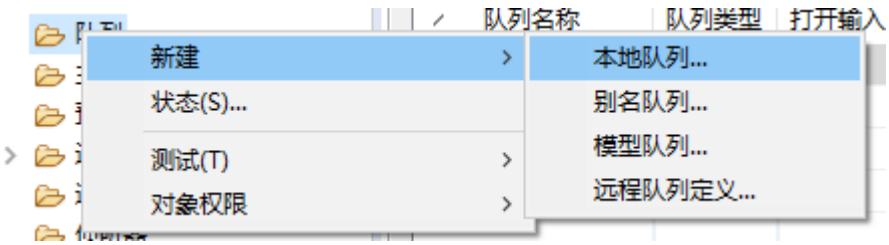
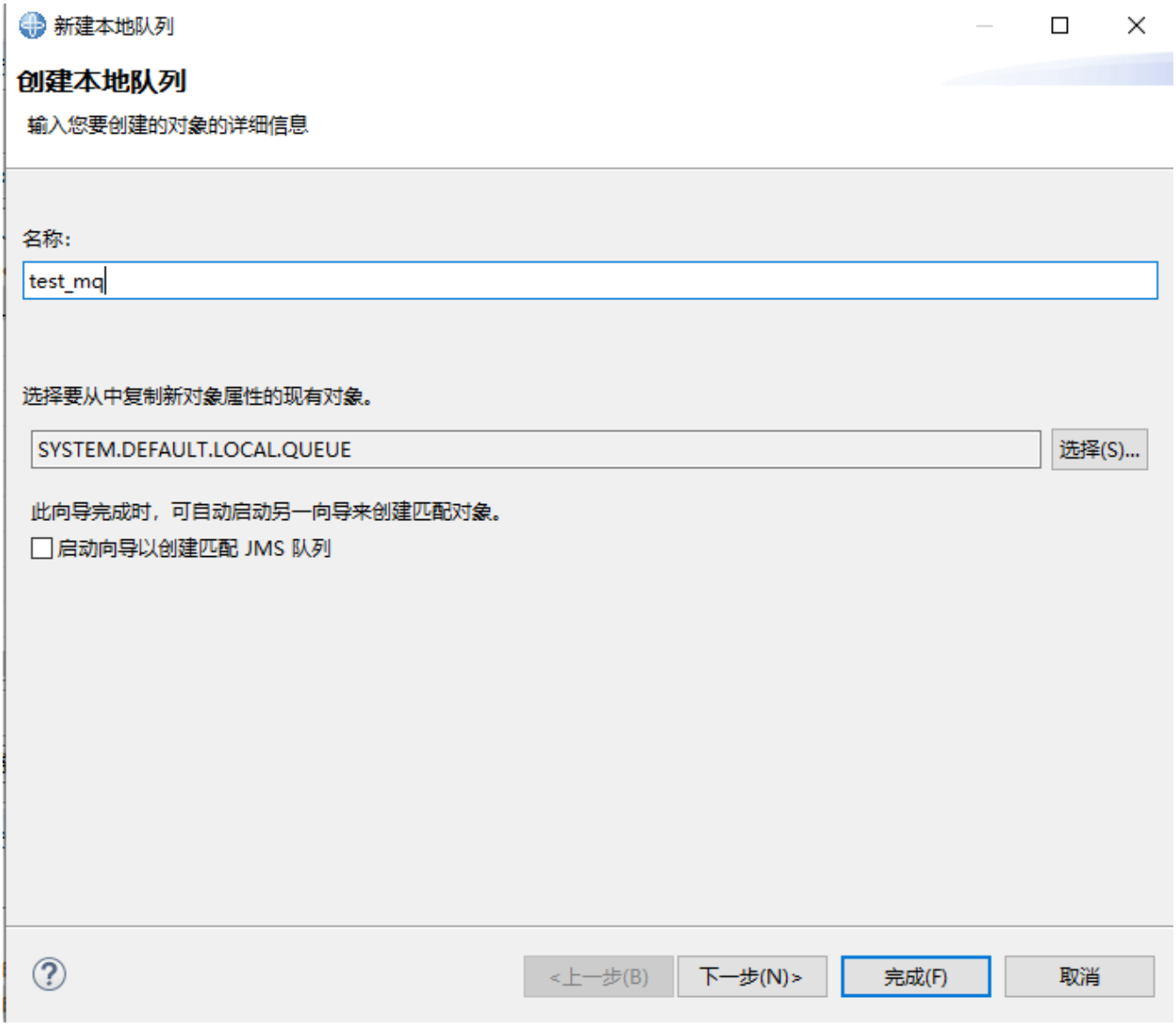
IBM MQ开发手册

下载地址

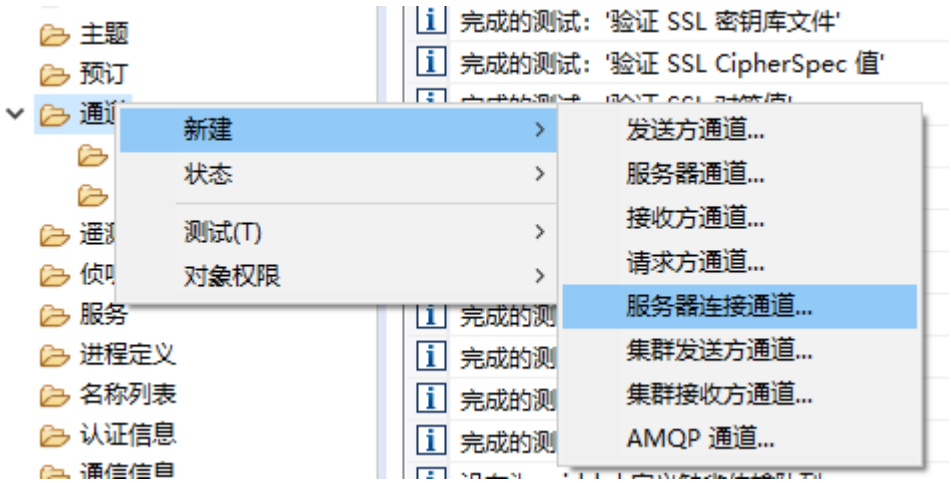
- 安装过程（略）
- 打开MQ资源管理器，新建 队列管理器

- 展开刚刚新建的队列管理器，选中 队列--新建--本地队列

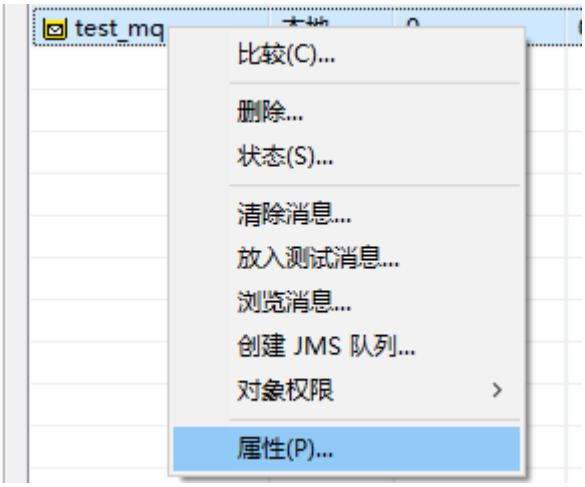

队列名称
队列类型
打开输入
新建
状态(S)...
测试(T)
对象权限
本地队列...
别名队列...
模型队列...
远程队列定义...


新建本地队列
创建本地队列
输入您要创建的对象的详细信息
名称:
test_mq
选择要从中复制新对象属性的现有对象。
SYSTEM.DEFAULT.LOCAL.QUEUE
选择(S)...
此向导完成时，可自动启动另一向导来创建匹配对象。
启动向导以创建匹配 JMS 队列
<上一步(B)
下一步(N)>
完成(F)
取消

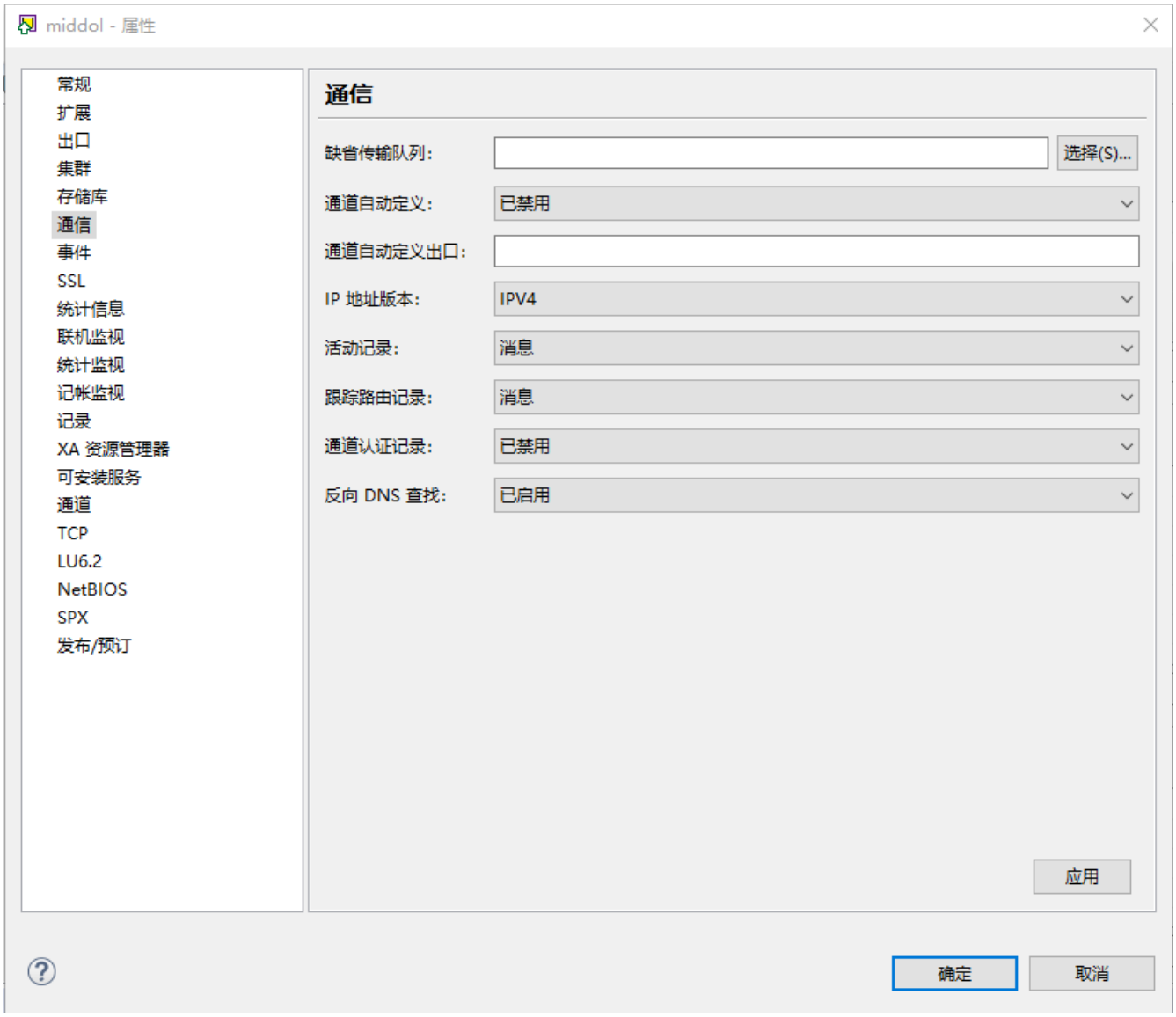
- 新建通道



- 设置队列属性



- 选中新建的队列管理器--属性



注意

- windows 系统需要设置登录账户的密码，一管理员身份运行 dos

```
net user Administrator 123456
```

项目实例

- 依赖

```
compile group: 'org.springframework', name: 'spring-jms'
compile group: 'javax.jms', name: 'javax.jms-api', version: '2.0.1'
compile group: 'com.ibm.mq', name: 'com.ibm.mq.allclient', version: '9.1.2.0'
```

- 配置文件

```
project:
    mq:
        host: 127.0.0.1
        port: 1414
        queue-manager: test # 队列管理器
        channel: MY_CHANNEL # 通道
        username: Administrator # 计算机登录账户
        password: 123456 # 计算机登录密码
        receive-timeout: 20000
```

- `src/main/groovy` 下新建类

> JmsConfig.groovy

```
package com.ibm.mq

import com.ibm.mq.jms.MQQueueConnectionFactory
import com.ibm.msg.client.wmq.WMQConstants
import org.springframework.beans.factory.annotation.Value
import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration
import org.springframework.context.annotation.Primary
import org.springframework.jms.connection.CachingConnectionFactory
import org.springframework.jms.connection.JmsTransactionManager
import org.springframework.jms.connection.UserCredentialsConnectionFactoryAdapter
import org.springframework.jms.core.JmsOperations
import org.springframework.jms.core.JmsTemplate
import org.springframework.transaction.PlatformTransactionManager

@Configuration
class JmsConfig {

    @Value('${project.mq.host}')
    private String host
    @Value('${project.mq.port}')
    private Integer port
    @Value('${project.mq.queue-manager}')
    private String queueManager
    @Value('${project.mq.channel}')
    private String channel
    @Value('${project.mq.username}')
    private String username
```

```
    @Value('${project.mq.password}')
    private String password
    @Value('${project.mq.receive-timeout}')
    private long receiveTimeout

    @Bean
    MQQueueConnectionFactory mqQueueConnectionFactory() {
        MQQueueConnectionFactory mqQueueConnectionFactory = new MQQueueConnectionFactory()
        mqQueueConnectionFactory.setHostName(host)
        try {
            mqQueueConnectionFactory.setTransportType(WMQConstants.WMQ_CM_CLIENT)
            mqQueueConnectionFactory.setCCSID(1208)
            mqQueueConnectionFactory.setChannel(channel)
            mqQueueConnectionFactory.setPort(port)
            mqQueueConnectionFactory.setQueueManager(queueManager)
        } catch (Exception e) {
            e.printStackTrace()
        }
        return mqQueueConnectionFactory
    }
    @Bean
    UserCredentialsConnectionFactoryAdapter
userCredentialsConnectionFactoryAdapter(MQQueueConnectionFactory mqQueueConnectionFactory)
{
        UserCredentialsConnectionFactoryAdapter userCredentialsConnectionFactoryAdapter =
new UserCredentialsConnectionFactoryAdapter()
        userCredentialsConnectionFactoryAdapter.setUsername(username)
        userCredentialsConnectionFactoryAdapter.setPassword(password)

 userCredentialsConnectionFactoryAdapter.setTargetConnectionFactory(mqQueueConnectionFacto
ry)
        return userCredentialsConnectionFactoryAdapter
    }
    @Bean
    @Primary
    CachingConnectionFactory
cachingConnectionFactory(UserCredentialsConnectionFactoryAdapter
userCredentialsConnectionFactoryAdapter) {
        CachingConnectionFactory cachingConnectionFactory = new CachingConnectionFactory()

 cachingConnectionFactory.setTargetConnectionFactory(userCredentialsConnectionFactoryAdapt
er)
        cachingConnectionFactory.setSessionCacheSize(500)
        cachingConnectionFactory.setReconnectOnException(true)
        return cachingConnectionFactory
    }
    @Bean
```

```
    PlatformTransactionManager jmsTransactionManager(CachingConnectionFactory
cachingConnectionFactory) {
        JmsTransactionManager jmsTransactionManager = new JmsTransactionManager()
        jmsTransactionManager.setConnectionFactory(cachingConnectionFactory)
        return jmsTransactionManager
    }
    @Bean
    JmsOperations jmsOperations(CachingConnectionFactory cachingConnectionFactory) {
        JmsTemplate jmsTemplate = new JmsTemplate(cachingConnectionFactory)
        jmsTemplate.setReceiveTimeout(receiveTimeout)
        return jmsTemplate
    }
}
```

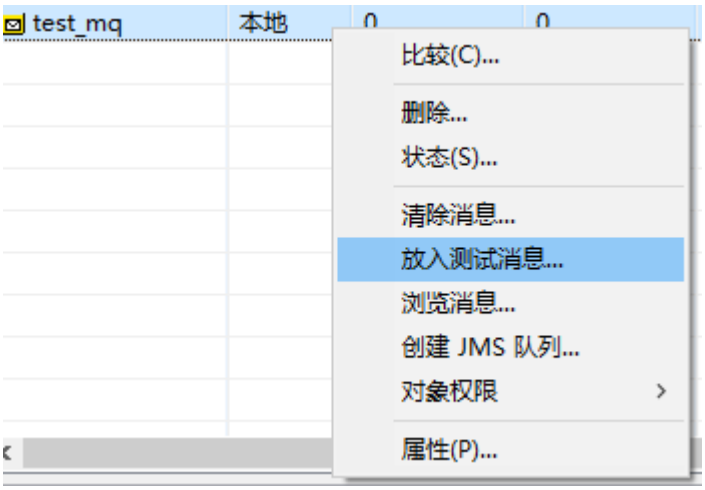
ReceiveMessage.groovy

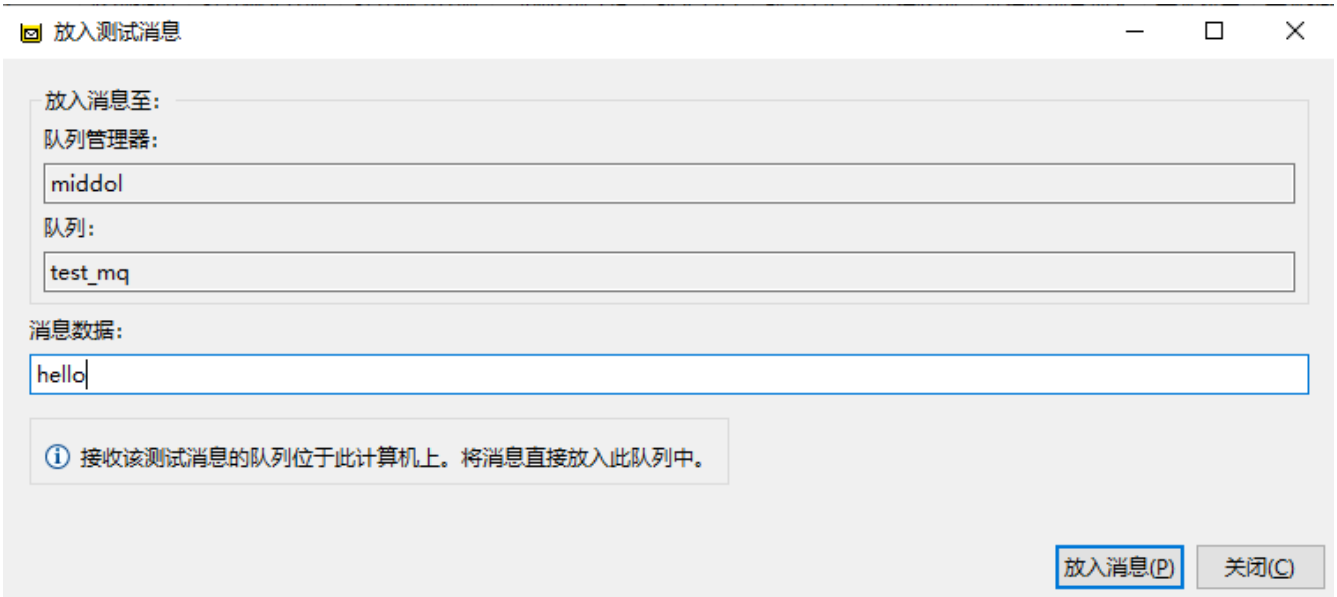
```
package com.ibm.mq

import org.springframework.beans.factory.annotation.Autowired
import org.springframework.jms.annotation.JmsListener
import org.springframework.jms.core.JmsOperations
import org.springframework.jms.listener.adapter.MessageListenerAdapter
import org.springframework.stereotype.Component

import javax.jms.Message

//消息消费者的类上必须加上@Component，或者是@Service，这样的话，消息消费者类就会被委派给Listener类，原理类似于使用SessionAwareMessageListener以及MessageListenerAdapter来实现消息驱动POJO
@Component
class ReceiveMessage extends MessageListenerAdapter {

    @Autowired
    JmsOperations jmsOperations;

    /**
     *  @destination test_mq对应ibm中创建的队列
     * @param message
     */
    @Override
    @JmsListener(destination = "test_mq")
    void onMessage(Message message) {
        String messageBody = new String(message.getBody(Object).toString());
        println "成功监听middol_send消息队列，传来的值为：${messageBody}"
    }
}
```

在主类添加扫描注解

```
@ComponentScan("com.ibm.mq")
```

测试，启动项目，打开IBM MQ资源管理器，选中 队列 -- 右键 -- 放入测试数据





- 后台控制台打印

```
|Running application...
Grails application running at http://localhost:8080 in environment: development
成功监听middol_send消息队列，传来的值为: hello
```