Pràctiques de laboratori

JMS: Java Message Service (3 sessions)

Concurrència i Sistemes Distribuïts

Introducció

Aquesta pràctica té com a objectiu desenvolupar una aplicació simple utilitzant el servei de missatgeria **Java Message Service** (JMS), així com instal·lar, configurar i executar un proveïdor JMS. En particular, es desenvoluparà el component de comunicació d'una aplicació de missatgeria instantània, a la qual hem anomenat "CSD Messenger".

Atès el caràcter introductori d'aquesta pràctica, es tractaran aspectes bàsics com ara la connexió amb el proveïdor JMS i l'enviament i recepció de missatges. Altres aspectes que serien fonamentals per al desenvolupament d'una aplicació real, com ara seguretat i tolerància davant fallades, s'han deixat intencionadament a l'marge en nom de la simplicitat.

La durada estimada d'aquesta pràctica és de tres setmanes. Cada setmana ha d'assistir a una sessió de laboratori on podrà comentar amb el professorat de pràctiques seus progressos i resoldre els dubtes que li sorgeixin. Recordeu que necessitarà destinar una mica de temps del seu treball personal per concloure la pràctica.

Aquesta practica està preparada per realitzar-la des dels sistemes **Windows** dels laboratoris de l'DSIC. Si voleu realitzar-la en el sistema Windows del seu ordinador personal, es requereix tenir instal·lada la versió 1.8 de l'JDK Java d'Oracle (la versió OpenJDK que ve en moltes distribucions és possible que falli) i adaptar els noms dels directoris a la seva instal·lació.

Activitat 1

En aquesta activitat es pretén instal·lar, configurar i executar un proveïdor JMS. El proveïdor JMS que s'instal·larà és Apache ActiveMQ Artemis (simplement Artemis en la resta d'aquest document).

Artemis és un conjunt de llibreries i aplicacions desenvolupades en Java que es distribueix com un arxiu comprimit. Per a aquesta pràctica, ho té disponible al Lessons de la pràctica 5 en el lloc PoliformaT de l'assignatura (fitxer "apache-artemis-1.4.0-bin.zip").

Per evitar problemes de quotes d'espai en disc, el contingut d'aquest fitxer ja està descomprimit en la unitat assignada (\\ fileserver.dsic.upv.es) (M :)

```
M:\ETSINF\csd\jms\artemis
```

Com hem comentat, en aquesta activitat anem a instal·lar, configurar i executar el proveïdor JMS Artemis. Així, crearem primer una instància de l'broker de missatgeria, definirem els ports a utilitzar pel broker i llançarem el broker a execució. Per a això, seguirem les instruccions que s'indiquen a continuació.

NOTA: Com a annex a aquest document s'inclou una breu descripció de les ordres d'Artemis que ens seran necessaris per a la nostra pràctica.

IMPORTANT: En totes les modificacions de fitxer que s'indiquen a continuació, aquestes modificacions s'han de realitzar "a mà" (escrivint directament en el fitxer) i no mitjançant còpia-enganxa des del butlletí de la pràctica (ja que llavors es podrien copiar caràcters no visibles que generaran errors en la creació i execució de l'broker).

Instruccions

1. Obre un explorador d'arxius i ens situarem a M:\ETSINF\csd\jms\artemis\bin
2. Creeu una instància de l'broker de missatgeria en un directori del teu disc (W :), per això creu l'estructura de directoris:
   - W:\csd\jms
   - Des del navegador de fitxers obert en el pas 1, utilitzant el menú "Arxiu" obri un "símbol de sistema". S'obrirà una nova finestra amb l'intèrpret d'ordres bàsic de Windows, ja situat en el directori de treball M:\ETSINF\csd\jms\artemis\bin
   - Escriu l'ordre: `artemis create W:\csd\jms\csdbroker` *(aquest mandat has d'escriure-ho en una sola línia)*
   - Contesta admin, admin, admin i Y a les quatre preguntes que realitzarà el mandat anterior. *(Definiràs així l'usuari, contrasenya i rol que s'utilitzarà per a les connexions anònimes, les quals al seu torn permetràs. Tingues en compte que no es mostrarà cap caràcter en pantalla quan introdueixis la contrasenya)*
3. Com vas a fer aquesta pràctica en un ordinador compartit amb altres usuaris (per realitzar-se a través del Escriptori Remot del DSIC), és imprescindible decidir de forma coordinada quins ports utilitzarà el teu bróker amb la finalitat de que no siguin els mateixos que els de altres brókers llançats per altres usuaris. El professor

de pràctiques t'indicarà què ports utilitzar. Concretament et facilitarà un port al que denominarem NOMBREPORT i el teu bróker utilitzarà aquest port i el següent al que cridarem NOMBREPORT+1

4. En el fitxer "W:\csd\jms\csdbroker\etc\broker.xml" canvia en la següent línia el port 61616 per NOMBREPORT:

```
<acceptor
name="artemis">tcp://0.0.0.0:61616?tcpSendBufferSize=1048576;tcpReceiv
eBufferSize=1048576</acceptor>
```

El fitxer broker.xml és l'arxiu de configuració de servidor central que conté l'element central o core, amb la configuració de servidor principal. Entre altres aspectes, permet definir els anomenats acceptadors (acceptors). Cada acceptador defineix una forma en la qual es poden fer connexions a servidor Artemis. En el nostre cas, hem definit un acceptador que fa servir Artemis per escoltar les connexions al port NUMPUERTO, indicant també les mides de la memòria intermèdia d'enviament de l'TCP i de la memòria intermèdia de recepció de l'TCP en bytes (1048576bytes = 1Mbyte)

5. En el mateix fitxer anterior, elimina la resta de línies amb elements `<acceptor … </acceptor>`

Eliminem aquestes línies, ja que només utilitzarem el acceptador definit en el pas anterior.

6. En el fitxer "W:\csd\jms\csdbroker\etc\bootstrap.xml" modifica en la següent línia el port 8161 per NOMBREPORT+1:

```
<web bind="http://localhost:8161" path="web">
```

El fitxer bootstrap.xml és el fitxer de configuració per defecte que s'utilitza per iniciar el servidor. Entre d'altres aspectes, en aquest fitxer s'indica quin és el fitxer de configuració de l'broker (broker.xml).

7. Ara que ja has configurat el teu bróker, pots executar-ho amb el següent mandat en un intèrpret d'ordres situat en W:\csd\jms\csdbroker\bin (*obri un "símbol de sistema" des del menú "Arxiu" de l'explorador d'arxius un cop situat en el directori W:\csd\jms\csdbroker\bin*)
    - `artemis run`

Veuràs en pantalla el broker en execució:

```
    _        _               _
   / \  ____| |_  ___ __  __ (_) _____
  / _ \|  _ \ __|/ _ \  \/  | |/  __/
 / ___ \ | \/ |_/  __/ |\/| | |\___ \
/_/   \_\|   \__\____|_|  |_|_|/___ /
Apache ActiveMQ Artemis 1.4.0
```

Comprovació

Al llançar a execució el broker, si tot ha funcionat de forma correcta, ens apareixerà en pantalla que el servidor Artemis està en funcionament.

Si no fora així, haurà de repetir el procés anterior, assegurant-se que s'escriu tot de forma correcta. En cas de dubte, consulta al teu professor.

Activitat 2

En aquesta activitat es pretén comprovar el funcionament de sistema de missatgeria instantània proporcionat en la pràctica, denominat CSD Messenger.

Conceptes previs

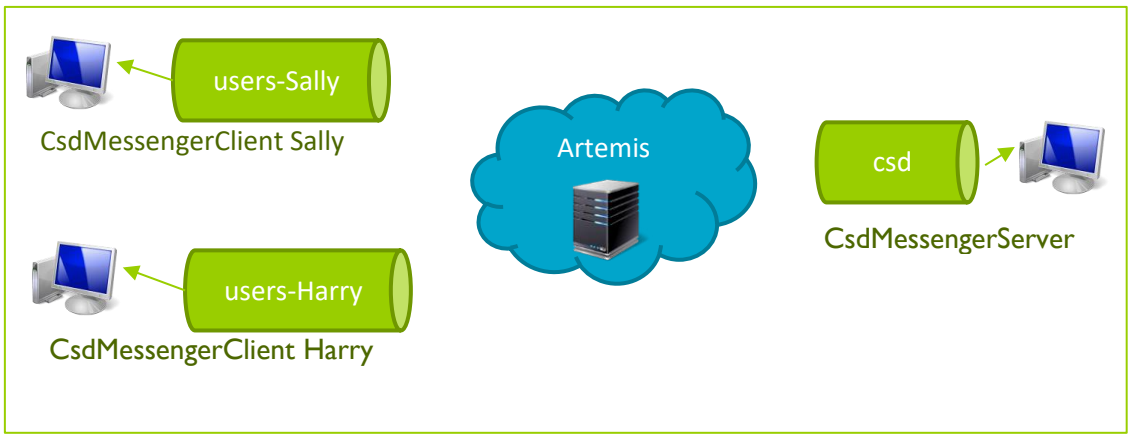
CSD Messenger està format principalment per dues aplicacions: una aplicació servidor (CsdMessengerServer) i una aplicació client (CsdMessengerClient). CsdMessengerServer s'encarrega de la creació de les cues JMS per als clients i d'informar a cada client els noms de la resta de clients que estan utilitzant el servei.

En un moment donat només ha d'haver-hi activa una aplicació CSDMessengerServer, però poden estar actives diverses CsdMessengerClient, una per cada usuari connectat al sistema. Quan s'executa CsdMessengerClient és necessari passar-li com a argument el nom de l'usuari que s'ha connectat.

L'aplicació CsdMessengerServer consumeix missatges de la cua anomenada “csd” i cada aplicació CsdMessengerClient de la cua “users-NOMDELUSUARI”.

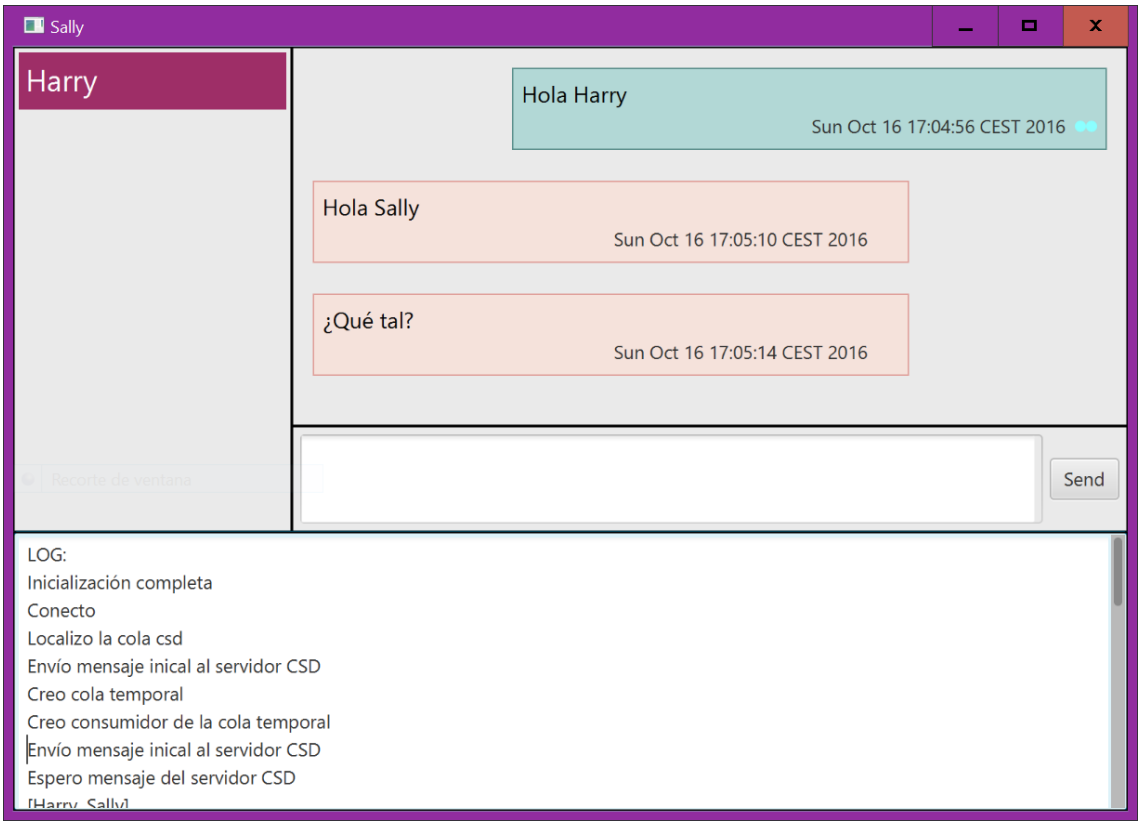
La cua “csd” ha d'existir prèviament, per la qual cosa cal crear-la en la configuració de Artemis. Les cues “users-NOMDELUSUARI” són creades per CSDMessengerServer sota demanda, la primera vegada que un usuari es connecta al sistema.

La següent figura resumeix els components prèviament citats:



CsdMessengerServer és una aplicació Java en mode consola i mostra en pantalla un registre de les peticions que rep en la cua“csd”.

CsdMessengerClient és una aplicació Java amb interfície gràfica d'usuari, desenvolupada amb JavaFX. El seu aspecte és el següent:



Els elements d'aquesta interfície són:

- Panell esquerre. Es mostren els usuaris coneguts pel sistema. El primer usuari que es connecti veurà aquest panell buit. Fes que es connecti almenys un altre usuari perquè comenci a mostrar-se la llista d'usuaris coneguts.
- Panell dret: Mostra la llista de missatges de la conversa amb l'usuari seleccionat en el panell esquerre. Disposa d'un àrea de text per compondre un missatge i un botó per enviar-ho (també pot enviar-se prement la tecla Entrar)
- Panell inferior: Mostra un registre de missatges informatius de l'aplicació.

Instruccions

1. Descarrega de Lessons de la pràctica 5 els fitxers "CsdMessengerServer.jar", "CsdMessengerClient.jar" i "jndi.properties". Deixa'ls en: W:\csd\jms
2. Els fitxers "CsdMessengerServer.jar" i "CsdMessengerClient.jar" contenen les aplicacions servidor i client prèviament esmentades. Abans de poder executar-les, és necessari configurar el port en el qual està escoltant el bróker JMS. Per a això fes el següent.

   2.1. Edita el fitxer "jndi.properties" i canvia el port 61616 per NOMBREPORT en la línia:

   ```
   connectionFactory.ConnectionFactory=tcp://localhost:61616
   ```

   2.2. Afegeix el fitxer "jndi.properties" modificat als arxius .jar de les aplicacions servidor i client (Obri un "símbol de sistema" des del menú "Arxiu" de l'explorador d'arxius un cop situat en el directori W:\csd\jms i escriu):

   ```
   jar uvf CsdMessengerServer.jar jndi.properties
   jar uvf CsdMessengerClient.jar jndi.properties
   ```

3. Ara és necessari canviar la configuració del bróker per adequar-la a l'aplicació que volem desenvolupar. Per a això, fes el següent:

   3.1. Detingues Artemis, prement Control-C a la finestra que s'està executant
   3.2. Crea la cua "csd". Per a això, en el fitxer:
   "W:\csd\jms\csdbroker\etc\broker.xml" afegeix un element <queue> tal com es mostra a continuació:

```
<jms xmlns="urn:activemq:jms">
...
   <queue name="csd"><durable>true</durable></queue>
</jms>
```

   Amb aquesta configuració es crea una cua anomenada "csd" que perdura fins i tot quan el bróker JMS es deté.

   3.3. Deshabilita la creació automàtica de cues. Per a això, afegeix en el mateix fitxer que en el pas anterior un element <auto-create-jms-queues> amb el contingut `false` tal com es mostra a continuació:

```
<address-settings>
   <address-setting>
      ...
      <auto-create-jms-queues>false</auto-create-jms-queues>
   </address-setting>
</address-settings>
```

   Per defecte, en Artemis quan un client referencia una cua JMS aquesta es crea immediatament de forma automàtica. Aquesta característica és inadequada per al servei de missatgeria que volem implementar, ja que la creació de les cues JMS ha d'estar governada per CsdMessengerServer.

   3.4. Executa de nou el bróker: `artemis run`

Comprovació

A partir d'aquest punt ja està tot preparat per provar les aplicacions. Per a això:

1. Executa en una altra terminal el servidor de CsdMessenger
   `java -jar CsdMessengerServer.jar`

2. Executa en una altra terminal un client, per exemple Sally:
   `java -jar CsdMessengerClient.jar Sally`

3. I executa en una altra terminal un altre client, per exemple Harry:
   `java -jar CsdMessengerClient.jar Harry`

4. Pots executar més clients amb altres noms si vols.

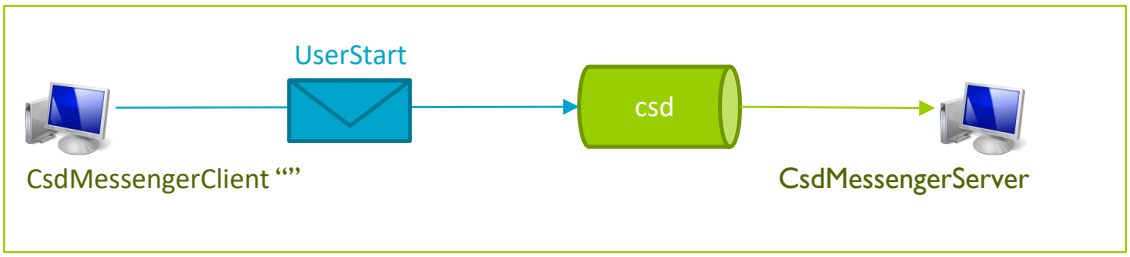
Utilitza les aplicacions client per conversar. Comprova que és possible enviar missatges a un usuari, encara que l'usuari no estigui connectat en aquest moment i que els missatges es lliuren quan l'usuari es connecta posteriorment. Això és gràcies a que els missatges es guarden en les cues del proveïdor JMS. Per exemple, tanca l'aplicació client de Sally, envia-li com Harry alguns missatges, torna a obrir l'aplicació client de Sally i hauries d'observar que en aquest moment rep els missatges enviats mentre no estava connectada.

Activitat 3

Tens a la teva disposició una implementació parcial de l'aplicació CsdMessengerClient, cridada **FuentesCsdMessenger**. En aquest cas, el component d'interfície d'usuari està totalment implementat, però en canvi el component de comunicació està pendent d'implementar, per la qual cosa no es produeix cap tipus d'enviament o recepció de missatges. En aquesta activitat es pretén aconseguir que la implementació FuentesCsdMessenger enviï un missatge a CsdMessengerServer.

La primera tasca que ha de realitzar el component de comunicació és enviar un missatge JMS a CsdMessengerServer, per notificar que es connecta un usuari al sistema.

De moment, amb la finalitat de no complicar a l'excés aquesta activitat, en el missatge no vas a indicar el nom de l'usuari. D'aquesta forma, CsdMessengerServer no contestarà a aquest missatge, per la qual cosa no és necessari que preparis el teu codi per tractar aquesta resposta (ho faràs en l'activitat següent).



Instruccions

1. Descarrega de Lessons els fitxers “**FuentesCsdMessenger.jar**” i “**ArtemisLib.jar**”
2. Si utilitzeu BlueJ, obriu-lo i afegeix en l’apartat Herramientas/Preferencias/Librerías la llibreria “ArtemisLib.Jar”. Reinicia BlueJ per aplicar aquest canvi. Si utilitzeu un altre entorn de programació, afegiu aquesta llibreria com es requereixi en aquest entorn.
3. Obri el fitxer "FuentesCsdMessenger.jar" amb el seu entorn de programació Java. S'haurà creat una carpeta anomenada "FuentesCsdMessenger".
4. Amb un editor extern a BlueJ edita el fitxer “jndi.properties” i canvia el port 61616 per NOMBREPORT, tal com havies fet en l’activitat anterior.
4. De nou, utilitzant el teu entorn de programació Java, edita la classe **comm.Communication**.
5. Implementa en el mètode **initialize()** el següent:
   (NOTA: en el codi de la classe CsdMessengerServer tens exemples molt similars al que es demana a continuació i entre parèntesis disposes d'enllaços a la documentació de Java del mètode que has d'utilitzar)

5.1. Inicialitzar un context JNDI (javax.naming.InitialContext()).
5.2. Establim la connexió amb el broker (javax.naming.InitialContext.lookup()).
5.3. Crear un context JMS (javax.jms.ConnectionFactory. createContext()).
5.4. Crear un *producer* associat al context anterior:
(javax.jms.JMSContext.createProducer()).
5.5. Crear un objecte de la classe *messageBodies.UserStart*, passant al constructor de l'objecte l'argument "". Per exemple:

```
UserStart us=new UserStart("");
```

5.6. Construir un missatge JMS i assignar-li l'objecte anterior
(javax.jms.JMSContext.createObjectMessage()).
5.7. Crear una cua associada al nom "dynamicQueues/csd"
(javax.naming.InitialContext.lookup()).
5.8. Enviar el missatge creat en el punt 5.6 a la cua creada en el punt 5.7 utilitzant el producer creat en el punt 5.4 (javax.jms.JMSProducer.Send())

Comprovació

Assegura't que està en execució CsdMessengerServer.

Executa el mètode ui.CsdMessengerClient.main(), passant com a argument el nom d'usuari que vulguis.

Hauria d'aparèixer una finestra amb diversos avisos i també la interfície gràfica del client buida excepte el missatge "LOG: Inicialización Completa".

D'altra banda, en la terminal on has executat CsdMessengerServer hauries de veure la següent línia:

```
"He recibido una petición UserStart anónima. No hago nada"
```

Activitat 4

Objectiu

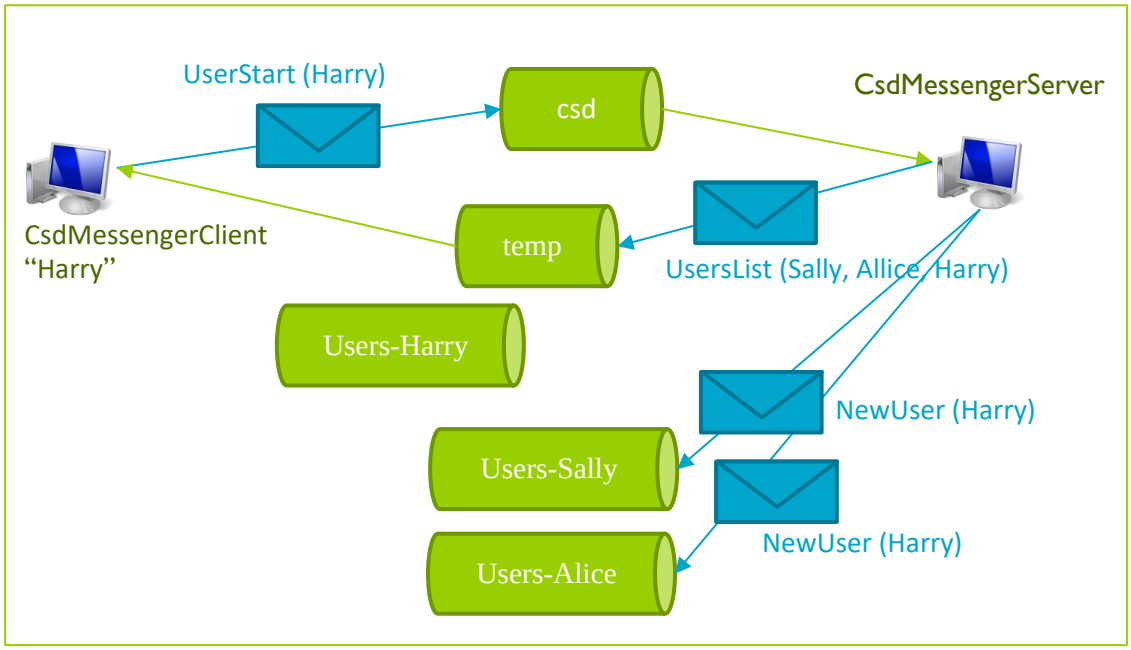
Completar el protocol d'intercanvi de missatges iniciat en l'activitat anterior, atenent la resposta de CsdMessengerServer davant la notificació que es connecta un usuari.

Conceptes previs

Quan CsdMessengerServer rep un missatge del tipus UserStart i se li indica correctament el nom de l'usuari, realitza les següents accions:

- Si és la primera vegada que l'usuari indicat es connecta:
  - Crea la cua JMS "users-NOMDEUSUARI".
  - Notifica a la resta d'usuaris coneguts l'existència del nou usuari.

- Contesta a la cua especificada en la propietat "ReplyTo" del missatge rebut amb una llista d'usuaris coneguts, formada per tots els usuaris que ja han executat l'aplicació client prèviament.



La finalitat dels missatges que envia CsdMessengerServer és que les aplicacions client actualitzin en el seu panell esquerre la llista d'usuaris conegut.

L'ús d'una cua temporal és un patró habitual en aquest tipus d'aplicacions. En aquest cas s'ha utilitzat perquè la primera vegada que un usuari es connecta no disposa encara de la seva cua permanent "users-NOMDEUSUARI", però necessita d'una cua on esperar una resposta.

Instruccions

1. Substitueix el que havies codificat per resoldre el punt 5.6 de l'activitat anterior pel següent (has de mantenir la resta de punts):
2. Construir un missatge JMS amb un objecte de la classe messageBodies.UserStart, passant al constructor de l'objecte el nom de l'usuari que s'ha connectat (resultat del mètode ui.API.getMyName()).
   2.1. Crear una cua JMS temporal (javax.jms.JMSContext.createTemporaryQueue()())
   2.2. Assignar a la propietat ReplyTo del missatge JMS la cua temporal (javax.jms.Message.setJMSReplyTo())
3. Després del codificat per a l'activitat anterior, realitza el següent:
   3.1. Crear un *consumer* associat a la cua temporal (consulta javax.jms.JMSContext.createConsumer()).
   3.2. Esperar l'arribada d'un missatge en la cua temporal (javax.jms.JMSConsumer.receive()). Aquest missatge contindrà un objecte de la classe UsersList que conté un array de strings amb la llista d'usuaris coneguts, recuperable amb el mètode getUsers();
   3.3. Cridar al mètode ui.API.updateUserList() passant el array rebut com a argument.

Comprovació

Assegura't que està en execució CsdMessengerServer.

Executa el mètode ui.CsdMessengerClient.main(), passant com a argument el nom d'usuari que vulguis.

En el panell esquerre es mostrarà la llista d'usuaris coneguts.

Activitat 5

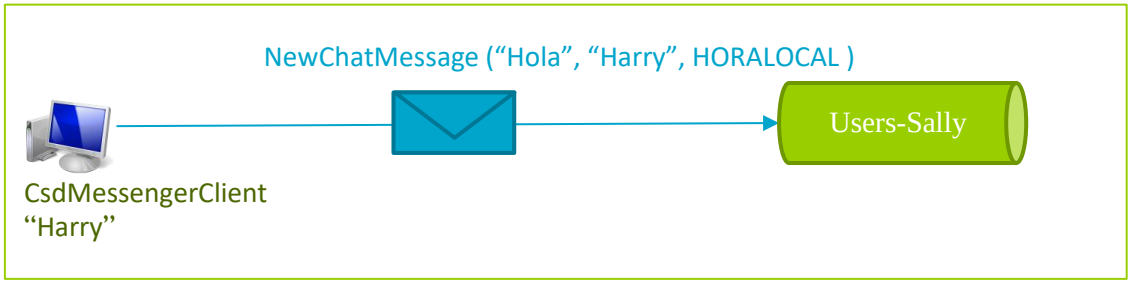
Objectiu

Enviar un missatge JMS a la cua de l'usuari que està seleccionat en el panell esquerre quan es prem el botó d'enviament de missatge o la tecla Entrar.

Conceptes previs

La interfície d'usuari, quan detecta que es prem el botó d'enviament o la tecla Entrar sol·licita al component de comunicació l'enviament d'un missatge del tipus NewChatMessage, en el qual s'indica el text del missatge, el nom de l'usuari emissor i el valor del rellotge local de l'emissor.

Aquest missatge es lliura a la cua JMS de l'usuari seleccionat en el panell esquerre. El missatge serà consumit immediatament o romandrà en la cua, depenent de si l'usuari està o no connectat en aquests moments.



Instruccions

1. Implementa en el mètode comm.Communication. sendChatMessage el següent:
   1.1. Buscar la cua associada al nom "dynamicQueues/users-" + destUser, sent destUser un paràmetre del mètode sendChatMessage (javax.naming.InitialContext.lookup()).
   1.2. Crear un objecte de la classe messageBodies.NewChatMessage, passant al constructor de l'objecte els arguments text, ui.API.getMyName() i timestamp.
   1.3. Construir un missatge JMS i assignar-li l'objecte anterior (javax.jms.JMSContext.createObjectMessage()).
   1.4. Enviar el missatge creat en el pas anterior utilitzant el mateix *producer* que s'havia emprat en l'activitat 3 (javax.jms.JMSProducer.Send()).

Comprovació

Executa l'aplicació client completa que t'hem facilitat:

```
java –jar CsdMessengerClient.jar "Sally"
```

Executa el mètode ui.CsdMessengerClient.main(), passant com a argument el nom d'un altre usuari diferent a Sally.

En la teva aplicació selecciona a Sally i envia-li un missatge. En l'aplicació de Sally hauria d'aparèixer el missatge que has enviat.

Si Sally contesta no podràs veure el seu missatge, ja que en la teva aplicació encara no tractes l'arribada de missatges de xat d'altres usuaris. Això ho vas a realitzar en la següent activitat.

Activitat 6

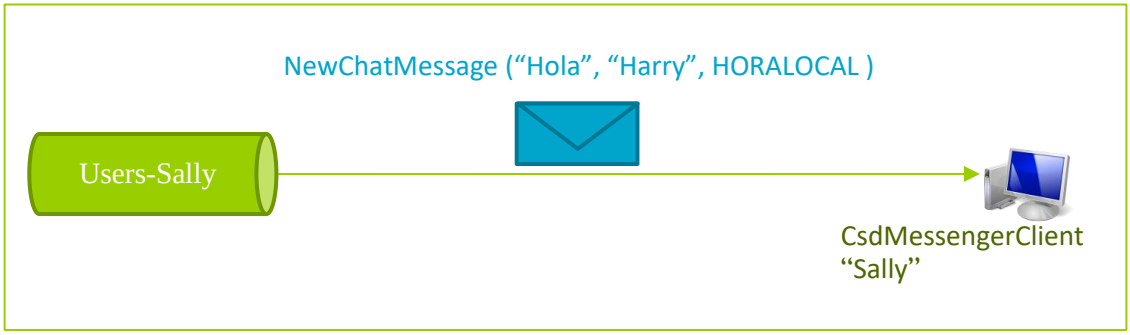
Objectiu

Processar l'arribada de missatges de xat provinents d'altres usuaris.

Conceptes previs

Tal com has vist en l'activitat anterior, un missatge de xat arriba a la cua JMS de l'usuari destinatari. L'aplicació CsdMessengerClient ha d'estar permanentment a l'escolta d'arribades de missatge a aquesta cua, i quan arriba un missatge notificar al component interfície d'usuari el nou missatge de xat perquè es pugui mostrar en el panell dret.

Atès que l'aplicació ha d'estar permanentment a l'escolta de nous missatges, però al mateix temps poder atendre la interacció de l'usuari, la recepció ha de dur-se a terme en un fil d'execució diferent al de la interfície d'usuari. A més, en tractar-se d'un fil diferent, és necessari crear un nou context JMS per a aquest fil, així com un nou *producer* i un nou *consumer*, ja que els contextos JMS estan pensats per ser utilitzats des d'un únic fil de forma simultània.



Instruccions

1. Implementa en el mètode `comm.Communication.run()` el següent:
   1.1. Crear el context JMS a partir del context JMS creat en l'activitat 3 (javax.jms.JMSContext. createContext(), passant com a paràmetre JMSContext.AUTO_ACKNOWLEDGE)
   1.2. Crear un *producer* associat al context anterior (javax.jms.JMSContext.createProducer()).
   1.3. Buscar la cua associada al nom "dynamicQueues/users-" + ui.API.getMyName() (javax.naming.InitialContext.lookup()).
   1.4. Crear un *consumer* per a la cua anterior associat al context creat en el punt 1.1. (javax.jms.JMSContext.createConsumer()).
   1.5. En un bucle, esperar permanentment l'arribada de missatges a la cua creada prèviament (javax.jms.JMSConsumer.receive()).
   1.6. Si l'objecte contingut en el missatge rebut és de la classe messageBodies.NewChatMessage:
      1.6.1. Notificar a la interfície d'usuari l'arribada d'un nou missatge (ui.API.chatMessageReceived());
   1.7. Si l'objecte contingut en el missatge és d'una altra classe diferent
      1.7.1. Mostrar en el Log el nom d'aquesta classe

Comprovació

Executa l'aplicació client completa que t'hem facilitat:

```
java –jar CsdMessengerClient.jar “Sally”
```

Executa el mètode `ui.CsdMessengerClient.main()`, passant com a argument el nom d'un altre usuari diferent a l'anterior. Ara, a més de que Sally vegi els missatges que la teva aplicació li envia, hauries de veure en la teva aplicació els missatges enviats per Sally.

Activitat 7

En aquesta activitat es pretén completar l'aplicació client. Els aspectes que han quedat pendents són els següents:

- Tractar el missatge NewUser enviat per CsdMessengerServer a les aplicacions client quan es connecta un usuari nou.
- Implementar el reconeixement d'arribada de missatges, per mostrar així un segon cercle gris en l'estat de cada missatge enviat .
- Implementar el reconeixement de lectura de missatges, per mostrar així els dos cercles associats a cada missatge en color blau .

L'alumne haurà d'aplicar els coneixements adquirits al llarg d'aquesta pràctica per a realitzar aquesta activitat.

ANNEX

Una explicació detallada sobre la configuració de servidor Apache ActiveMQ Artemis està disponible en:

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

Preguntes i respostes

Creació d'un bróker Artemis:

```
artemis create csdbroker
```

Com iniciar Artemis?

Artemis es pot iniciar amb el següent mandat, que ens llançarà el broker en background:

```
artemis-service start
```

O bé amb el mandat:

```
artemis run
```

Que llançarà el broker, mostrant tots els missatges explicatius de configuració.

En la nostra pràctica, es recomana iniciar-lo amb el mandat "run", ja que en cas que s'hagi configurat de manera errònia el broker, ens mostrarà per pantalla aquells punts de la configuració on hi ha problemes. I podrem així tractar de solucionar-los.

Per tant, per a la nostra pràctica haurem d'utilitzar:

```
W:\csd\jms\csdbroker\bin\artemis run
```

Com detenir Artemis?

Artemis s'atura amb el mandat:

```
artemis-service stop
```

Per tant, per a la nostra pràctica haurem d'utilitzar:

```
W:\csd\jms\csdbroker\bin\artemis-service stop
```

Com reinicialitzar Artemis?

Per eliminar totes les bústies creades i poder així tornar a usar Artemis com si acabés de ser instal·lat, hem aturar el servei Artemis, eliminar els fitxers de directori "data" del broker instal·lat i tornar a reactivar el servei Artemis.

En concret, en la nostra pràctica haurem d'executar:

```
W:\csd\jms\csdbroker\bin\artemis-service stop
```

Eliminarem tots els fitxers de la carpeta:

```
W:\csd\jms\csdbroker\data
```

I tornarem a llançar Artemis:

```
W:\csd\jms\csdbroker\bin\artemis-service start
```

Com eliminar l'historial de converses?

Aquesta pràctica està configurada perquè les converses es guardin en un directori anomenat "csdmessenger". Per suprimir l'historial de converses, només cal eliminar els fitxers d'aquest directori.

En concret, en la nostra pràctica hem d'eliminar tots els fitxers de directori:
`W:\csdmessenger`

Com executar i detenir el servidor CsdMessengerServer?

Per iniciar-ho, obri un terminal i executa

```
java -jar CsdMessengerServer.jar
```

Per detenir-ho, prem Control-C

Com executar i detenir la meua aplicació CsdMessengerClient?

Per executar-la, en BlueJ selecciona la classe `ui.CsdMessengerClient`, prem el botó dret i selecciona el mètode main(), passant com a argument el nom d'usuari que vulguis. Per detenir-la simplement tanca la finestra.

Com tractar les excepcions de JMS?

Diversos mètodes de la API de JMS llancen excepcions i per això quan els utilitzes hauries de tancar el teu codi en una sentència try-catch com la següent:

```
try {
   ……
} catch (NamingException | JMSException e) {
   e.printStackTrace();
}
```

Com mostrar un missatge en el panell d'informació de l'aplicació client?

```
ui.API.addToLog(0, "mensaje a mostrar");
```

Quines classes i mètodes dels proporcionats són els que tinc que modificar?

- Solament els mètodes de la classe `comm.Communication`

Quines classes i mètodes dels proporcionats són els que puc cridar?

- Tots els mètodes públics estàtics de la classe `ui.API`
- Totes les classes i els seus mètodes públics del paquet `messageBodies`

La documentació d'aquests mètodes està disponible en el propi codi de cada classe.