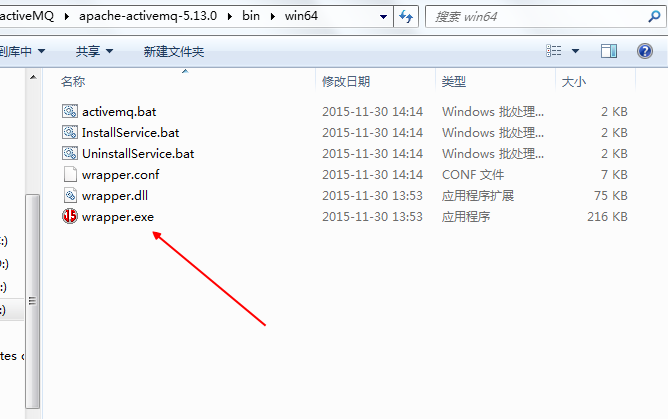
**activeMQ环境搭建详解**

下载demo代码

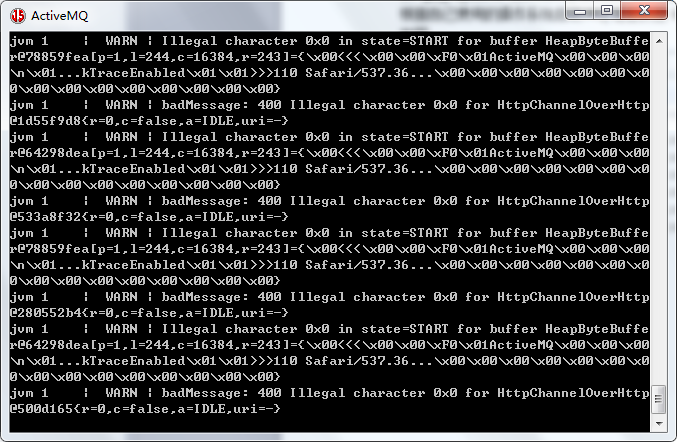
1. 官网下载activeMQ [http://activemq.apache.org/](http://activemq.apache.org/" \t "http://www.cnblogs.com/xwdreamer/archive/2012/02/21/_blank)
2. 解压apache-activemq-5.13.0-bin.zip

根据自己使用的操作系统启动MQ笔者的是64位win7 系统

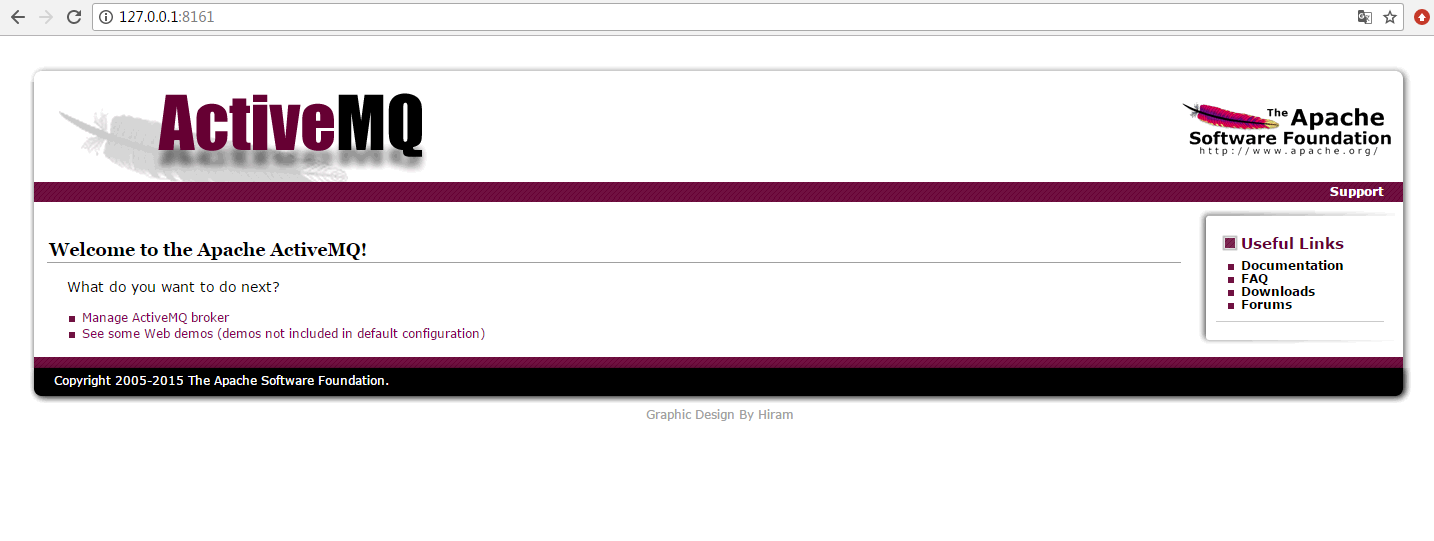
如图



启动好的cmd窗口

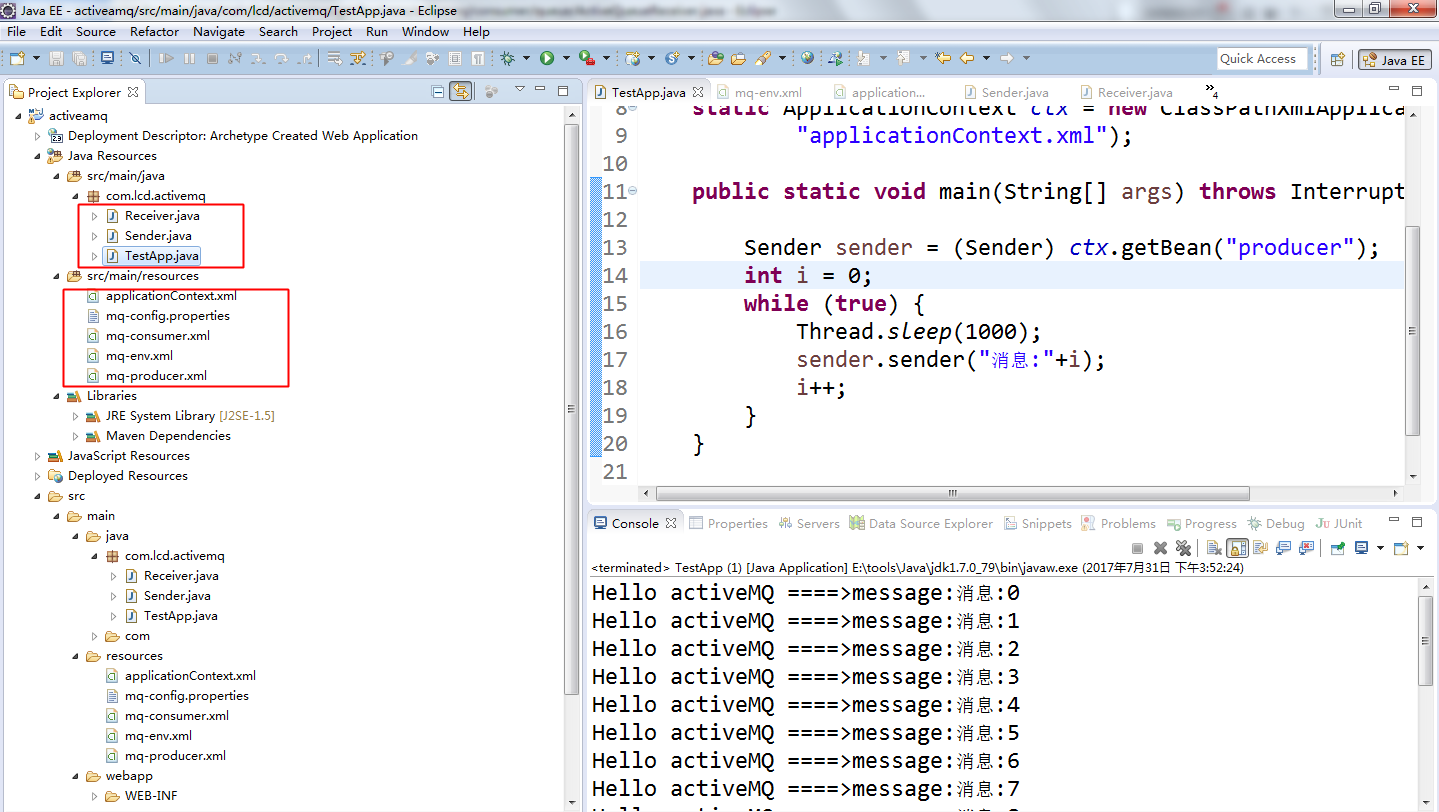


访问网页测试



# 普通消息

1. 编写代码如图包含发送者、接收者、测试类和spring的xml配置文件



1. 代码清单

(配置文件)

**mq-config.properties**

activemq.connection.brokerURL=tcp://localhost:61616

activemq.connection.userName=admin

activemq.connection.password=admin

activemq.queue.example=example.queue

**mq-env.xml**

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context" xmlns:oxm="http://www.springframework.org/schema/oxm"

xsi:schemaLocation="http://www.springframework.org/schema/oxm http://www.springframework.org/schema/oxm/spring-oxm-3.2.xsd

http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.2.xsd">

<bean id="configureHolder" class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">

<property name="locations">

<list>

<value>classpath:mq-config.properties</value>

</list>

</property>

</bean>

<bean id="mqConnectionFactory" class="org.apache.activemq.spring.ActiveMQConnectionFactory">

<property name="brokerURL" value="${activemq.connection.brokerURL}"></property>

<property name="userName" value="${activemq.connection.userName}"></property>

<property name="password" value="${activemq.connection.password}"></property>

</bean>

<bean id="mqPooledConnectionFactory" class="org.apache.activemq.pool.PooledConnectionFactory" destroy-method="stop">

<property name="connectionFactory" ref="mqConnectionFactory">

</property>

<property name="maxConnections" value="100"></property>

</bean>

<bean id="exampleQueue" class="org.apache.activemq.command.ActiveMQQueue">

<constructor-arg index="0" value="${activemq.queue.example}"></constructor-arg>

</bean>

<bean id="jmsTemplate" class="org.springframework.jms.core.JmsTemplate">

<property name="connectionFactory" ref="mqPooledConnectionFactory"></property>

<property name="defaultDestination" ref="exampleQueue"></property>

</bean>

</beans>

**mq-producer.xml**

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="producer" class="com.lcd.activemq.Sender"></bean>

</beans>

**mq-consumer.xml**

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:jms="http://www.springframework.org/schema/jms" xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/jms http://www.springframework.org/schema/jms/spring-jms-3.2.xsd

http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.2.xsd">

<bean id="consumer" scope="prototype" class="com.lcd.activemq.Receiver" />

<jms:listener-container acknowledge="auto" connection-factory="mqPooledConnectionFactory">

<jms:listener destination="${activemq.queue.example}" ref="consumer" method="process"/>

</jms:listener-container>

</beans>

**applicationContext.xml**

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:p="http://www.springframework.org/schema/p"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd">

<context:component-scan base-package="\*"></context:component-scan>

<import resource="mq-env.xml"/>

<import resource="mq-producer.xml"/>

<import resource="mq-consumer.xml"/>

</beans>

**(java代码)**

**Sender.java**

package com.lcd.activemq;

import javax.jms.JMSException;

import javax.jms.Message;

import javax.jms.Session;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.jms.core.JmsTemplate;

import org.springframework.jms.core.MessageCreator;

public class Sender {

@Autowired

@Qualifier(value = "jmsTemplate")

private JmsTemplate jmsTemplate;

public void sender(final String message) {

jmsTemplate.send(new MessageCreator() {

public Message createMessage(Session session) throws JMSException {

StringBuilder sb = new StringBuilder();

sb.append("Hello activeMQ ====>message:" + message);

return session.createObjectMessage(sb.toString());

}

});

}

}

**Receiver.java**

package com.lcd.activemq;

public class Receiver {

public void process(String message) throws Exception {

//Thread.sleep(1000);

System.out.println(message);

}

}

**TestApp.java**

package com.lcd.activemq;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class TestApp {

static ApplicationContext ctx = new ClassPathXmlApplicationContext(

"applicationContext.xml");

public static void main(String[] args) throws InterruptedException {

Sender sender = (Sender) ctx.getBean("producer");

int i = 0;

while (true) {

Thread.sleep(1000);

sender.sender("消息:"+i);

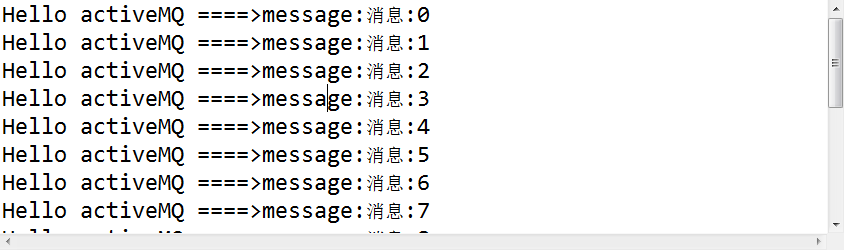
i++;

}

}

}

运行结果:



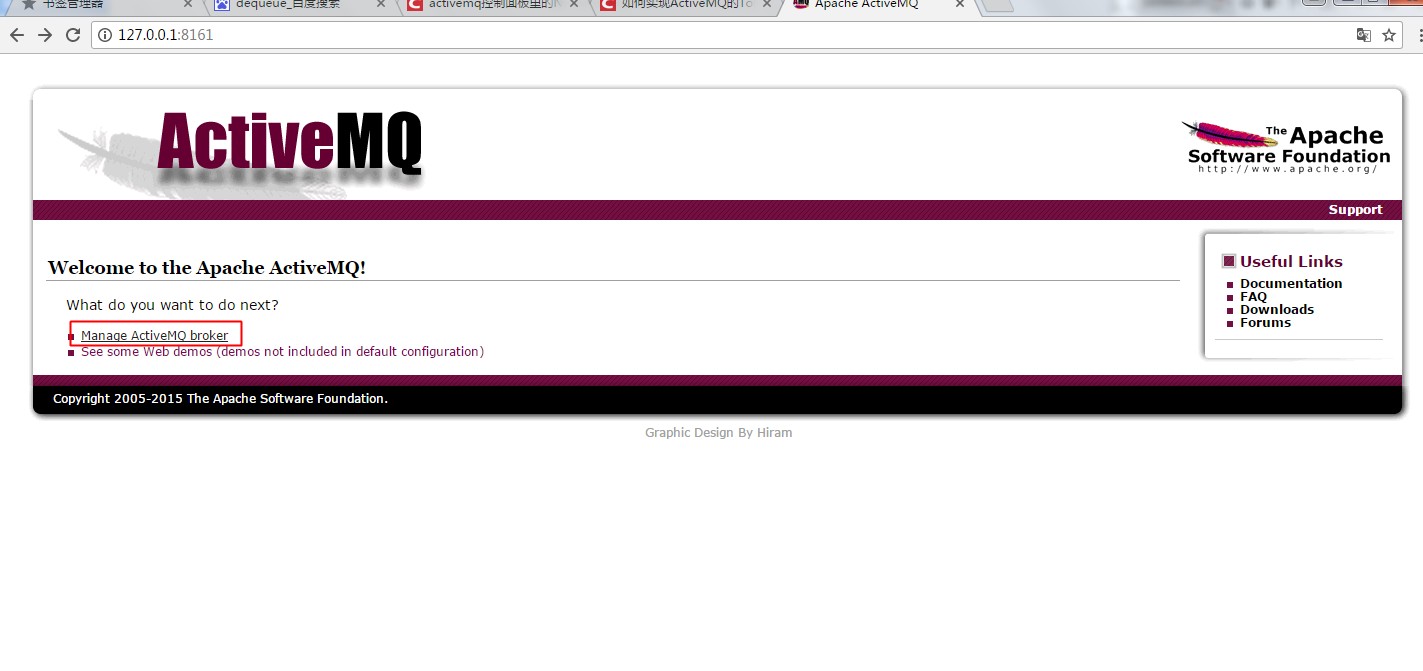
欢迎扫描头像关注嘟嘟技术下载更多技术源码！

# 订阅(主题)消息(消息持久化)

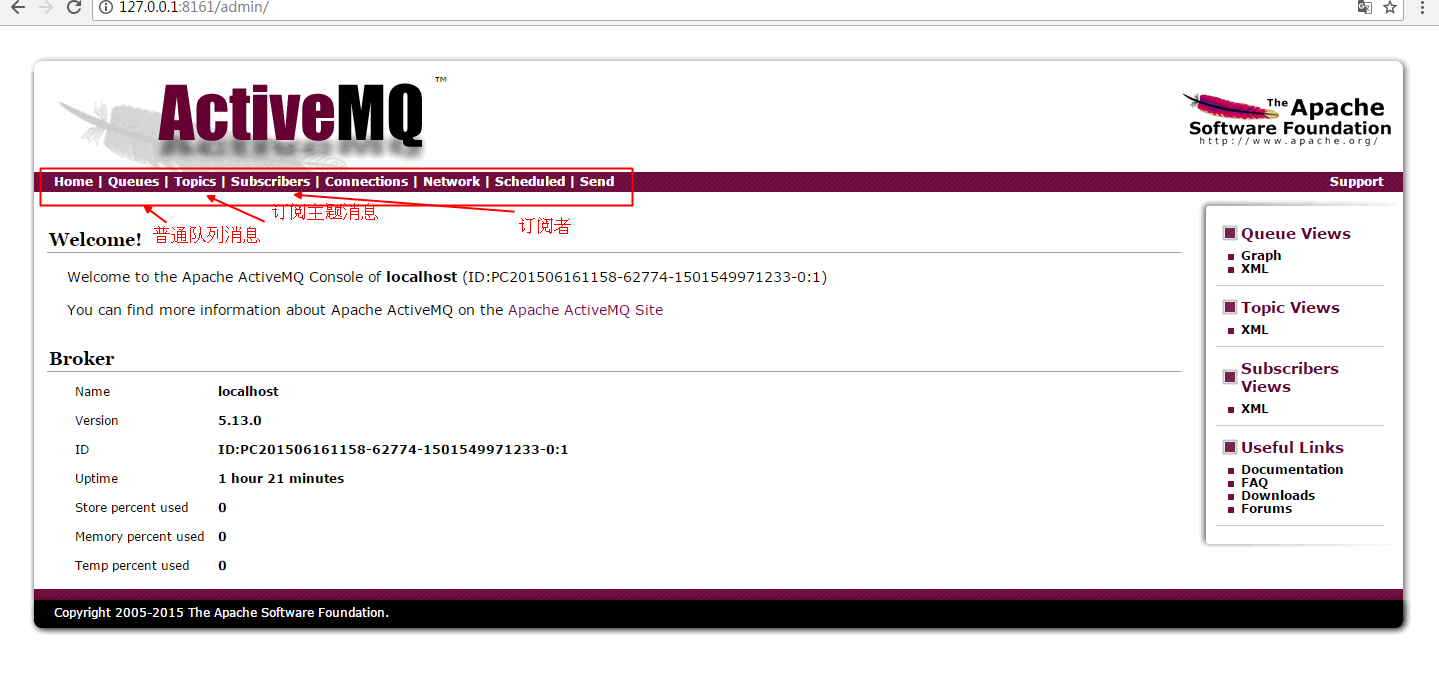
管理后台探究

1. 网页打开http://127.0.0.1:8161/

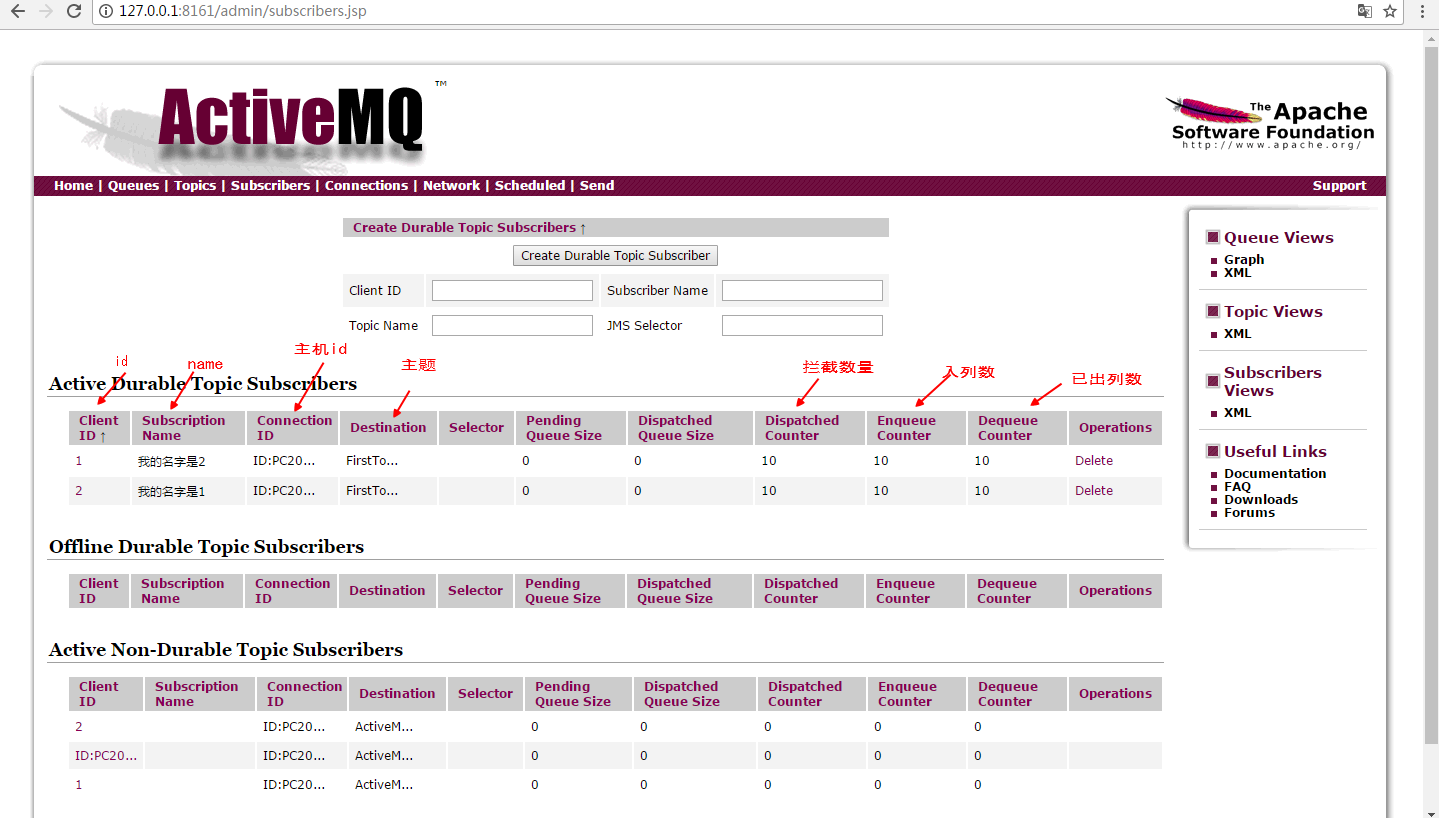
输入默认的用户名密码 admin/admin



1. 进入后可以看到消息监控页面

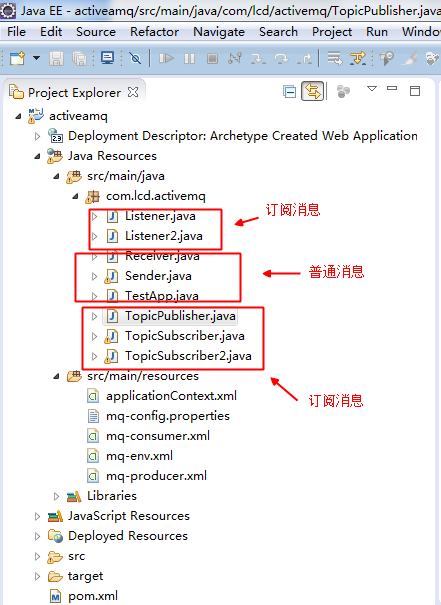


1. 订阅者详情页面解释



初步了解了后台监控现在开始撸代码

1. 代码结构 普通消息已经在上一节说过了这里就不再赘述(待更新)



1. 代码清单（由于订阅者中差别就只是一个id和一个name不同这里只贴出一份）

**TopicPublisher.java**

package com.lcd.activemq;

import javax.jms.Connection;

import javax.jms.ConnectionFactory;

import javax.jms.Destination;

import javax.jms.JMSException;

import javax.jms.MessageProducer;

import javax.jms.Session;

import javax.jms.TextMessage;

import org.apache.activemq.ActiveMQConnection;

import org.apache.activemq.ActiveMQConnectionFactory;

/\*\*

\* 订阅消息

\* @author Administrator

\*

\*/

public class TopicPublisher {

private static final String USERNAME = ActiveMQConnection.DEFAULT\_USER; // 默认的连接用户名

private static final String PASSWORD = ActiveMQConnection.DEFAULT\_PASSWORD; // 默认的连接密码

private static final String BROKEURL = ActiveMQConnection.DEFAULT\_BROKER\_URL; // 默认的连接地址

private static final int SENDNUM = 10; // 发送的消息数量

public static void main(String[] args) {

ConnectionFactory connectionFactory; // 连接工厂

Connection connection = null; // 连接

Session session; // 会话 接受或者发送消息的线程

Destination destination; // 消息的目的地

MessageProducer messageProducer; // 消息生产者

// 实例化连接工厂

connectionFactory = new ActiveMQConnectionFactory(TopicPublisher.USERNAME, TopicPublisher.PASSWORD,

TopicPublisher.BROKEURL);

try {

connection = connectionFactory.createConnection(); // 通过连接工厂获取连接

connection.start(); // 启动连接

session = connection.createSession(Boolean.TRUE, Session.AUTO\_ACKNOWLEDGE); // 创建Session

// destination=session.createQueue("FirstQueue1"); // 创建消息队列

destination = session.createTopic("FirstTopic1");

messageProducer = session.createProducer(destination); // 创建消息生产者

sendMessage(session, messageProducer); // 发送消息

session.commit();

} catch (Exception e) {

e.printStackTrace();

} finally {

if (connection != null) {

try {

connection.close();

} catch (JMSException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

}

/\*\*

\* 发送消息

\*

\* @param session

\* @param messageProducer

\* @throws Exception

\*/

public static void sendMessage(Session session, MessageProducer messageProducer) throws Exception {

for (int i = 0; i < TopicPublisher.SENDNUM; i++) {

TextMessage message = session.createTextMessage("ActiveMQ 发送的消息" + i);

System.out.println("发送消息：" + "ActiveMQ 发布的消息" + i);

messageProducer.send(message);

}

}

}

订阅者

**TopicSubscriber.java**

package com.lcd.activemq;

import javax.jms.Connection;

import javax.jms.ConnectionFactory;

import javax.jms.Destination;

import javax.jms.JMSException;

import javax.jms.MessageConsumer;

import javax.jms.Session;

import javax.jms.Topic;

import org.apache.activemq.ActiveMQConnection;

import org.apache.activemq.ActiveMQConnectionFactory;

public class TopicSubscriber {

private static final String USERNAME=ActiveMQConnection.DEFAULT\_USER; // 默认的连接用户名

private static final String PASSWORD=ActiveMQConnection.DEFAULT\_PASSWORD; // 默认的连接密码

private static final String BROKEURL=ActiveMQConnection.DEFAULT\_BROKER\_URL; // 默认的连接地址

public static void main(String[] args) {

ConnectionFactory connectionFactory; // 连接工厂

Connection connection = null; // 连接

Session session; // 会话 接受或者发送消息的线程

Topic topic; // 消息的目的地

MessageConsumer messageConsumer; // 消息的消费者

// 实例化连接工厂

connectionFactory=new ActiveMQConnectionFactory(TopicSubscriber.USERNAME, TopicSubscriber.PASSWORD, TopicSubscriber.BROKEURL);

try {

connection=connectionFactory.createConnection(); // 通过连接工厂获取连接

connection.setClientID("2"); //持久订阅需要设置这个订阅id

connection.start(); // 启动连接

session=connection.createSession(Boolean.FALSE, Session.AUTO\_ACKNOWLEDGE); // 创建Session

// destination=session.createQueue("FirstQueue1"); // 创建连接的消息队列

topic =session.createTopic("FirstTopic1");

//messageConsumer=session.createConsumer(destination); // 创建消息消费者

//这里的121是Subscription Name可以和id不同

messageConsumer = session.createDurableSubscriber(topic,"我的名字是1"); //持久订阅

messageConsumer.setMessageListener(new Listener()); // 注册消息监听

} catch (JMSException e) {

e.printStackTrace();

}

}

}

**订阅者监听**

**Listener.java**

package com.lcd.activemq;

import javax.jms.JMSException;

import javax.jms.Message;

import javax.jms.MessageListener;

import javax.jms.TextMessage;

public class Listener implements MessageListener {

public void onMessage(Message message) {

try {

System.out.println("订阅者一收到的消息："+((TextMessage)message).getText());

} catch (JMSException e) {

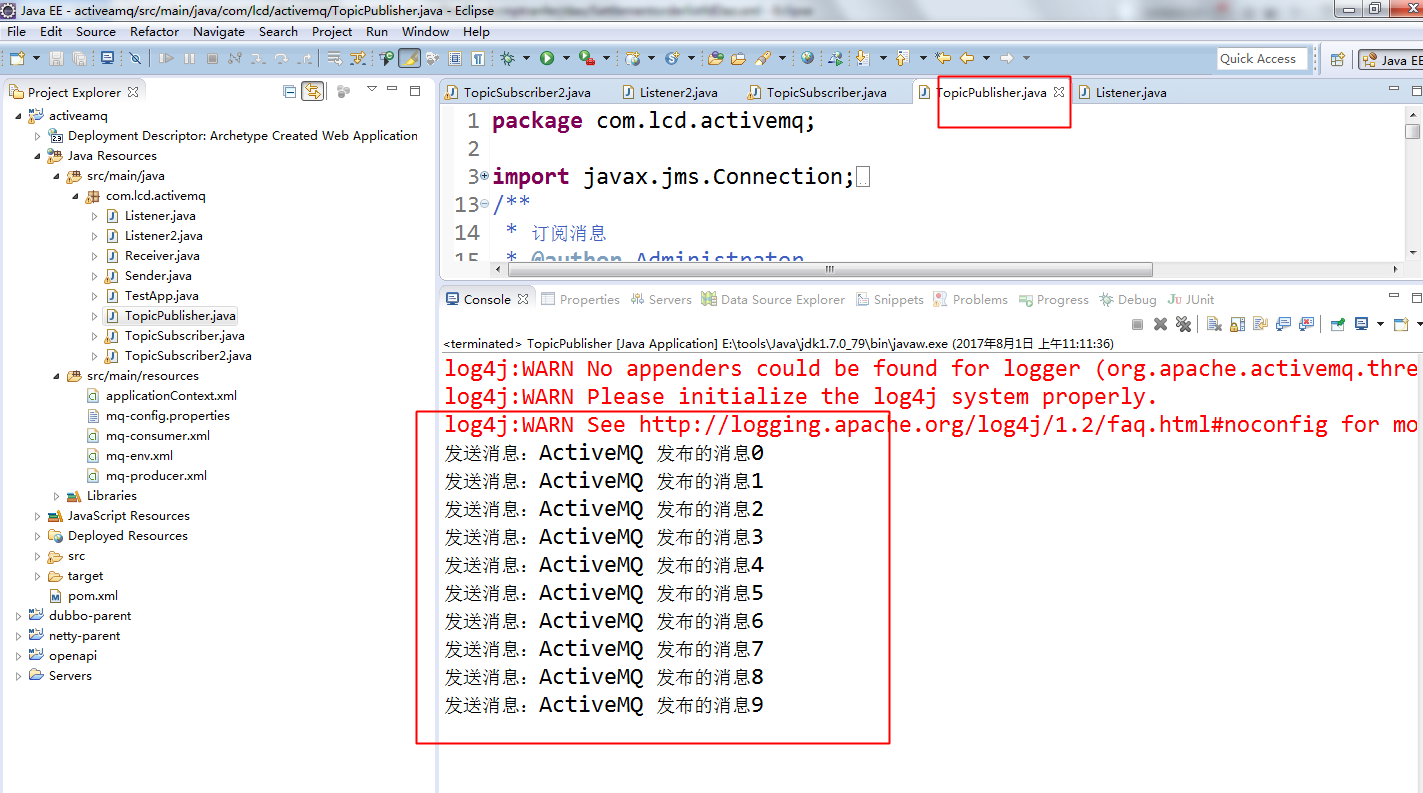
e.printStackTrace();

}

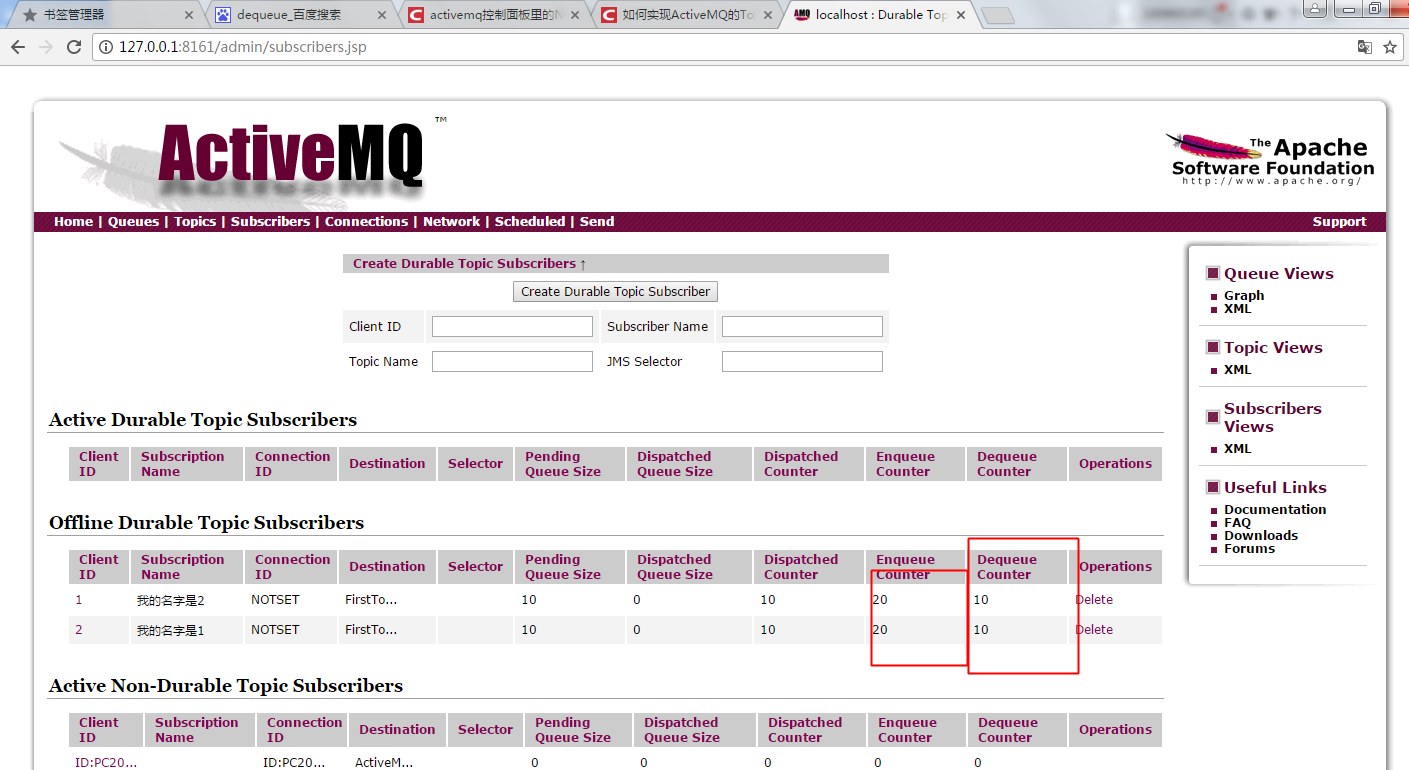
}

}

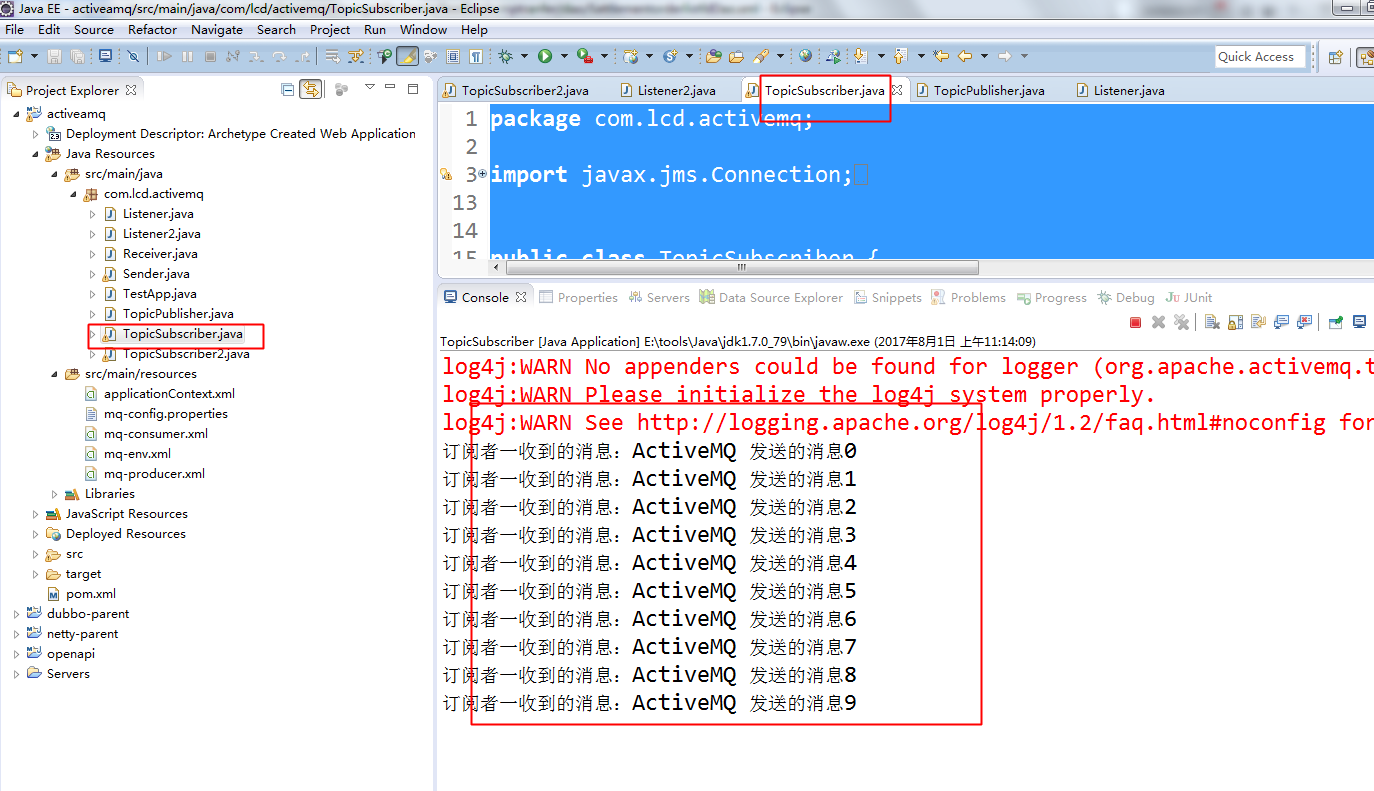
1. 运行测试



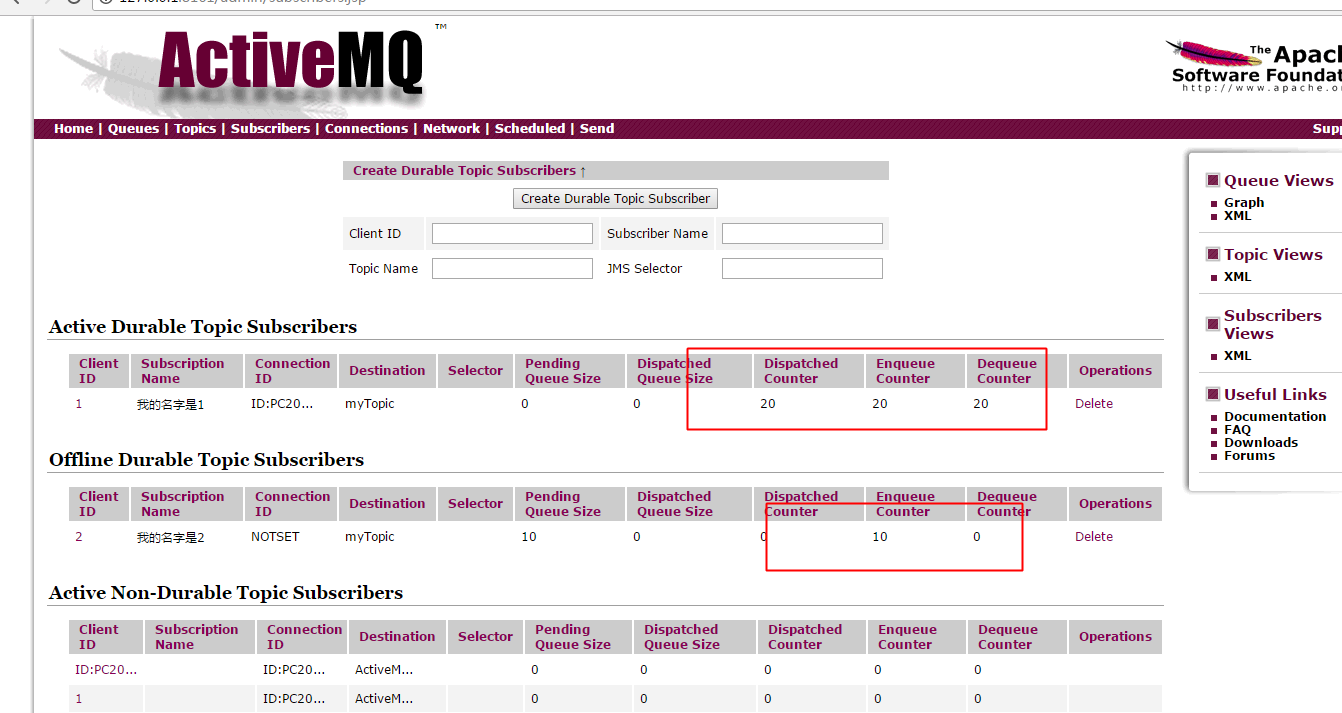
后台:因为我还没有启动订阅者所以有10条消息还没有出队列 没有消费



**启动订阅者1:成功消费消息**



后台变化：id=1的已经推送所有消息 2 的还有10条消息未推送



后续：

常见异常

1、消息无法订阅接收

原因：推送者无session.commit();消息未提交到服务器 、订阅者没有第一次启动注册到服务器

2、同时启动连个id一样的订阅者报错与名字无关

log4j:WARN No appenders could be found for logger (org.apache.activemq.thread.TaskRunnerFactory).

log4j:WARN Please initialize the log4j system properly.

log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.

javax.jms.InvalidClientIDException: Broker: localhost - Client: 1 already connected from tcp://127.0.0.1:52650

at org.apache.activemq.broker.region.RegionBroker.addConnection(RegionBroker.java:255)

at org.apache.activemq.broker.jmx.ManagedRegionBroker.addConnection(ManagedRegionBroker.java:230)

at org.apache.activemq.broker.BrokerFilter.addConnection(BrokerFilter.java:98)

at org.apache.activemq.advisory.AdvisoryBroker.addConnection(AdvisoryBroker.java:116)

at org.apache.activemq.broker.BrokerFilter.addConnection(BrokerFilter.java:98)

at org.apache.activemq.broker.BrokerFilter.addConnection(BrokerFilter.java:98)

at org.apache.activemq.broker.MutableBrokerFilter.addConnection(MutableBrokerFilter.java:103)

at org.apache.activemq.broker.TransportConnection.processAddConnection(TransportConnection.java:818)

at org.apache.activemq.broker.jmx.ManagedTransportConnection.processAddConnection(ManagedTransportConnection.java:77)

at org.apache.activemq.command.ConnectionInfo.visit(ConnectionInfo.java:139)

at org.apache.activemq.broker.TransportConnection.service(TransportConnection.java:339)

at org.apache.activemq.broker.TransportConnection$1.onCommand(TransportConnection.java:188)

at org.apache.activemq.transport.MutexTransport.onCommand(MutexTransport.java:50)

at org.apache.activemq.transport.WireFormatNegotiator.onCommand(WireFormatNegotiator.java:125)

at org.apache.activemq.transport.AbstractInactivityMonitor.onCommand(AbstractInactivityMonitor.java:300)

at org.apache.activemq.transport.TransportSupport.doConsume(TransportSupport.java:83)

at org.apache.activemq.transport.tcp.TcpTransport.doRun(TcpTransport.java:233)

at org.apache.activemq.transport.tcp.TcpTransport.run(TcpTransport.java:215)

at java.lang.Thread.run(Thread.java:745)