



Prácticas de laboratorio JMS: Java Message Service (2 sesiones)

Concurrencia y Sistemas Distribuidos

Introducción

Esta práctica tiene como objetivo desarrollar una aplicación simple utilizando el servicio de mensajería **Java Message Service** (JMS). En particular, se trabajará con una aplicación de chat que tiene la misma interfaz gráfica que en la práctica anterior, aunque usa JMS, y se desarrollará un ChatRobot para ella.

Dado el carácter introductorio de esta práctica, se tratarán aspectos básicos tales como la conexión con el proveedor JMS y el envío y recepción de mensajes. Otros aspectos que serían fundamentales para el desarrollo de una aplicación real, tales como seguridad y tolerancia ante fallos, se han dejado intencionadamente al margen en aras de la simplicidad.

La duración estimada de esta práctica es de dos semanas. Cada semana debe asistir a una sesión de laboratorio donde podrá comentar con el profesorado de prácticas sus progresos y resolver las dudas que le surjan. Tenga en cuenta que necesitará destinar algo de tiempo de su trabajo personal para concluir la práctica.

Esta practica está preparada para realizarla desde los sistemas **Linux** de Polilabs del DSIC.

Actividad 1

En esta actividad se pretende instalar, configurar y ejecutar un proveedor JMS. El proveedor JMS que se instalará es Apache ActiveMQ Artemis (simplemente *Artemis* en el resto de este documento).

Artemis es un conjunto de librerías y aplicaciones desarrolladas en Java que se distribuye como un archivo comprimido. Para esta práctica, ofrecemos a los alumnos un script de instalación (*install-artemis-csd.sh*) que facilita dicho proceso. En el anexo se explica con detalle el contenido del script, así como una breve descripción de los comandos de Artemis que nos van a ser necesarios para nuestra práctica.

Artemis está disponible para su descarga en la página web de Apache ActiveMQ¹. Para evitar problemas de cuotas de espacio en disco, la distribución de Artemis utilizada está disponible en la carpeta asigDSIC, común a todos los usuarios. Accederemos a dicha unidad a través de Polilabs. En el caso de que un alumno desee realizar una instalación de Artemis en su ordenador personal, en el Anexo de este documento se explica cómo proceder.

Instrucciones

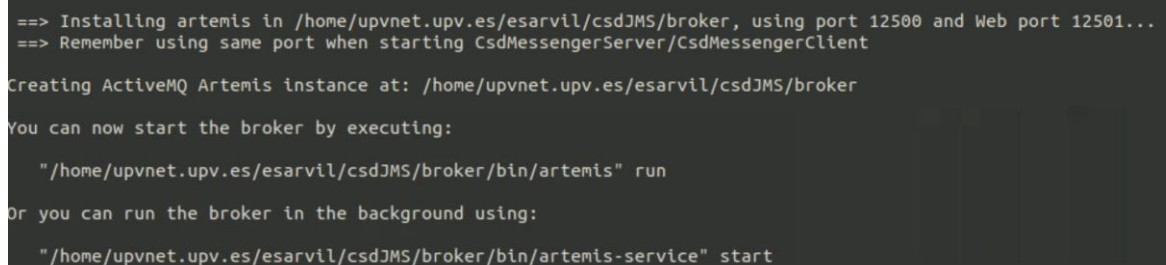
1. Entre en DSIC-Linux de Polilabs. Abra un terminal y cree un directorio llamado `csdJMS` en su home. En este directorio será donde se instale el broker de Artemis.

```
$ cd
$ mkdir csdJMS
```

2. Sitúese ahora en el directorio `asigDSIC/ETSINF/csd/jms/artemis/bin` y ejecute el script de instalación `install-artemis-csd.sh`. Como parámetros de dicho script, deberá indicar un **número de puerto** y el directorio en el que instalar el broker de Artemis (que en este caso será el directorio `broker`, dentro del directorio `csdJMS` creado en el paso anterior. `~` indica el directorio home). Respecto al número de puerto, puede utilizar el puerto 12500, como se muestra a continuación:

```
$ cd
$ cd asigDSIC/ETSINF/csd/jms/artemis/bin
$ bash install-artemis-csd.sh 12500 ~/csdJMS/broker
```

Si la instalación es correcta, se mostrará en el terminal un mensaje similar al siguiente (siendo `esarvil` el nombre de usuario correspondiente, según cada alumno).



```
==> Installing artemis in /home/upvnet.upv.es/esarvil/csdJMS/broker, using port 12500 and Web port 12501...
==> Remember using same port when starting CsdMessengerServer/CsdMessengerClient

Creating ActiveMQ Artemis instance at: /home/upvnet.upv.es/esarvil/csdJMS/broker

You can now start the broker by executing:

    "/home/upvnet.upv.es/esarvil/csdJMS/broker/bin/artemis" run

Or you can run the broker in the background using:

    "/home/upvnet.upv.es/esarvil/csdJMS/broker/bin/artemis-service" start
```

¹ <https://activemq.apache.org/components/artemis/download/>

- Para lanzar el broker Artemis, sitúese en su home, vaya al directorio `csdJMS/broker/bin` y ejecute la orden `artemis run`.

Si todo ha funcionado de forma correcta, verá en pantalla el bróker en ejecución:

Y entre los muchos mensajes que aparecen, al final se le indicará que la consola está disponible en el puerto 12501.

Recuerde que, una vez instalado Artemis, desde el directorio `~/csdJMS/broker/bin`, puede iniciarlo con la siguiente orden:

y puede detenerlo tecleando Ctrl-C en la misma terminal.

Nota: si desea disponer de la instalación de Artemis de forma permanente, puede instalarlo también en su disco W. De esta forma, no necesitará reinstalarlo cada vez que desee utilizarlo, aunque el acceso a la consola web será mucho más lento (debido a la configuración y características del disco W).

Al lanzar a ejecución el bróker, si todo ha funcionado de forma correcta, nos aparecerá en pantalla que el servidor Artemis está en funcionamiento, así como un conjunto de mensajes del tipo:

```

2021-09-08 16:21:02,786 INFO [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
2021-09-08 16:21:02,787 INFO [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.17.0 [0.0.0, nodeID=f3350612-10af-11ec-abd6-0050562302e9]
2021-09-08 16:21:04,700 INFO [org.apache.activemq.hawtio.branding.PluginContextListener] Initialized activemq-branding plugin
2021-09-08 16:21:06,383 INFO [org.apache.activemq.hawtio.plugin.PluginContextListener] Initialized artemis-plugin plugin
2021-09-08 16:21:20,889 INFO [io.hawt.HawtioContextListener] Initialising hawtio services
2021-09-08 16:21:20,998 INFO [io.hawt.system.ConfigManager] Configuration will be discovered via system properties
2021-09-08 16:21:21,001 INFO [io.hawt.jmx.JmxTreeWatcher] Welcome to Hawtio 2.11.0
2021-09-08 16:21:21,008 INFO [io.hawt.web.auth.AuthenticationConfiguration] Starting hawtio authentication filter, JAAS realm: "activemq" and authorized role(s): "amq" role principal classes: "org.apache.activemq.artemis.spi.core.security.jaas.RolePrincipal"
2021-09-08 16:21:21,035 INFO [io.hawt.web.proxy.ProxyServlet] Proxy servlet is disabled
2021-09-08 16:21:21,035 INFO [io.hawt.web.servlets.JolokiaConfiguredAgentServlet] Jolokia overridden property: [key=policyLocation, value=file:/home/upvnet.upv.es/esarvil/W/docencia/apache-artemis-2.17.0/bin/miArtemis/etc/jolokia-access.xml]
2021-09-08 16:21:21,130 INFO [org.apache.activemq.artemis] AMQ241001: HTTP Server started at http://localhost:12501
2021-09-08 16:21:21,130 INFO [org.apache.activemq.artemis] AMQ241002: Artemis Jolokia REST API available at http://localhost:12501/console/jolokia
2021-09-08 16:21:21,130 INFO [org.apache.activemq.artemis] AMQ241004: Artemis Console available at http://localhost:12501/console

```

Estos mensajes nos indican, entre otros aspectos, que el servidor Artemis está activo y disponible en <http://localhost:12501> (es decir, en el NÚMERO DE PUERTO + 1, donde NÚMERO DE PUERTO es el que habíamos indicado en la instalación) y que la consola de Artemis está disponible en <http://localhost:12501/console>.

Nota: Si ocurriera que el puerto indicado ya estuviera ocupado (por ejemplo, porque varios usuarios están utilizando la misma máquina virtual y uno de ellos ya ha creado y lanzado a ejecución su broker Artemis), se obtendrán mensajes de error del siguiente tipo:

```
java.io.IOException: Failed to bind to localhost/127.0.0.1:12501
```

```
Caused by: java.net.BindException: La dirección ya se está usando
```

Para solucionar este error, borre el contenido del directorio broker que ha creado previamente y repita los pasos 2 y 3 utilizando otro número de puerto (por ejemplo el 12510, o bien el 12520, etc.) En caso de dudas, consulte a su profesor.

Actividad 2

En esta actividad se comprobará el funcionamiento de la aplicación de Chat implementada en Java Message Service, disponible en DistributedChatJMS.

Conceptos previos

DistributedChatJMS hace uso de la misma interfaz gráfica que en la práctica anterior. Disponemos ahora de una aplicación "servidor" (ChatServerJMS), que se encarga de los usuarios conectados y los canales disponibles (y podría servirnos para controlar la autenticación de los usuarios, aunque esto ahora no se ha tenido en cuenta); y una aplicación cliente (ChatClientJMS), que utiliza una interfaz gráfica para permitir a un usuario conectarse a la aplicación de Chat, unirse a canales de chat y enviar mensajes a los usuarios de ese canal (tanto mensajes al canal, como mensajes privados a un usuario).

En la aplicación propuesta, en realidad no sería necesario disponer de un servidor ChatServerJMS, si los clientes fueran quienes solicitaran al broker Artemis la creación de sus propias colas y los temas de los canales ya estuvieran inicialmente creados. Sin embargo, hemos incluido el servidor ChatServerJMS para que se encargue de gestionar los usuarios conectados y los canales disponibles. De esta manera evitamos que usuarios diferentes se conecten con nicks duplicados y podemos llevar el seguimiento de cuántos usuarios se conectan a cada canal.

Cuando se lanza el servidor ChatServerJMS por primera vez, se encargará de crear los canales "Linux", "Spain" y "Friends". Y solicitará a Artemis la creación de una cola de tipo multicast por cada canal creado, a la que se suscribirá, para así poder recibir los mensajes LEAVE y poder mantener actualizada

la lista de suscriptores al canal. Cada una de estas colas se denomina `channel-NAME` (siendo NAME el nombre del canal correspondiente).

Nota: En Artemis no existe el concepto de "tema" o Topic. Su concepto más equivalente es una dirección MULTICAST. Estas direcciones multicasts pueden ser usadas como tema o topic JMS cuando se utiliza Artemis como proveedor JMS.

El servidor ChatServerJMS solicitará también a Artemis la creación de la cola `ChatServer`, de tipo anycast (equivalente al destino de tipo "queue" de JMS), a través de la cual recibirá los mensajes de los usuarios.

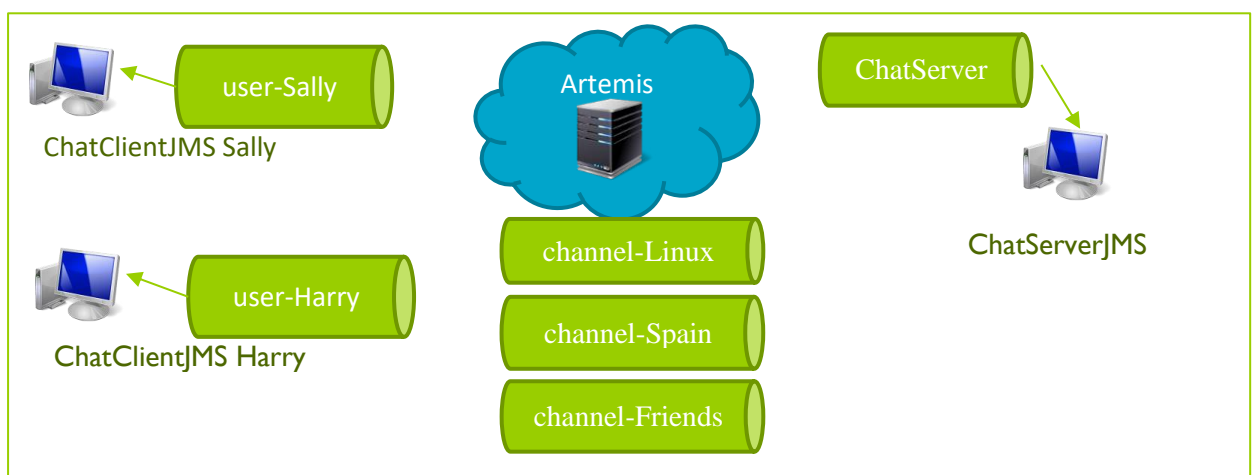
Por su parte, cuando un usuario se conecta por primera vez, nuestro servidor ChatServerJMS ordena a "artemis" que cree una cola para el usuario (de tipo anycast), con el nombre `user-USERNAME` (siendo USERNAME el nombre concreto del usuario). Dicha cola existirá de forma permanente, hasta que se reinstale Artemis.

La principal ventaja de tener un servidor dedicado ChatServerJMS es la independencia del proveedor en la gestión de usuarios, de modo que el servidor ChatServerJMS controla cuántos usuarios están conectados en cada momento a los canales.

En un momento dado solo debe haber activa una aplicación ChatServerJMS, pero pueden estar activas varias ChatClientJMS, una por cada usuario conectado al sistema.

Las colas "user-USERNAME" son gestionadas por ChatServerJMS bajo demanda, de forma que ChatServerJMS solicita al broker Artemis su creación la primera vez que un usuario se conecta al sistema.

La siguiente figura resume los componentes previamente citados, asumiendo los usuarios Sally y Harry:



Instrucciones

1. Descargue el fichero ***DistributedChatJMS.tgz*** proporcionado para la práctica (disponible en el Lessons de la práctica 5 en el sitio PoliformaT de la asignatura) en su unidad W, descomprímalo y acceda al directorio recién creado. En concreto, realice:

```
$ tar xvf DistributedChatJMS.tgz
$ cd DistributedChatJMS
```

2. En el directorio DistributedChatJMS verá que se encuentran los ficheros ChatClientJMS.java y ChatServerJMS.java, correspondientes a la aplicación proporcionada en esta práctica. Compile estos ficheros:

```
$ javac -cp lib/*:. *.java
```

3. Lance a ejecución la aplicación **ChatServerJMS**, indicando como parámetro de entrada la dirección donde está escuchando el broker Artemis (en nuestro caso, en localhost:12500).

```
$ java -cp lib/*:. ChatServerJMS url=localhost:12500
```

Nota: A continuación se muestran los parámetros permitidos para la aplicación ChatServerJMS

```
USAGE HELP

  java ChatServerJMS [url=...]

USAGE -- valid arguments:

  url = <URL where artemis is running>

Examples:
  java -cp lib/*:. ChatServerJMS url=localhost:9000
  java -cp lib/*:. ChatServerJMS url=158.42.185.162:12500
```

4. En otro terminal, lance a ejecución la aplicación ChatClientJMS, indicando como parámetro de entrada la dirección donde está escuchando el broker Artemis (en nuestro caso, en localhost:12500).

```
$ java -cp lib/*:. ChatClientJMS url=localhost:12500
```

5. Puede lanzar a ejecución varias aplicaciones ChatClientJMS. Recomendamos que los lance desde terminales distintos.

Nota: A continuación se muestran los parámetros permitidos para la aplicación ChatClientJMS

```
USAGE HELP

  java ChatClientJMS [url=...] [nick=...] [channel=...]

USAGE -- valid arguments:

  url = <URL where artemis is running>
  nick = <User name to use when connecting to a ChatServer>
  channel = <Channel to auto-join when program starts>

Examples:
  java -cp ./lib/* ChatClientJMS url=127.0.0.1:4000 nick=troll channel=Linux
  java -cp ./lib/* ChatClientJMS nick=pep
```

Comprobación

Al lanzar ChatServerJMS, verá por pantalla unos mensajes similares a estos:

```
url: tcp://localhost:12500
Registered channels: [Friends, Spain, Linux]
Registered users: []
Waiting for client messages...
```

El servidor ChatServerJMS ha creado los canales "Friends", "Spain" y "Linux". Además, inicialmente no tendremos ningún usuario registrado en el sistema.

Al lanzar ChatClientJMS, verá la misma interfaz gráfica que la práctica anterior, en la que podrá indicar el nombre de usuario y, una vez conectado, aparecerá el listado de canales disponible. Fíjese que en el campo Server le aparece la url del broker de Artemis (`tcp://localhost:12500` en nuestro caso).

Utilice las aplicaciones cliente para conversar. Compruebe que es posible enviar mensajes a un usuario, aunque el usuario no esté conectado en ese momento y que los mensajes se entregan cuando el usuario se conecta posteriormente. Esto es gracias a que los mensajes se guardan en las colas del proveedor JMS.

a) Comprobación de funcionamiento en una única máquina

En primer lugar, vamos a probar el broker Artemis y la aplicación ChatServerJMS que hemos lanzado a ejecución, usando dos aplicaciones ChatClientJMS que lanzaremos en esa misma máquina.

Para ello, lance dos aplicaciones ChatClientJMS, por ejemplo una para el usuario Harry y otra para el usuario Sally. Compruebe que pueden conversar entre sí, a través de algún canal (por ejemplo, en el canal Linux). Observe si un cliente, al salir de un canal y volver a entrar, puede leer los mensajes que se hayan enviado a ese canal en su ausencia. Fíjese también en los mensajes que muestra ChatServerJMS en su terminal.

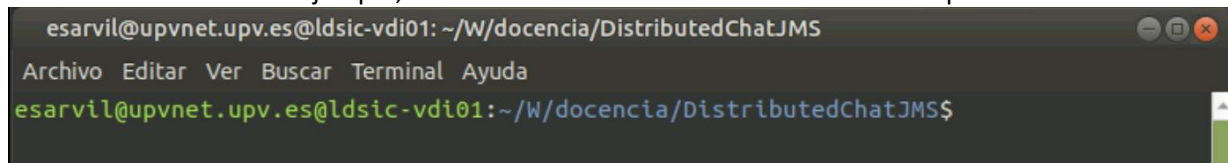
A continuación, desde el usuario Harry envíe varios mensajes privados al usuario Sally. En la pantalla del usuario Harry, la interfaz mostrará el destino "Chatting area Sally". Compruebe que Sally recibe los mensajes privados de Harry. Si es así, cierre la aplicación cliente Sally, pero conserve el área de chat con Sally y siga enviando desde Harry más mensajes a Sally.

Importante: no cambie de usuario o canal, pues entonces se perdería el área de Chat con Sally y ya no se podrá seleccionar a este usuario.

Vuelva a abrir la aplicación cliente de Sally y debería observar que en ese momento recibe los mensajes que se habían enviado mientras no estaba conectada. Mantenga estas aplicaciones clientes abiertas mientras realiza el siguiente apartado.

b) Comprobación de funcionamiento en varias máquinas

En este caso vamos a ejecutar la aplicación ChatClientJMS en otra máquina, y nos conectaremos con ella a nuestro broker y aplicación ChatServerJMS. En el prompt del terminal se le indica a qué máquina está conectado. En este ejemplo, el usuario `esarvil` está conectado a la máquina `ldsic-vdi01`:



```
esarvil@upvnet.upv.es@ldsic-vdi01: ~/W/docencia/DistributedChatJMS
Archivo Editar Ver Buscar Terminal Ayuda
esarvil@upvnet.upv.es@ldsic-vdi01:~/W/docencia/DistributedChatJMS$
```

A continuación, conéctese vía ssh a otra máquina virtual del dsic (por ejemplo, a `ldsic-vdi04`), y lance varios clientes para que interactúen con el broker que había lanzado en el apartado anterior. Por ejemplo, asumiendo que nuestro servidor reside en `ldsic-vdi01`, y que nos queremos ahora conectar a `ldsic-vdi04`, realizaríamos los siguientes pasos:

1º) Nos conectamos a la máquina `ldsic-vdi04` vía ssh

```
$ ssh -X ldsic-vdi04
```


2º) Ponemos a continuación el password y, si todo funciona de forma correcta, estaremos ya conectados a la máquina indicada. Por ejemplo:

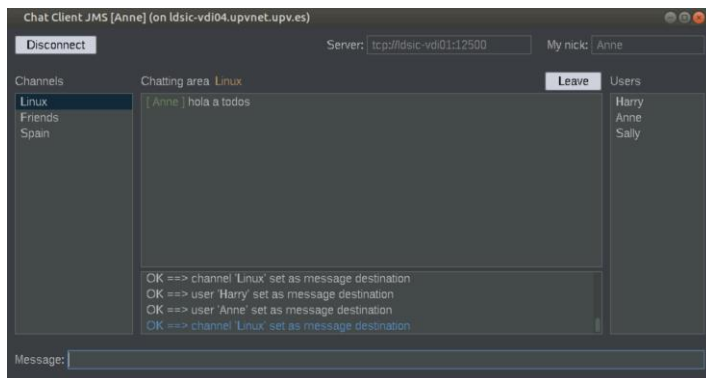
```
esarvil@upvnet.upv.es@ldsic-vdi04:~$
```

3º) Nos colocamos en el directorio DistributedChatJMS de W creado anteriormente y lanzamos ChatClientJMS indicando la máquina donde reside nuestro servidor (en este caso, en ldsic-vdi01)

```
$ cd W/csd/DistributedChatJMS
```

```
$ java -cp lib/*:. ChatClientJMS url=ldsic-vdi01:12500
```

Conéctese al servidor con un nuevo nombre de usuario (por ejemplo, Anne). Si tiene aún abiertas las aplicaciones cliente del apartado anterior, podrá ver que los usuarios Sally, Harry y Anne pueden ahora conversar entre sí. Fíjese también que en el campo Server le aparece la url del broker de Artemis (tcp://ldsic-vdi01:12500 en este caso).

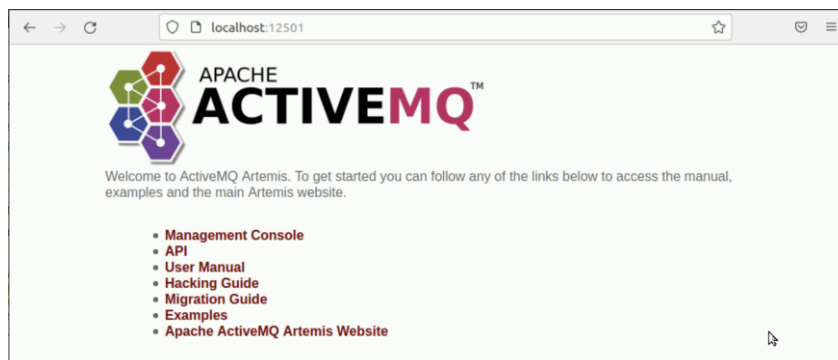


Puede conectarse también al broker Artemis de otro compañero de prácticas, indicando al lanzar su aplicación cliente ChatClientJMS la dirección url en la que está activo el broker de su compañero, de forma similar a lo realizado en este apartado.

Actividad 3

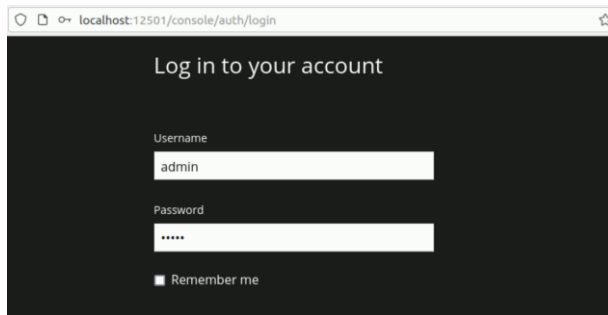
En esta actividad se pretende consultar la consola web para la gestión de Artemis. Como hemos comentado, una vez lanzado Artemis en el puerto 12500, podemos acceder a la consola Artemis desde un navegador con la dirección: localhost:12501

Nos aparecerá la siguiente ventana, donde además del manual de usuario y ejemplos, podemos acceder a la consola de gestión de Artemis (*Management Console*).

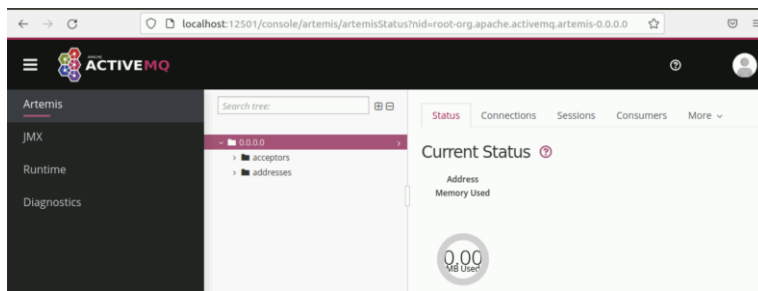


Instrucciones

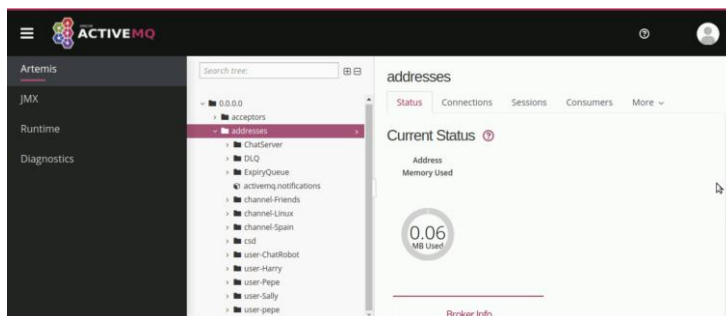
1. Acceda a la consola de gestión de Artemis (**Management Console**), disponible en <http://localhost:12501/console> desde cualquier navegador. Le solicitará un usuario y contraseña. Deberá indicar *admin* (para Username) y *admin* (para Password).



2. Una vez en la consola Artemis, veremos la siguiente pantalla, donde se muestra el estado y podremos consultar las colas y conexiones existentes.



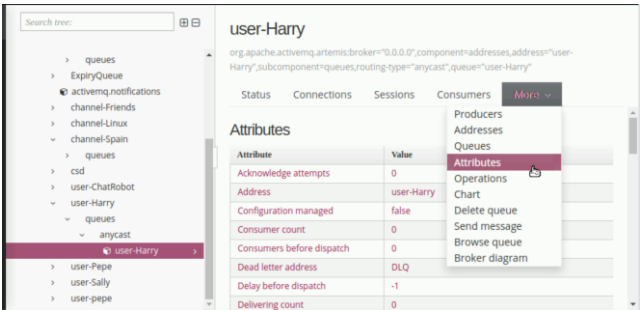
Para ello, pinchando en "addresses" podremos ver las colas que tenemos creadas.



Compruebe que se disponen de los siguientes "addresses" (destinos en Artemis):

- **chatServer**: un destino para el servidor ChatServerJMS, con una cola "queue anycast" asociada, que equivale a una cola (Queue) de JMS.
- **channel-Friends, channel-Linux, channel-Spain**: es decir, un destino para cada canal del chat. Todos ellos tienen una cola "queue multicast" asociada, que equivale al tópico (Topic) de JMS.
- **user-Harry, user-Sally** (o user-Nick correspondiente): es decir, un destino para cada usuario que se haya conectado al chat en algún momento. En este caso, cada uno tendrá una cola "queue anycast" asociada, que equivale a una cola (Queue) de JMS.
- **csd**: esta es la cola que se ha creado en el proceso de instalación de Artemis (según se indica en el script de instalación). En nuestra aplicación de Chat no se ha hecho uso de ella. Pero nos permite mostrar cómo se pueden crear colas en el proceso de instalación de Artemis.

Si pinchamos en cada una de las colas asociadas a estos addresses, podemos consultar sus atributos. Por ejemplo, si pinchamos en la cola de user-Harry y seleccionamos la opción del menú **More - Attributes** (como se muestra en la siguiente figura), veremos las características concretas de esa cola. En este caso, la cola se ha creado de tipo "anycast" y se ha fijado el atributo Max Consumers al valor 1. Esto implica que solamente 1 consumidor podrá leer mensajes de esta cola.



Nota: En nuestra aplicación, hemos utilizado ChatServerJMS para controlar que dos usuarios no puedan estar utilizando el mismo nombre a la vez (es decir, que no puedan conectarse como usuario "Harry") los dos a la vez. Sin embargo, con las propiedades de las colas que ofrece Artemis, esto estaría ya controlado, pues como las colas de usuario se han creado de tipo anycast y con valor Max Consumer a 1, entonces solo puede haber como máximo 1 consumidor asociado a esa cola. ²

Para finalizar con esta actividad, indique en la siguiente tabla el valor de Max Consumer para las colas, así como el tipo asociado a dicha cola (i.e. anycast / multicast). Comente por qué cree que tienen esos valores.

Queues	Max Consumer	Queue type	Comentario
user-Harry			
chatServer			
channel-Friends			
csd			

Actividad 4

Deseamos implementar un ChatRobot que se conecte inicialmente a un canal y que salude cada vez que un usuario entre a dicho canal. Para ello, en el directorio DistributedChatJMS se ofrece el fichero **ChatRobot**, que contiene una implementación parcial de la funcionalidad de ChatRobot para JMS. En esta actividad se pretende que el alumno complete la implementación ChatRobot para que ofrezca la siguiente funcionalidad:

- La aplicación ChatRobot se conectará por defecto al canal Spain y enviará un mensaje de saludo a la cola channel-Spain cada vez que un usuario se conecte a dicho canal.

² La creación de las colas se realiza a través del broker Artemis. Para configurar el tipo de colas a crear, en nuestra aplicación se hace uso de la clase ArtemisCore (disponible en la carpeta utils_jms). En dicha clase, en el método createQueue, se solicita a Artemis la creación de los addresses y queues correspondientes. En caso de tratarse de una cola de tipo anycast, se fija el número de Max Consumers al valor 1.

Instrucciones

En el fichero `ChatRobot.java`, complete el método **main** según se indica. Como guía, puede utilizar el código de `ChatClientJMS` que se ha proporcionado con esta práctica. En concreto, debe

1. Realizar la conexión a Artemis y obtener la cola del servidor.
 - a) Conectar a Artemis, utilizando la factoría de conexión denominada "ConnectionFactory".
 - b) Crear un contexto JMS por defecto.
 - c) Obtener la cola del servidor de nuestra aplicación (llamada "ChatServer"). Este nombre está guardado en la constante `SERVER_QUEUE` de la clase `Destinations` (en paquete `utils_jms`).
2. Envíe un mensaje `CONNECT` a la cola del servidor (i.e. cola `ChatServer`) y espere una respuesta. Dicha respuesta incluye la lista de usuarios y los canales registrados en el servidor.
 - a) Crear un productor.
 - b) Construir un mensaje de tipo "ConnectMessage", para indicar al servidor `ChatServerJMS` que queremos conectarnos al chat. Utilice la clase `MessageFactory` para construir el mensaje del tipo `ConnectMessage`.
 - c) Crear una cola temporal en la que recibir la respuesta del servidor.
 - d) Asignar esta cola temporal en la propiedad "replyTo" del mensaje, para así permitir que el servidor, al recibir nuestro mensaje, sepa a qué cola debe responder.
 - e) Utilizando el productor creado anteriormente, enviar el mensaje a la cola de `ChatServer`.
 - f) Crear un consumidor para nuestra cola temporal.
 - g) El consumidor espera la recepción de un mensaje. Si todo ha ido bien, el servidor habrá creado, en su caso, la cola asociada al usuario y nos habrá enviado un mensaje de tipo `ConnectOKMessage`. Opcionalmente, el alumno puede implementar el código necesario para leer el mensaje recibido y obtener de dicho mensaje la lista de usuarios y canales.
3. Ahora haremos que `ChatRobot` se conecte al canal indicado y envíe un mensaje de bienvenida. Para ello, debe:
 - a) Construir un mensaje de tipo "Join", para indicar al servidor `ChatServerJMS` que queremos unirnos al canal. Asigne la cola temporal creada en el paso 2 a la propiedad "replyTo" del mensaje, para así permitir que el servidor, al recibir nuestro mensaje, sepa a qué cola debe responder.
 - b) Enviar este mensaje creado, utilizando el productor que hemos creado en el paso 2, a la cola de `ChatServer`.
 - c) Obtener la referencia al Topic JMS asociado al canal. (Nota: este paso ya se proporciona en el código).
 - d) Crear un mensaje (de tipo `ChatMessage`) para enviar un mensaje de bienvenida al canal. Enviararlo usando el productor.
4. Finalmente, haremos que `ChatRobot` itere de forma indefinida, esperando mensajes en el canal, los procese y, en caso de que sean mensajes de tipo `JOIN`, enviará un mensaje de saludo al canal. Como guía, puede utilizar el código de `ChatServerJMS` (revise el método `runServerLoop` de dicha clase). En nuestro caso, debe realizar:
 - a) Crear un consumidor para el Topic asociado al canal.
 - b) El consumidor espera la recepción de un mensaje, con espera bloqueante.
 - c) En caso de recibir un mensaje de tipo `USER_JOINS`, enviamos un mensaje al canal, saludando al usuario concreto que se ha unido al canal.

Actividad 5- Opcional

En esta actividad, totalmente opcional, se pretende completar la aplicación ChatRobot, dotándola de la siguiente funcionalidad:

1. Cuando un usuario se conecte al canal en el que está conectado el ChatRobot, enviará un mensaje de saludo, tanto a la cola del usuario como a la cola del canal.
2. Cuando un usuario se desconecte del canal en el que está conectado el ChatRobot, enviará un mensaje de despedida, tanto a la cola del usuario como a la cola del canal.
3. Como el chatRobot aparecerá como otro usuario en la lista de usuarios del chat, cualquier otro usuario conectado podría enviarle un mensaje privado. En dicho caso, el ChatRobot deberá contestarle con un mensaje del tipo "Soy un bot".

ANEXO 1: Instalación de Artemis

Una explicación detallada sobre la configuración del servidor Apache ActiveMQ Artemis está disponible en:

<https://activemq.apache.org/components/artemis/documentation/1.5.3/using-server.html>

A continuación, describiremos los aspectos más relevantes de este servidor relativos a esta práctica.

Descarga de última versión de Artemis

Para instalar Artemis, acceda a la página web:

https://activemq.apache.org/components/artemis/download/past_releases

y descargue **apache-artemis-2.17.0-bin.tar.gz** en un directorio de su disco. Descomprima dicho fichero y colóquese dentro del directorio que se ha creado. Acceda al subdirectorio `bin`. Dentro verá que está el fichero `artemis` que utilizaremos para crear un broker Artemis.

Creación de un broker Artemis. Para instalar el broker en un directorio DIR, abra una terminal, vaya al directorio `bin` que contiene el fichero `artemis`, y ejecute la orden:

```
$ ./artemis create DIR
```

Nota: en esta práctica se hace uso del script "install-artemis-csd.sh" que, además de instalar Artemis, establece ciertos parámetros para su instalación, detallados en el ANEXO 2.

ANEXO 2: Preguntas y respuestas sobre Artemis

¿Cómo iniciar Artemis?

Artemis se puede iniciar con el siguiente mandato, que nos lanzará el broker en primer plano:

```
$ ./artemis run
```

O bien, para lanzarlo en background:

```
$ ./artemis-service start
```

En nuestra práctica, se recomienda iniciarlo con el mandato "run", pues en caso de que se haya configurado de forma errónea el broker, nos mostrará mensajes sobre aquellos puntos de la configuración donde existen problemas. Y podremos así tratar de solucionarlos.

¿Cómo detener Artemis?

Si Artemis se ha iniciado en primer plano (con "artemis run"), se puede detener tecleando Ctrl-C en la terminal. Si se ha iniciado en background, se detiene con el mandato:

```
$ ./artemis-service stop
```

¿Cómo reinicializar Artemis?

Para eliminar todos los buzones creados y poder así volver a usar Artemis como si acabara de ser instalado, debemos detener el servicio Artemis (ver más arriba), eliminar los ficheros del directorio "data" del broker instalado y volver a reactivar el servicio Artemis (con `artemis run`).

ANEXO 3: Detalle del script de instalación de Artemis proporcionado en la práctica

Para esta práctica hemos proporcionado el script `install-artemis-csd.sh` que facilita el proceso de instalación de Artemis. En este Anexo describiremos los aspectos más relevantes de las órdenes incluidas en dicho script. Este script requiere que el usuario introduzca dos parámetros de entrada: PORT y DIR, que se corresponden con el número de puerto (PORT) donde Artemis estará escuchando cuando se lance a ejecución, así como el directorio (DIR) en el que se instalará Artemis. El script comprueba que se hayan introducido estos dos valores y que sean correctos (i.e. que PORT sea un número mayor que 1024 y que DIR no sea un directorio ya existente). Toda esta comprobación se realiza en las líneas 1 a 47.

En la línea 50 se guarda en WPORT el número de puerto para la consola Web de Artemis (que será el número de puerto indicado por el usuario más 1, es decir, `PORT + 1`). Y en las líneas 59 en adelante se procede a instalar Artemis utilizando la orden `.\artemis create DIR` con las opciones que se comentan a continuación (indicamos las más relevantes para nuestra práctica):

<code>--default-port \$PORT</code>	se indica el número de puerto donde artemis quedará a la escucha (PORT es el valor indicado por el usuario).
<code>--http-port \$WPORT</code>	se indica el número de puerto para la consola web artemis (siendo <code>WPORT = PORT + 1</code>).
<code>--allow-anonymous</code>	permite la configuración anónima en la seguridad.
<code>--user admin</code>	indica el nombre de usuario ("admin" en este caso).
<code>--password admin</code>	indica el password del usuario ("admin" en este caso).
<code>--queues csd:anycast</code>	se indica que se cree la cola llamada "csd", de tipo anycast (mensajería punto a punto, equivalente al concepto "Queue" visto en teoría, en el que un mensaje enviado por un productor sólo tiene un consumidor.)

Nota: Artemis permite también crear colas de tipo "multicast", con mensajería publicación - suscripción (publish-suscribe), equivalentes al concepto "Topic" visto en teoría, donde los mensajes se envían a todos los consumidores suscritos a una dirección dada. Las direcciones multicasts pueden ser usadas como Topic JMS cuando se utiliza Artemis como proveedor JMS.

<code>--no-autocreate</code>	Desactiva la creación automática de direcciones. Una dirección representa un punto final de mensajería. Dentro de la configuración, una dirección típica recibe un nombre único, 0 o más colas y un tipo de enrutamiento.
<code>--relax-jolokia</code>	Desactiva la comprobación estricta en <code>jolokia-access.xml</code> . Jolokia es uno de los mecanismos que ofrece Artemis para modificar la configuración del servidor y gestionar los recursos (colas y direcciones).
<code>--no-autotune</code>	Desactiva la sintonización automática en el diario.
<code>--no-amqp-acceptor</code>	Desactiva el aceptador específico de AMQP
<code>--no-hornetq-acceptor</code>	Desactiva el aceptador específico de HornetQ
<code>--no-mqtt-acceptor</code>	Desactiva el aceptador específico de MQTT.
<code>--no-stomp-acceptor</code>	Desactiva el aceptador específico de STOMP.

El elemento `acceptor` contiene un URI que define el tipo de Acceptor a crear junto con su configuración. Cada `acceptor` define la forma en que se pueden realizar las conexiones con el servidor Apache ActiveMQ Artemis. En este caso se han desactivado todos los aceptadores específicos, para así dejar solamente el `acceptor` que usa Artemis para escuchar las conexiones en el puerto indicado anteriormente.