ActiveMQ入门指南

1 ActiveMQ简介

一定要看官网的介绍 https://activemq.apache.org/

从官网介绍能很好地获知ActiveMQ的用处：

Apache ActiveMQ™ is the most popular open source, multi-protocol, Java-based messaging server. It supports industry standard protocols so users get the benefits of client choices across a broad range of languages and platforms. Connectivity from C, C++, Python, .Net, and more is available. Integrate your multi-platform applications using the ubiquitous **AMQP** protocol. Exchange messages between your web applications using **STOMP** over websockets. Manage your IoT devices using **MQTT**. Support your existing **JMS** infrastructure and beyond. ActiveMQ offers the power and flexibility to support any messaging use-case.

一款用Java开发的、开源的、支持多种协议的、非常流行的消息服务（消息中间件）。因为它的支持多协议，支持工业标准的协议，所以我们可以跨平台、跨语言来使用它。

- 用AMQP工业标准协议进行多平台应用集成；
- Web应用可基于websocket用STOMP协议与ActiveMQ直接交互
- 物联网设备用MQTT协议
- 基于JMS的已有基础设施也支持
- 还有更多...

当前有两个版本：ActiveMQ 5 "Classic" 和 ActiveMQ Artemis (下一代版本)。

我们学习 ActiveMQ 5 "Classic" 经典版。

有兴趣可以了解 ActiveMQ Artemis。

2 ActiveMQ 官方学习资源介绍

ActiveMQ作为一个老牌的消息中间件，其提供了详细的官方文档，强烈大家从官网文档来进行学习，及工作中新需求解决办法的查找。

官网文档入口：https://activemq.apache.org/components/classic/documentation

- Overview
  - New Features
  - Getting Started ← 快速开始，安装看这里
  - FAQ ← 快速掌握某用法、配置看这里
  - Articles ← 一些某些人写有用的文章，如MQ比较类的，使用类的，值得去看看都有些什么
  - Books
  - License
  - Download
  - Latest Javadoc ← 可可去浏览浏览，了解了解分包情况
- Community
- Features ← 特性介绍，一定要去学习的
- Connectivity ← 各种互联的说明，一定要去学习
- Using ActiveMQ 5 ← 使用说明，一定要去学习
- Tools
- Support
- Developers ← 想成为它的开发者，了解这里
- Tests ← 想对ActiveMQ来场测试，看这里
  - Maven2 Performance Plugin
  - Benchmark Tests
  - JMeter System Tests
  - JMeter Performance Tests
  - Integration Tests

3 Linux 安装指南

各种环境下安装，请参考官网文档：https://activemq.apache.org/getting-started

下面说明一下linux上的安装。

1 环境准备

- 虚拟机软件：Oracle VM VirtualBox

  下载地址：https://www.virtualbox.org/wiki/Downloads
- Linux: Centos 7 CentOS-7-x86_64-Minimal-1810.iso

  阿里云镜像下载地址：https://mirrors.aliyun.com/centos/7.6.1810/isos/x86_64/
- Jdk 8 jdk-8u221-linux-x64.rpm

2 安装

2.1 下载安装包：

https://activemq.apache.org/components/classic/download/

Windows	apache-activemq-5.15.9-bin.zip	SHA512	GPG Signature
Unix/Linux/Cygwin	apache-activemq-5.15.9-bin.tar.gz	SHA512	GPG Signature
Source Code Distribution:	activemq-parent-5.15.9-source-release.zip	SHA512	GPG Signature

也可在linux机器上直接下载：

```
wget -c http://mirror.bit.edu.cn/apache/activemq/5.15.9/apache-activemq-5.15.9-bin.tar.gz
```

2.2 安装

1. 创建安装目录

```
mkdir /usr/activemq
```

2. 解压安装包到安装目录

```
tar -zxvf apache-activemq-5.15.9-bin.tar.gz -C /usr/activemq
```

3. 为方便配置时书写，创建软链接

```
ln -s /usr/activemq/apache-activemq-5.15.9 /usr/activemq/latest
```

4. 熟悉activemq的目录构成：

2.3 启停

1. 启动

```
cd /usr/activemq/latest/bin
```

作为前台进程启动

```
./activemq console
```

作为后台守护进程启动

```
./activemq start
```

启动输出

```
INFO: Loading '/usr/activemq/apache-activemq-5.15.9//bin/env'
INFO: Using java '/usr/bin/java'
INFO: Starting - inspect logfiles specified in logging.properties and
log4j.properties to get details
INFO: pidfile created : '/usr/activemq/apache-activemq-
5.15.9//data/activemq.pid' (pid '14887')
```

2. 启动成功检测：

   访问管理控制台：http://ip:8161/admin **如果防火墙阻止了，请看下面防火墙开发端口**

   ActiveMQ的管理页面默认开启了身份校验：

   账号：admin

   密码：admin

   或在启动Console 或 日志文件（data/activemq.log）中看到日志输出：

```
Apache ActiveMQ 5.15.9 (localhost, ID:ntbk11111-50816-1428933306116-0:1)
started | org.apache.activemq.broker.BrokerService | main
```

   或用 jps命令查看

```
[root@localhost latest]# jps
25778 activemq.jar
25805 Jps
```

3. 停止

```
./activemq stop
```

4. 了解activemq 命令的用法（快速了解一下）：

```
./activemq
```

浏览用法说明信息。也可以查看官网连接：https://activemq.apache.org/unix-shell-script

```
Usage: ./activemq [--extdir <dir>] [task] [task-options] [task data]

Tasks:
    browse                  - Display selected messages in a specified
destination.
    bstat                   - Performs a predefined query that displays
useful statistics regarding the specified broker
    consumer                - Receives messages from the broker
    create                  - Creates a runnable broker instance in the
specified path.
    decrypt                 - Decrypts given text
    dstat                   - Performs a predefined query that displays
useful tabular statistics regarding the specified destination type
    encrypt                 - Encrypts given text
    export                  - Exports a stopped brokers data files to an
archive file
    list                    - Lists all available brokers in the specified
JMX context
    producer                - Sends messages to the broker
    purge                   - Delete selected destination's messages that
matches the message selector
    query                   - Display selected broker component's
attributes and statistics.
    start                   - Creates and starts a broker using a
configuration file, or a broker URI.
    stop                    - Stops a running broker specified by the
broker name.

Task Options (Options specific to each task):
    --extdir <dir>  - Add the jar files in the directory to the classpath.
    --version       - Display the version information.
    -h,-?,--help    - Display this help information. To display task
specific help, use Main [task] -h,-?,--help

Task Data:
    - Information needed by each specific task.

JMX system property options:
    -Dactivemq.jmx.url=<jmx service uri> (default is:
'service:jmx:rmi:///jndi/rmi://localhost:1099/jmxrmi')
    -Dactivemq.jmx.user=<user name>
    -Dactivemq.jmx.password=<password>


Tasks provided by the sysv init script:
    kill            - terminate instance in a drastic way by sending SIGKILL
    restart         - stop running instance (if there is one), start new
instance
    console         - start broker in foreground, useful for debugging
purposes
    status          - check if activemq process is running

Configuration of this script:
    The configuration of this script is read from the following files:
```

```
    /etc/default/activemq /root/.activemqrc /usr/activemq/apache-activemq-
5.15.9//bin/env
    This script searches for the files in the listed order and reads the
first available file.
    Modify /usr/activemq/apache-activemq-5.15.9//bin/env or create a copy of
that file on a suitable location.
    To use additional configurations for running multiple instances on the
same operating system
    rename or symlink script to a name matching to activemq-instance-
<INSTANCENAME>.
    This changes the configuration location to /etc/default/activemq-
instance-<INSTANCENAME> and
    $HOME/.activemqrc-instance-<INSTANCENAME>.
```

2.4 防火墙开放ActiveMQ的端口

```
#web管理端口默认为8161，通讯端口默认为61616
firewall-cmd --zone=public --add-port=8161/tcp --permanent
firewall-cmd --zone=public --add-port=61616/tcp --permanent
```

重启防火墙

```
systemctl restart firewalld.service
```

学习用，可以直接关闭防火墙

```
systemctl stop firewalld
systemctl disable firewalld
```

3 Linux服务安装

参考链接：https://activemq.apache.org/unix-shell-script

1. 以普通用户activemq 身份来运行

   ```
   useradd activemq
   chown -R activemq:users /usr/activemq
   ```

2. 创建全局默认的配置文件，并配置activemq

   ```
   cp /usr/activemq/latest/bin/env /etc/default/activemq
   sed -i '~s/^ACTIVEMQ\_USER=""/ACTIVEMQ\_USER="activemq"/'
   /etc/default/activemq
   ```

   编辑activemq配置文件，进行如下配置（**生产环境需要考虑配置**）

   - Configure the java heap to a size suitable to your system environment and usage
   - Consider to move the folders “data”, “tmp” and “conf” out of the installation path

```
vim /etc/default/activemq
```

配置内容如下所示：

```
# Active MQ installation dirs
# ACTIVEMQ_HOME="<Installationdir>/"
# ACTIVEMQ_BASE="$ACTIVEMQ_HOME"
# ACTIVEMQ_CONF="$ACTIVEMQ_BASE/conf"
# ACTIVEMQ_DATA="$ACTIVEMQ_BASE/data"
# ACTIVEMQ_TMP="$ACTIVEMQ_BASE/tmp"

# Set jvm memory configuration (minimal/maximum amount of memory)
ACTIVEMQ_OPTS_MEMORY="-Xms64M -Xmx1G"
```

修改权限模式

```
chmod 644 /etc/default/activemq
```

3. 安装启动脚本

```
ln -snf  /usr/activemq/latest/bin/activemq /etc/init.d/activemq
```

4. 激活启动服务

```
# RHEL
chkconfig --add activemq
chkconfig activemq on
```

或

```
systemctl enable activemq
```

5. 手动启动服务

```
systemctl start activemq
```

4 多实例运行【了解】

参考链接：https://activemq.apache.org/unix-shell-script

To use additional configurations for running multiple instances on the same operating system rename or symlink script to a name matching to `activemq-instance-<INSTANCENAME>`. This changes the configuration location to `/etc/default/activemq-instance-<INSTANCENAME>` and `$HOME/.activemqrc-instance-<INSTANCENAME>`. Configuration files in /etc have higher precedence.

Example procedure suitable to the procedure “Running activemq as a unix daemon”

Example

```
mkdir /srv/activemq/instance1
cp -av /srv/activemq/current/conf/ /srv/activemq/instance1/
mkdir /srv/activemq/instance1/{data,tmp}
ln -snf  /srv/activemq/current/bin/activemq /etc/init.d/activemq-instance-test1
cp /srv/activemq/install/bin/env /etc/default/activemq-instance-instance-test1
```

Modify the configuration variables in /etc/default/activemq-instance-instance-test1

```
ACTIVEMQ_HOME="/srv/activemq/current/"
ACTIVEMQ_CONF="/srv/activemq/instance1/conf"
ACTIVEMQ_DATA="/srv/activemq/instance1/data"
ACTIVEMQ_TMP="/srv/activemq/instance1/tmp"
```

Control the instance

```
/etc/init.d/activemq-instance1 start|stop|restart|console|....
```

> **Hint**
>
> If you are using multiple instances you can only add the main instance to the automatic system start using (“update-rc.d” or “chkconfig”) because the LSB Header “Provides” needs to be uniq.

4 快速上手使用

4.1 普通JAVA应用中使用ActiveMQ

1. 引入activemq-all.jar

```
<dependency>
    <groupId>org.apache.activemq</groupId>
    <artifactId>activemq-all</artifactId>
    <version>5.15.9</version>
</dependency>
```

2. 编写消息生产者Producer

```
package com.study.activemq.le1.helloworld.queue;

import javax.jms.Connection;
import javax.jms.DeliveryMode;
import javax.jms.Destination;
import javax.jms.JMSException;
import javax.jms.MessageProducer;
import javax.jms.Session;
```

```java
import javax.jms.TextMessage;

import org.apache.activemq.ActiveMQConnectionFactory;

/**
 * 简单生产者
 */
public class Producer {
    public static void main(String[] args) {
        new ProducerThread("tcp://mq.study.com:61616", "queue1").start();
    }

    static class ProducerThread extends Thread {
        String brokerUrl;
        String destinationUrl;

        public ProducerThread(String brokerUrl, String destinationUrl) {
            this.brokerUrl = brokerUrl;
            this.destinationUrl = destinationUrl;
        }

        @Override
        public void run() {
            ActiveMQConnectionFactory connectionFactory;
            Connection conn;
            Session session;

            try {
                // 1、创建连接工厂
                connectionFactory = new
ActiveMQConnectionFactory(brokerUrl);

                // 2、创建连接
                conn = connectionFactory.createConnection();
                conn.start(); // 一定要start

                // 3、创建会话（可以创建一个或者多个session）
                session = conn.createSession(false,
Session.AUTO_ACKNOWLEDGE);

                // 4、创建消息发送目标 (Topic or Queue)
                Destination destination =
session.createQueue(destinationUrl);

                // 5、用目的地创建消息生产者
                MessageProducer producer =
session.createProducer(destination);
                // 设置递送模式(持久化 / 不持久化)
                producer.setDeliveryMode(DeliveryMode.PERSISTENT);

                // 6、创建一条文本消息
                String text = "Hello world! From: " +
Thread.currentThread().getName() + " : "
                        + System.currentTimeMillis();
                TextMessage message = session.createTextMessage(text);

                // 7、通过producer 发送消息
                System.out.println("Sent message: " + text);
```

```
                producer.send(message);

                // 8、 清理、关闭连接
                session.close();
                conn.close();
            } catch (JMSException e) {
                e.printStackTrace();
            }
        }
    }
}
```

3. 编写消息消费者Consumer

```
package com.study.activemq.le1.helloworld.queue;

import javax.jms.Connection;
import javax.jms.Destination;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageConsumer;
import javax.jms.Session;
import javax.jms.TextMessage;

import org.apache.activemq.ActiveMQConnectionFactory;

/**
 * 简单消费者
 */
// http://activemq.apache.org/consumer-features.html
public class Consumer {
    public static void main(String[] args) {
        new ConsumerThread("tcp://mq.study.com:61616", "queue1").start();
        new ConsumerThread("tcp://mq.study.com:61616", "queue1").start();
    }
}

class ConsumerThread extends Thread {

    String brokerUrl;
    String destinationUrl;

    public ConsumerThread(String brokerUrl, String destinationUrl) {
        this.brokerUrl = brokerUrl;
        this.destinationUrl = destinationUrl;
    }

    @Override
    public void run() {
        ActiveMQConnectionFactory connectionFactory;
        Connection conn;
        Session session;
        MessageConsumer consumer;
```

```
        try {
            // brokerURL
            // http://activemq.apache.org/connection-configuration-uri.html
            // 1、创建连接工厂
            connectionFactory = new
ActiveMQConnectionFactory(this.brokerUrl);

            // 2、创建连接对象
            conn = connectionFactory.createConnection();
            conn.start(); // 一定要启动

            // 3、创建会话（可以创建一个或者多个session）
            session = conn.createSession(false, Session.AUTO_ACKNOWLEDGE);

            // 4、创建消息消费目标(Topic or Queue)
            Destination destination = session.createQueue(destinationUrl);

            // 5、创建消息消费者 http://activemq.apache.org/destination-
options.html
            consumer = session.createConsumer(destination);

            // 6、接收消息(没有消息就持续等待)
            Message message = consumer.receive();
            if (message instanceof TextMessage) {
                System.out.println("收到文本消息：" + ((TextMessage)
message).getText());
            } else {
                System.out.println(message);
            }

            consumer.close();
            session.close();
            conn.close();
        } catch (JMSException e) {
            e.printStackTrace();
        }
    }
}
```

4.2 spring boot 中使用ActiveMQ

学习连接：

https://docs.spring.io/spring-boot/docs/2.1.6.RELEASE/reference/html/boot-features-messaging.html

https://spring.io/guides/gs/messaging-jms/

1. 引入starter : spring-boot-starter-activemq

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-activemq</artifactId>
</dependency>
```

2. 配置activemq broker连接参数 (application.yml)

```
spring:
  activemq:
    broker-url: tcp://mq.study.com:61616
    #user: admin
    #password: secret
```

可配置参数有spring.activemq.* ， spring.jms.* 。

3. 发送消息

a POJO class

```
package com.study.activemq.le1.spring;

public class Email {

    private String to;
    private String body;

    public Email() {
    }

    public Email(String to, String body) {
        this.to = to;
        this.body = body;
    }

    public String getTo() {
        return to;
    }

    public void setTo(String to) {
        this.to = to;
    }

    public String getBody() {
        return body;
    }

    public void setBody(String body) {
        this.body = body;
    }

    @Override
    public String toString() {
        return String.format("Email{to=%s, body=%s}", getTo(), getBody());
    }
```

```
}
```

Producer

```
package com.study.activemq.le1.spring;

import javax.annotation.PostConstruct;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.jms.core.JmsTemplate;
import
org.springframework.jms.support.converter.MappingJackson2MessageConverter;
import org.springframework.jms.support.converter.MessageConverter;
import org.springframework.jms.support.converter.MessageType;

@SpringBootApplication
public class Producer {

    @Bean // Serialize message content to json using TextMessage
    public MessageConverter jacksonJmsMessageConverter() {
        MappingJackson2MessageConverter converter = new
MappingJackson2MessageConverter();
        converter.setTargetType(MessageType.TEXT);
        converter.setTypeIdPropertyName("_type");
        return converter;
    }

    @Autowired
    private JmsTemplate jmsTemplate;

    @PostConstruct
    public void sendMessage() {
        // Send a message with a POJO - the template reuse the message
converter
        System.out.println("Sending an email message.");
        jmsTemplate.convertAndSend("mailbox", new Email("info@example.com",
"Hello"));
    }

    public static void main(String[] args) {
        SpringApplication.run(Producer.class, args);
    }
}
```

4. 消费消息

```
package com.study.activemq.le1.spring;

import org.springframework.boot.SpringApplication;
```

```
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.jms.annotation.JmsListener;

@SpringBootApplication
public class Consumer {

    @JmsListener(destination = "mailbox")
    public void receive(Email email) {
        System.out.println("Received <" + email + ">");
    }

    public static void main(String[] args) {
        SpringApplication.run(Consumer.class, args);
    }
}
```

关于MessageConverter

The default `MessageConverter` is able to convert only basic types (such as `String`, `Map`, `Serializable`) and our `Email` is not `Serializable` on purpose. We want to use Jackson and serialize the content to json in text format (i.e. as a `TextMessage`). Spring Boot will detect the presence of a `MessageConverter` and will associate it to both the default `JmsTemplate` and any `JmsListenerContainerFactory` created by `DefaultJmsListenerContainerFactoryConfigurer`.

5 配置指南

5.1 conf 目录了解

5.2 管理控制台配置

管理控制台使用Jetty做为web服务器。

配置文件为：conf/

5.3 ActiveMQ配置

https://activemq.apache.org/xml-configuration

5.4 安全配置

https://activemq.apache.org/security

6 使用指南

brokerUrl 连接参数说明

http://activemq.apache.org/connection-configuration-uri.html

Destination Features

http://activemq.apache.org/destination-features

Consumer Features

http://activemq.apache.org/consumer-features

JmsPoolConnectionFactory

```
<dependency>
    <groupId>org.messaginghub</groupId>
    <artifactId>pooled-jms</artifactId>
    <version>1.0.6</version>
</dependency>
```

```
spring.activemq.pool.enabled=true
spring.activemq.pool.max-connections=50
```