

# Основные положения технологии Java Message Service (JMS)

## ActiveMQ

Подготовила Уренева Юлия

# Часть I.

# Определение

JMS - это система передачи сообщений, изначально разработанная компанией Sun, с целью предоставления разработчикам возможности создавать гибкие и слабосвязанные приложения с использованием асинхронного обмена данными между приложениями (клиентами/ серверами) через посредника.

# Принцип работы

Общий алгоритм работы с JMS следующий: некое приложение (поставщик) хочет передать данные другому приложению (получателю), или даже нескольким приложениям. Поставщик отправляет свое сообщение на JMS-сервер, на котором оно хранится указанное время или же до востребования получателем. Сам же получатель проверяет JMS-сервер с целью проверки и считывания имеющихся для него сообщений. JMS-сервер может самостоятельно рассылать одно и то же приложение всем получателям (если получателей больше одного), что снимает большой объем работы с поставщика.

# Модели передачи сообщений

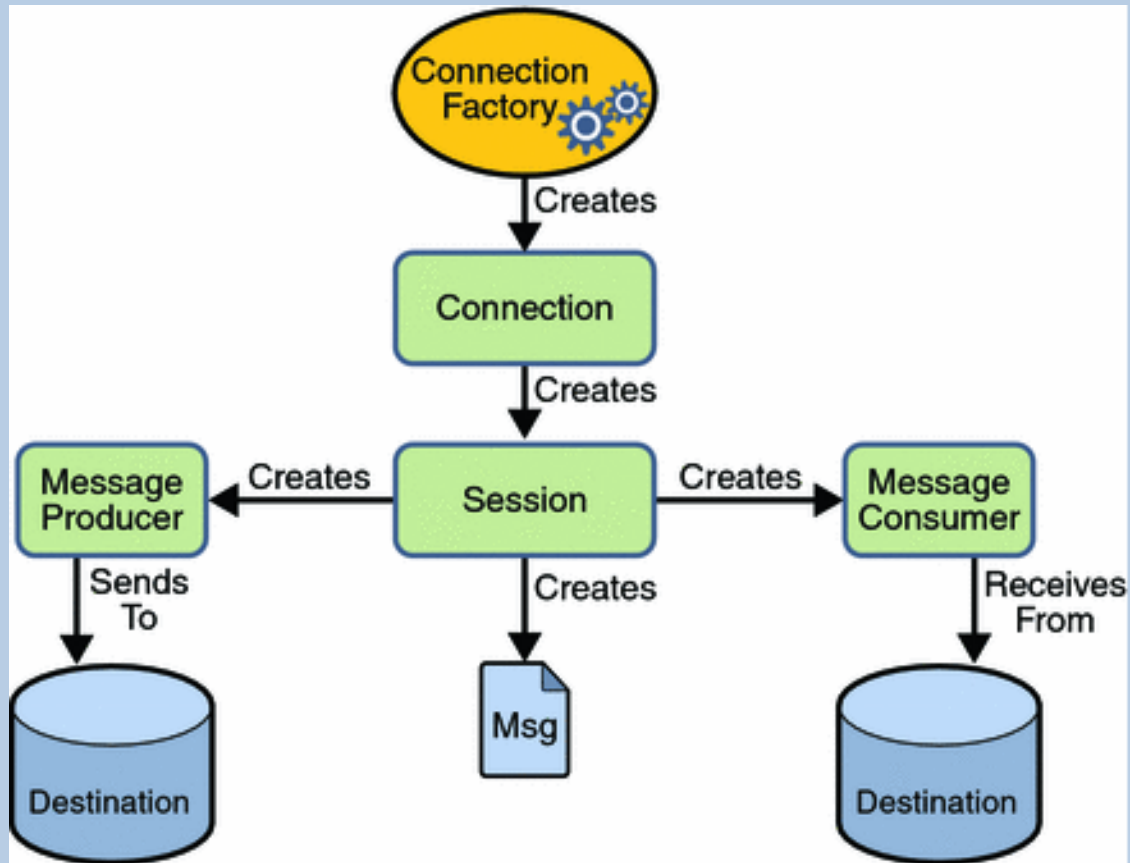
В JMS имеется две модели передачи сообщений, для которых используются два вида объекта:

1. Модель Точка-Точка (Point-to-Point). Объект Queue.
2. Модель Подписчик-Издатель (Publisher-Subscriber). Объект Topic.

# Интерфейсы JMS

ConnectionFactory	Объект, создающий Connection. В параметре инициализации нужно передавать данные JMS-сервера.
Connection	Соединение с сервером JMS. Создает объект Session.
Session	Контекст, реализующий передачу и получение сообщений. Управляет транзакцией в JMS и его использование разными потоками невозможно или ограничено. Рекомендуется создавать разные интерфейсы для каждого потока. Создает объект Destination.
Destination	Объект, хранящий адреса сообщений (имя топика или очереди). С его помощью создаются объекты, отвечающие за отсылку сообщений на указанный адрес и их получения оттуда.
MessageProducer	Объект для отправки сообщений
MessageConsumer	Объект для получения сообщений

# Алгоритм создания программ, работающих с JMS



**Часть II.**

**Apache ActiveMQ**



# Введение

Apache ActiveMQ — это message broker с открытым исходным кодом (распространяется под лицензией Apache 2.0), который полностью реализует Java Message Service 1.1 (JMS). Он обеспечивает такие «Enterprise Features», как кластеризация, хранение сообщений с возможностью использовать различные базы данных, кэширование и ведение журналов.

Кроме Java, ActiveMQ может также быть использованным из .NET, C/C++ или Delphi или из скриптовых языков, как Perl, Python, PHP и Ruby через различные «Кросс Языковые Клиенты» совместно с многочисленными протоколами и платформами.

Преимущества данной системы - высокая производительность, открытость и возможность реализации клиентов на любых языках.

# Введение

1. AMQP 1.0 - (Advanced Message Queuing Protocol)
2. MQTT - (Message Queuing Telemetry Transport)
3. OpenWire - кроссплатформенный протокол сообщений (основной для ActiveMQ)
4. REST – (Representational State Transfer)
5. RSS – (Really Simple Syndication)
6. Atom
7. Stomp - кроссплатформенный протокол сообщений (основной для ActiveMQ)
8. WSIF – (Web Services Invocation Framework)
9. WS Notification
10. XMPP – (Extensible Messaging and Presence Protocol)
11. Поддержка транспортных протоколов, таких как - in-VM, TCP, SSL, NIO, UDP, multicast, JGroups and JXTA.
12. Ajax.
13. Spring FrameWork.

# Установка ActiveMQ

Для пользователей Windows:

1. Скачать архив с сайта <http://activemq.apache.org>
2. Разархивировать архив и в папке bin запустить bat-файл `activemq.bat`.

Для пользователей Linux:

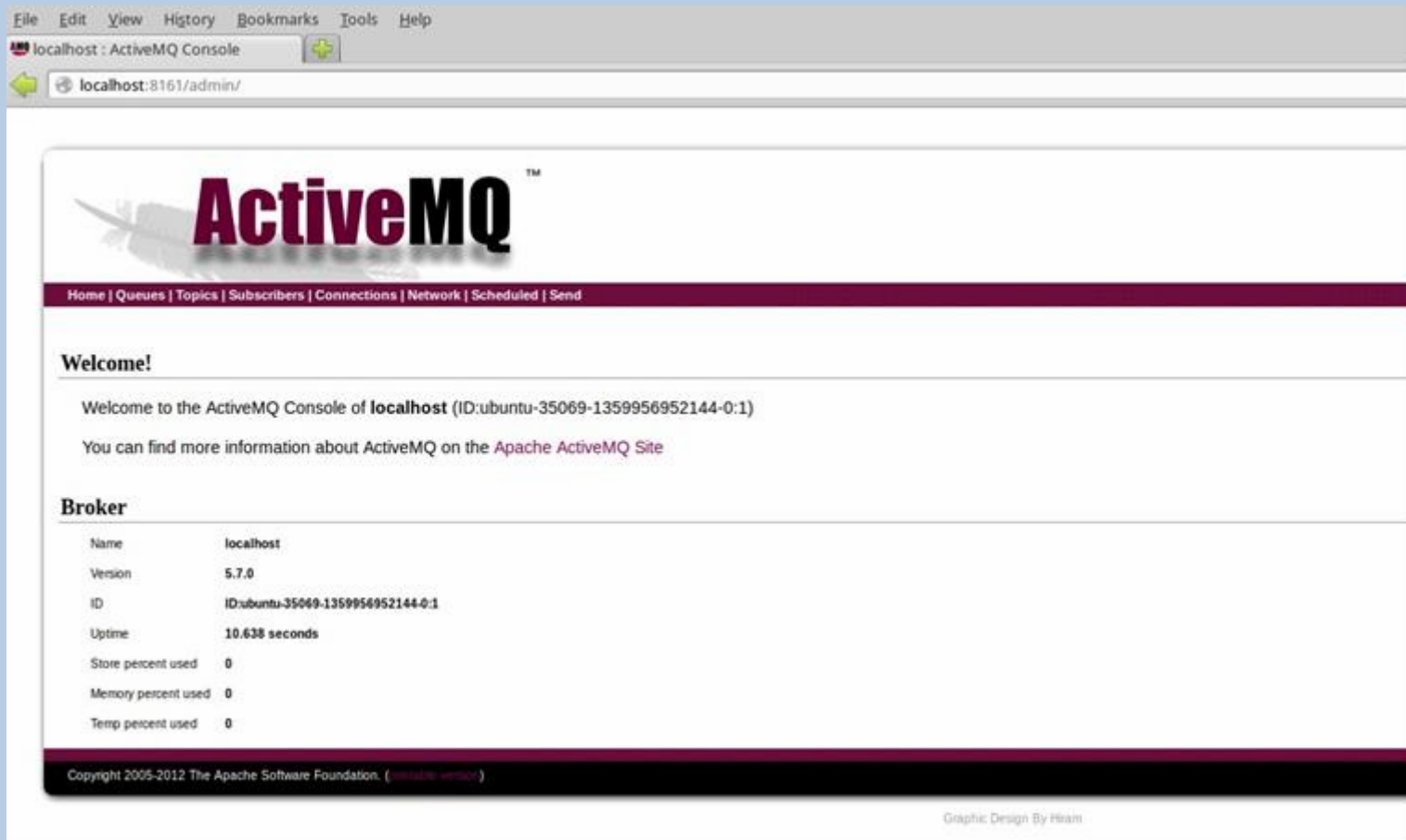
1. Скачать архив с сайта <http://activemq.apache.org> (`wget`)
2. Разархивировать архив (`tar xvfz apache*.tar.gz`)
3. Запустить ActiveMQ командой `./activemq start`

# Знакомство с веб-консолью ActiveMQ

Веб-консоль находится по адресу:

<http://localhost:8161/admin/>

# Знакомство с веб-консолью ActiveMQ



The image shows a screenshot of a web browser displaying the ActiveMQ console. The browser's address bar shows 'localhost:8161/admin/'. The page features the ActiveMQ logo at the top, followed by a navigation menu with links for Home, Queues, Topics, Subscribers, Connections, Network, Scheduled, and Send. Below the menu, a 'Welcome!' section provides information about the console instance on 'localhost' and includes a link to the Apache ActiveMQ site. A 'Broker' section displays a table of system metrics.

Broker	
Name	localhost
Version	5.7.0
ID	ID:ubuntu-35069-1359956952144-0:1
Uptime	10.638 seconds
Store percent used	0
Memory percent used	0
Temp percent used	0

Copyright 2005-2012 The Apache Software Foundation. ([readable version](#))

Graphic Design By Hiram

# Знакомство с веб-консолью ActiveMQ

The screenshot shows the ActiveMQ web console interface. At the top, there is a navigation bar with the following links: Home | Queues | Topics | Subscribers | Connections | Network | Scheduled | Send. Below the navigation bar, there is a form to create a new queue with a text input field labeled "Queue Name" and a "Create" button. A red arrow points from this input field to the "MYQUEUE" row in the table below.

**Queues**

Name	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued	Views	Operations
FOO.BAR	0	0	0	0	Browse Active Consumers atom rss	Send To Purge Delete
MYQUEUE	0	0	0	0	Browse Active Consumers atom rss	Send To Purge Delete

# Знакомство с веб-консолью ActiveMQ



The screenshot shows the ActiveMQ web console interface for sending a JMS message. At the top, the ActiveMQ logo is displayed with a navigation menu containing links for Home, Queues, Topics, Subscribers, Connections, Network, Scheduled, and Send. Below the navigation bar, the page title is "Send a JMS Message".

The main form is titled "Message Header" and contains the following fields:

Destination	<input type="text" value="MYQUEUE"/>	Queue or Topic	<input type="text" value="Queue"/>
Correlation ID	<input type="text"/>	Persistent Delivery	<input type="checkbox"/>
Reply To	<input type="text"/>	Priority	<input type="text"/>
Type	<input type="text"/>	Time to live	<input type="text"/>
Message Group	<input type="text"/>	Message Group Sequence Number	<input type="text"/>
delay(ms)	<input type="text"/>	Time(ms) to wait before scheduling again	<input type="text"/>
Number of repeats	<input type="text"/>	Use a CRON string for scheduling	<input type="text"/>
Number of messages to send	<input type="text" value="1"/>	Header to store the counter	<input type="text" value="JMSXMessageCounter"/>

Below the form are two buttons: "Send" and "Reset".

The "Message body" section contains a text area with the placeholder text "Enter some text here for the message body..." and a large empty space for input.

# Главный конфигурационный файл ActiveMQ

Главный конфигурационный файл ActiveMQ,  
использующийся по умолчанию, находится в  
`conf/activemq.xml`



# Главный конфигурационный файл ActiveMQ

1. Объявляем схемы:

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

```
xmlns:amq="http://activemq.apache.org/schema/core"
```

```
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  
http://activemq.apache.org/schema/core  
http://activemq.apache.org/schema/core/activemq-core.xsd">
```

# Главный конфигурационный файл ActiveMQ

2. Создаём брокер:

```
<broker xmlns="http://activemq.apache.org/schema/core"  
brokerName="localhost" dataDirectory="{activemq.data}">  
  
<transportConnectors>  
<transportConnector name="openwire"  
uri="tcp://0.0.0.0:61616?maximumConnections=1000&wireform  
at.maxFrameSize=104857600"/>  
</transportConnectors>
```

# Главный конфигурационный файл ActiveMQ

## 3. Устанавливаем Destination Policy:

```
<destinationPolicy>
  <policyMap>
    <policyEntries>
      <policyEntry topic="" producerFlowControl="true">
        <pendingMessageLimitStrategy>
          <constantPendingMessageLimitStrategy limit="1000"/>
        </pendingMessageLimitStrategy>
      </policyEntry>
      <policyEntry queue="" producerFlowControl="true" memoryLimit="1mb">
      </policyEntry>
    </policyEntries>
  </policyMap>
</destinationPolicy>
```

# Главный конфигурационный файл ActiveMQ

4. Описываем использование системных ресурсов:

```
<systemUsage>  
  <systemUsage>  
    <memoryUsage>  
      <memoryUsage limit="64 mb"/>  
    </memoryUsage>  
    <storeUsage>  
      <storeUsage limit="100 gb"/>  
    </storeUsage>  
    <tempUsage>  
      <tempUsage limit="50 gb"/>  
    </tempUsage>  
  </systemUsage>  
</systemUsage>
```

# Главный конфигурационный файл ActiveMQ

5. Настраиваем Persistence options:

```
<persistenceAdapter>
```

```
  <kahaDB directory="{activemq.data}/kahadb"/>
```

```
</persistenceAdapter>
```

# Возможности и особенности Apache ActiveMQ

1. JVM Memory
2. Broker Memory
3. Prefetch limit
4. Producer flow control

# JVM Memory

```
# Set jvm memory configuration
if [ -z "$ACTIVEMQ_OPTS_MEMORY" ]; then
    ACTIVEMQ_OPTS_MEMORY="-Xms1G -Xmx1G"
fi
```

# Broker Memory

JVM Memory → Broker Memory → Broker features Memory



# Prefetch limit

1. Создать экземпляр класса `ActiveMQPrefetchPolicy` в `ActiveMQConnectionFactory` или `ActiveMQConnection`:
  - persistent queues (default value: 1000)
  - non-persistent queues (default value: 1000)
  - persistent topics (default value: 100)
  - non-persistent topics (default value: `Short.MAX_VALUE - 1`)

# Prefetch limit

2. При установлении соединения брокера:

```
tcp://localhost:61616?jms.prefetchPolicy.all=50
```

или

```
tcp://localhost:61616?jms.prefetchPolicy.queuePrefetch=1
```

# Prefetch limit

3. Настроить в опциях Destination:

```
queue =
```

```
new ActiveMQQueue("TEST.QUEUE?consumer.prefetchSize=10");
```

```
consumer = session.createConsumer(queue);
```

# Persistence Options

```
<persistenceAdapter>
```

```
<jdbcPersistenceAdapter dataDirectory="activemq-data" dataSource="#mysql-ds"/>
```

```
</persistenceAdapter>
```

```
<bean id="mysql-ds" class="org.apache.commons.dbcp.BasicDataSource" destroy-  
method="close">
```

```
  <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
```

```
  <property name="url"  
value="jdbc:mysql://localhost/activemq?relaxAutoCommit=true"/>
```

```
  <property name="username" value="activemq"/>
```

```
  <property name="password" value="activemq"/>
```

```
  <property name="poolPreparedStatements" value="true"/>
```

```
</bean>
```

# Кластеризация

1. Master/Slave (s) – хорошая надёжность
2. Сеть брокеров (Network of Brokers) – качественная. К преимуществам также относятся доступность, масштабируемость и надёжность.

# Сеть брокеров

## 1. Жёстко задать список URI:

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

```
<beans xmlns="http://activemq.org/config/1.0">
```

```
  <broker brokerName="receiver" persistent="false" useJmx="false">
```

```
    <networkConnectors>
```

```
      <networkConnector uri="static:(tcp://localhost:62001)"/>
```

```
    </networkConnectors>
```

```
    ...
```

```
  <transportConnectors>
```

```
    <transportConnector uri="tcp://localhost:62002"/>
```

```
  </transportConnectors>
```

```
</broker>
```

```
</beans>
```

# Сеть брокеров

2. Использовать групповую адресацию:

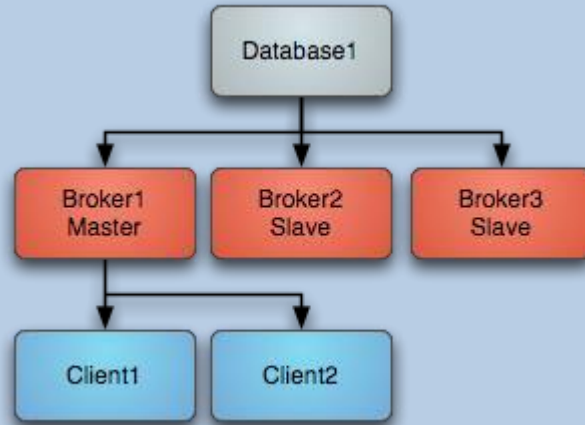
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://activemq.org/config/1.0">
  <broker name="sender" persistent="false" useJmx="false">
    <networkConnectors>
      <networkConnector uri="multicast://default"/>
    </networkConnectors>
    ...
  <transportConnectors>
    <transportConnector uri="tcp://localhost:0" discoveryUri="multicast://default"/>
  </transportConnectors>
</broker>
</beans>
```

# Master/Slave(s)

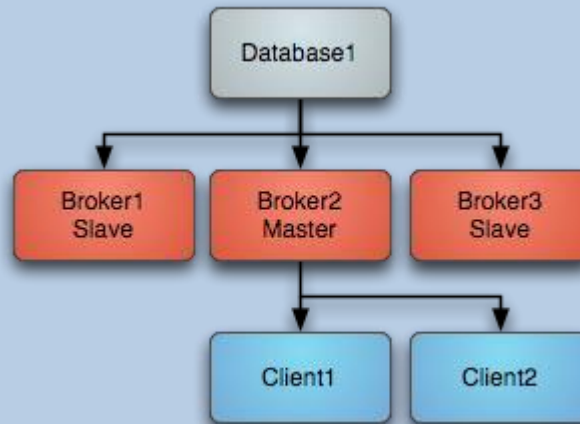
1. Shared File System Master Slave  
(распределённая файловая система SAN).
2. JDBC Master Slave (распределённая база данных).



# Master/Slave(s)



# Master/Slave(s)



# Shared File System Master Slave

```
<persistenceAdapter>
```

```
  <kahaDB directory="/sharedFileSystem/sharedBrokerData"/>
```

```
</persistenceAdapter>
```

или

```
<persistenceAdapter>
```

```
  <amqpPersistenceAdapter
```

```
directory="/sharedFileSystem/sharedBrokerData"/>
```

```
</persistenceAdapter>
```

# JDBC Master Slave

```
<persistenceAdapter>  
  <jdbcPersistenceAdapter dataDirectory="activemq-data" dataSource="#mysql-ds"/>  
</persistenceAdapter>
```

```
<bean id="mysql-ds" class="org.apache.commons.dbcp.BasicDataSource" destroy-  
method="close">  
  <property name="driverClassName" value="com.mysql.jdbc.Driver"/>  
  <property name="url"  
value="jdbc:mysql://localhost/activemq?relaxAutoCommit=true"/>  
  <property name="username" value="activemq"/>  
  <property name="password" value="activemq"/>  
  <property name="poolPreparedStatements" value="true"/>  
</bean>
```

## **Часть III.**

# **Пример конфигурации Apache ActiveMQ**

# Пример конфигурационного файла

## 1. Объявляем схемы:

```
<xmlns="http://www.springframework.org/schema/beans"
  xmlns:jdbc="http://www.springframework.org/schema/jdbc"
  xmlns:aop="http://www.springframework.org/schema/aop"
  xmlns:amq="http://activemq.apache.org/schema/core"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:jms="http://www.springframework.org/schema/jms"
  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.xsd
    http://www.springframework.org/schema/jms
    http://www.springframework.org/schema/jms/spring-jms.xsd
    http://activemq.apache.org/schema/core
    http://activemq.apache.org/schema/core/activemq-core.xsd">
```

# Пример конфигурационного файла

2. Создаём брокер:

```
<amq:broker id="ActiveMQBroker"  
dataDirectory="${activemq.data}" persistent="true"  
brokerName="ActiveMQBroker" >
```

# Пример конфигурационного файла

3. Создаём очереди:

```
<amq:queue id="SMSCompQueue" physicalName="SMSCompQueue"/>
```

...

```
<amq:queue id="MBloxNotificationQueue"  
physicalName="MBloxNotificationQueue"/>
```

```
<amq:queue id="MBloxDeactivationQueue"  
physicalName="MBloxDeactivationQueue"/>
```



# Пример конфигурационного файла

4. Создаём фабрику соединений:

```
<amq:connectionFactory id="jmsFactory"  
brokerURL="tcp://optit.mmq:61616?wireFormat.maxInactivityDura  
tion=-1"/>
```

# Пример конфигурационного файла

## 5. Создаём контейнер слушателей:

```
<bean id="jmsConsumerConnectionFactory"  
    class="org.apache.activemq.pool.PooledConnectionFactory"  
    depends-on="ActiveMQBroker"  
    p:ConnectionFactory-ref="jmsFactory"/>  
  
    <bean id="jmsMessageListenerPrepareConsumer"  
class="com.optit.messaging.jms.consumers.PrepareMessageConsumer"/>  
    <bean id="jmsMessageListenerSMSXActionalConsumer"  
class="com.optit.messaging.jms.consumers.SMSXActionalConsumer"/>  
    ...  
    <bean id="jmsMessageListenerSMSCompConsumer"  
class="com.optit.messaging.jms.consumers.SMSCompConsumer"/>
```

# Пример конфигурационного файла

```
<jms:listener-container container-type="default"
    connection-factory="jmsConsumerConnectionFactory"
    acknowledge="auto"
    concurrency="10" >

<jms:listener destination="PrepareQueue" ref="jmsMessageListenerPrepareConsumer"/>

<jms:listener destination="SMSXActionalQueue"
ref="jmsMessageListenerSMSXActionalConsumer"/>

...

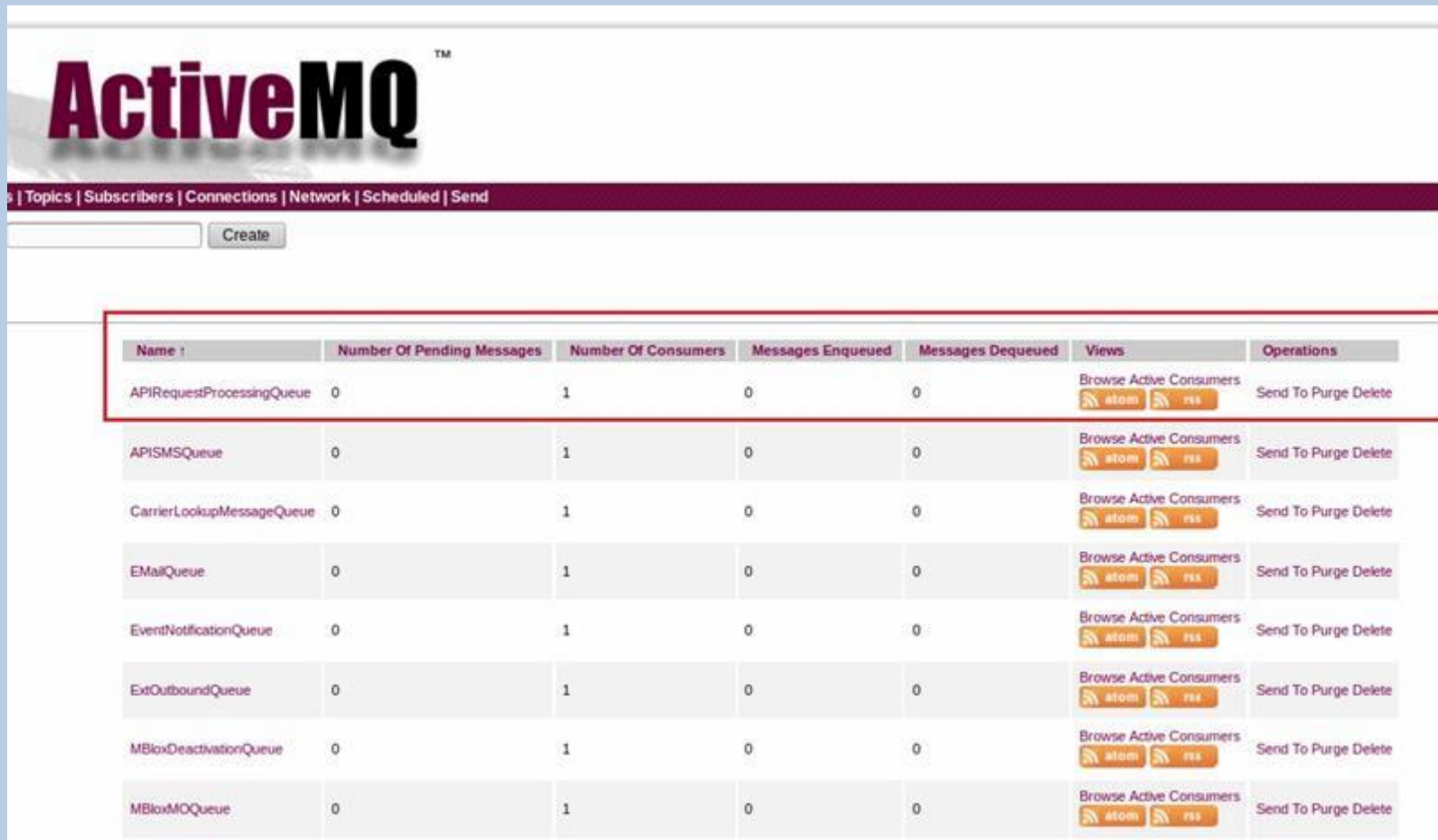
<jms:listener destination="SMSCompQueue" ref="jmsMessageListenerSMSCompConsumer"/>

</jms:listener-container>
```

















# Пример конфигурационного файла

1. `bin/activemq start xbean:conf/activemq_1.xml;`
2. `bin/activemq console xbean:conf/activemq_1.xml;`
3. `bin/activemq stop.`

































# Пример конфигурационного файла



The screenshot shows the ActiveMQ web console interface. At the top left is the ActiveMQ logo. Below it is a navigation bar with links for Topics, Subscribers, Connections, Network, Scheduled, and Send. There is a search input field and a 'Create' button. The main content area displays a table of queues with the following columns: Name, Number Of Pending Messages, Number Of Consumers, Messages Enqueued, Messages Dequeued, Views, and Operations. The first row of the table is highlighted with a red border.

Name	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued	Views	Operations
APIRequestProcessingQueue	0	1	0	0	Browse Active Consumers  	Send To Purge Delete
APISMSQueue	0	1	0	0	Browse Active Consumers  	Send To Purge Delete
CarrierLookupMessageQueue	0	1	0	0	Browse Active Consumers  	Send To Purge Delete
EMailQueue	0	1	0	0	Browse Active Consumers  	Send To Purge Delete
EventNotificationQueue	0	1	0	0	Browse Active Consumers  	Send To Purge Delete
ExtOutboundQueue	0	1	0	0	Browse Active Consumers  	Send To Purge Delete
MBloxDeactivationQueue	0	1	0	0	Browse Active Consumers  	Send To Purge Delete
MBloxMOQueue	0	1	0	0	Browse Active Consumers  	Send To Purge Delete

# Пример конфигурационного файла

Name	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued	Views	Operations
APIRequestProcessingQueue	0	1	0	0	Browse Active Consumers  	Send To Purge Delete
APISMSQueue	0	1	0	0	Browse Active Consumers  	Send To Purge Delete
CarrierLookupMessageQueue	0	1	0	0	Browse Active Consumers  	Send To Purge Delete
EMailQueue	0	1	0	0	Browse Active Consumers  	Send To Purge Delete
EventNotificationQueue	0	1	0	0	Browse Active Consumers  	Send To Purge Delete
ExtOutboundQueue	0	1	0	0	Browse Active Consumers  	Send To Purge Delete
MBloxDeactivationQueue	0	1	0	0	Atom 1.0 consumers  	Send To Purge Delete
MBloxMOQueue	0	1	0	0	Browse Active Consumers  	Send To Purge Delete
MBloxNotificationQueue	0	1	0	0	Browse Active Consumers  	Send To Purge Delete
MBloxPrepareQueue	0	2	1	1	Browse Active Consumers  	Send To Purge Delete
MotricityMOQueue	0	1	0	0	Browse Active Consumers  	Send To Purge Delete
MotricityNotificationQueue	0	1	0	0	Browse Active Consumers  	Send To Purge Delete
MotricityPrepareQueue	0	1	0	0	Browse Active Consumers  	Send To Purge Delete
OpenMarketMOQueue	0	1	0	0	Browse Active Consumers  	Send To Purge Delete
OpenMarketPrepareQueue	0	1	0	0	Browse Active Consumers  	Send To Purge Delete
PrepareQueue	0	2	1	1	Browse Active Consumers  	Send To Purge Delete

# Список используемых ИСТОЧНИКОВ

1. <https://onedevolver.ru/article?id=12>
2. <http://activemq.apache.org/>

**Спасибо за  
внимание!**