENT '读取状态，0：未读；1：已读',

`send\_time` datetime DEFAULT NULL COMMENT '发送时间',

`receive\_time` datetime DEFAULT NULL COMMENT '接收时间',

`event\_id` varchar(128) DEFAULT NULL COMMENT '事件code',

`attr1` varchar(128) DEFAULT NULL COMMENT '存放事件的id',

`attr2` int(11) DEFAULT NULL COMMENT '备用字段2',

`attr3` int(11) DEFAULT NULL COMMENT '备用字段3',

`attr4` varchar(255) DEFAULT NULL COMMENT '备用字段4',

`attr5` varchar(255) DEFAULT NULL COMMENT '备用字段5',

`update\_time` datetime DEFAULT NULL COMMENT '更新时间',

`create\_time` datetime DEFAULT NULL COMMENT '创建时间',

`media\_id` varchar(128) DEFAULT NULL,

`accesstoken` varchar(255) DEFAULT NULL,

`sendtoname` varchar(400) DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `id` (`id`),

KEY `eventId` (`event\_id`),

KEY `send\_from` (`send\_from`) USING BTREE

) ENGINE=InnoDB AUTO\_INCREMENT=4593 DEFAULT CHARSET=utf8;

1. **配置变更（新增Q）时参考说明**
2. 新增消费者（从现在信息服务系统中新增一个监听器接收消息）

需要修改如下几处：

ISMMQSvc中spring-common.xml中增加一个JmsTemplate配置

<bean id="msg**XX**Destination" class="org.apache.activemq.command.ActiveMQQueue">

<constructor-arg value="Message.**XX**.Listener" />

</bean>

<bean id="JmsSendTo**XX**Svc" class="org.springframework.jms.core.JmsTemplate" >

<constructor-arg ref="connectionFactory" />

<property name="defaultDestination" ref="msg**XX**Destination"/>

<property name="deliveryMode" value="2" />

<property name="pubSubDomain" value="false" />

<property name="messageConverter" ref="objMessageConverter"/>

</bean>

com.sg.ism.mq包的RecISMQueueMessage.java中，在onMessage方法里用上述JmsTemplate把需要消息发送到出去；

然后在自己的项目中配置MQ的连接工厂，再配置一个监听器

<jms:listener-container destination-type="queue"

container-type="default" connection-factory="connectionFactory"

acknowledge="client">

<jms:listener destination="Message.**XX**.Listener" ref="rec**XX**QueueMessage" />

</jms:listener-container>

再建一个如下的java文件，即可在程序中接收到来自信息服务系统中发出的消息；

@Component(value = "rec**XX**QueueMessage")

public class Rec**XX**QueueMessage implements MessageListener {

@Override

public void onMessage(Message message) {

try {

if (message instanceof ObjectMessage) {

MessageSndRec msgSndRec = (MessageSndRec) messageConverter

.fromMessage(message);

// 这里写自己的逻辑处理 TODO

}

} catch (JMSException e) {

log.error("receiveMessage" + e.getMessage());

}

}

}