PRACTICA 4 CSD

Chat distribuido orientado a objetos basado en Java-RMI. Se emplea RMI como middleware distribuido subyacente.

EXPLICACIÓN COMPONENTES DE LA APLICACIÓN Y ORDEN DE LANZAMIENTO

Cada componente se debe lanzar en un terminal distinto.

1. **1 SERVIDOR DE NOMBRES (NameServer)**

    - Recibe peticiones de servidores (ChatServer en nuestro caso) para registrar objetos remotos: asociar un nombre lógico a una determinada referencia a objeto remoto.
    - Recibe peticiones de clientes (ChatClient en nuestro caso) para obtener la referencia del servidor asociada al nombre lógico.
    - El servidor de nombres únicamente está para que los clientes de chat encuentres a su correspondiente servidor de chat.

2. **1 o + SERVIDOR DE CHAT (ChatServer)**

    - Identificado mediante un nombre (por defecto “TestServer”)
    - Podemos lanzar varios servidores de chat y registrarlos en el mismo servidor de nombres (con nombres lógico diferentes).

3. **0 o + CLIENTES DE CHAT (ChatCliente)**

    - Cada cliente se identifica mediante un nombre o nick único y si conectará al servidor de chat que se le indique.
    - Si se propone un nick que ya está es uso, el servidor genera el correspondiente mensaje de error (java.rmi.RemoteException: User exists).
    -

PARÁMETROS DE LOS COMPONENTOS DE LA APLICACIÓN

A la hora de lanzar a ejecución los diferentes componentes de la aplicación debemos tener en consideración los parámetros con los que los debemos acompañar.

- NameServer: java NameServer [host=…] [port=…]

    - host: máquina donde se lanza el servidor de nombres (por defecto, asume localhost, es decir, la máquina local)

    - port: número de puerto en el que escuchará el servidor de nombres (por defecto el puerto 9000)
- ServidorChat: java ChatServer [nsHost=…] [nsPort=…] [serverName=…] [host=…]
-
    - nsHost: la máquina donde se ha lanzado el servidor de nombres (por defecto, asume localhost, es decir, la máquina local)
    - nsport: el número de puerto en el que escucha el servidor de nombres (por defecto, asume el puerto 9000)
    - serverName: nombre a asignar al servidor de chat (por defecto "TestServer")
    - hort: dirección propia (por defecto, localhost)

- ChatClient: java ChatClient [nsHost=…] [nsPort=…] [serverName=…] [host=…] [nick=…] [channel=…]
-
    - nshost : la máquina donde se ha lanzado el servidor de nombres (por defecto, asume localhost)
    - nsport: el número de puerto en el que escucha el servidor de nombres (por defecto, asume el puerto 9000)
    - serverName: el nombre lógico por el que preguntará al servidor de nombres para obtener la referencia de objeto asociada (por defecto, "TestServer")
    - host: dirección propia (por defecto, localhost)
    - nick: nombre de usuario con el que se conectará
    - channel: chat al que unirse de forma automática cuando se lance la aplicación.

Todos los parámetros son opcionales, ya que, o bien se pueden introducir mas adelante en la interfaz o existen unos valores predefinidos que adoptaría cada componente en caso de no concretarlos en el lanzamiento a ejecución desde la consola.

Si no se cumple el orden de lanzamiento de los componentes saltarán errores en la consola que nos advertirán del fallo de conexión a alguno de los otros componentes.

Una vez establecida la conexión, el servidor responde con la lista actual de canales, que se muestra en la columna izquierda. Los canales se denominan #xxx. Cuando un usuario se une a un canal, aparece la lista de usuarios suscritos a ese canal en la columna de la derecha.

PROCESO DE LANZAMIENTO A EJECUCIÓN DE COMPONENTES (y mensajes en consola)

- **Tras lanzar a ejecución a un ChatServer:**

  - Terminal servidor de nombres:

    ```
    ==> rebind ( TestServer, {IChatServer,endpoint:[ldsic-vdi13.upvnet.upv.es:9001]} )
    ```

    Indica que el ChatServer se ha registrado en el servidor de nombres, mostrándose su nombre lógico y referencia (endpoint) asociada.

  - Terminal servidor de chat:

    ```
    Channel '#Spain' created.
    Channel '#Linux' created.
    Channel '#Friends' created.
    OK ==> 'TestServer' Running at ldsic-vdi13.upvnet.upv.es:9001
    ```

- **Tras lanzar a ejecución a un ChatClient**

  - Terminal servidor de nombres:

    ```
    ==> resolve (TestServer) --> {IChatServer,endpoint:[ldsic-vdi13.upvnet.upv.es:9001]}
    ```

    Indica que se ha solicitado al servidor de nombre que nos proporcione la referencia asociada a un nombre lógico (TestServer)

  - Terminal del servidor ChatServer: `User 'paca' connected.`
    Por cada cliente que se conecte. Igualmente cuando un cliente se desconecta: `User 'paca' disconnected.`

  - Terminal del cliente ChatClient:

    ```
    OK ==> 'ChatClient' running at ldsic-vdi13.upvnet.upv.es:9002
    ```

A cada componente se le asigna un puerto distinto de escucha.

NO HAY PERSISTENCIA: Esto implica que si el cliente destino no estaba conectado, ya no podrá recibir ese mensaje. Tienen que estar conectados emisor y destinatario para que se producza el el envió y recepción de mensajes de la forma esperada.

En los clientes ChatClient, en la interfaz que proporciona la aplicación, podemos ver como en el centro abajo del "Chatting area" nos llegan distintos mensajes. Estos mensajes nos lo manda el **servidor ChatServer**: mensaje de conexión al servidor, mensaje de entrada a un canal, mensaje de salida de un canal.

ChatRobot

Recibe argumentos en línea de órdenes: nshost, nsport, serverName, channelName, nick.

- nshost y nsport se utilizan para localizar el NameServer.
- serverName el nombre lógico del servidor.
- channelName indica a qué canal debe conectarse.
- nick indica el nick o nombre que tendrá el ChatRobot como usuario del chat.

Debee conectarse al servidor y canal indicados usando el nick proporcionado, y monitorizar todos los mensajes de entrada que le lleguen:

- Si se trata del aviso de que un nuevo cliente X se ha conectado a ese canal, el ChatRobot debe enviar un mensaje público de saludo "Hola X" a través del canal.
- Si se trata de un mensaje privado procedente de un ChatClient, deberá contestarle con otro mensaje privado donde indique "Soy el robot ‘Nick’, y la única respuesta que he aprendido hasta ahora es que 1+1 = 2 ".

DETALLES DE SU CONSTRUCCIÓN (PARA TERMINAR DE COMPRENDER LA PRÁCTICA) EN EL ARCHIVO: “PRACT4 EJERCICIOS EXPLICADOS”