



Instituto Politecnico Nacional

Escuela Superior de Cómputo



Bocanegra Heziquio Yestlanezi

Practica 9

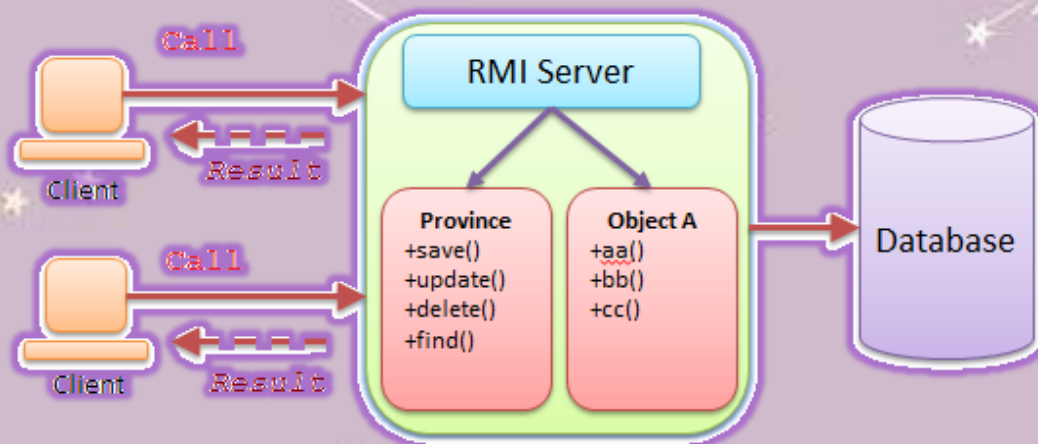
RMI

I. ChatBot

Fecha: 12 de junio de 2021

2CM13

Programación Orientada a Objetos



contenido

introducción 3

arquitectura rmi..... 3

desarrollo..... 6

conclusión 8

Figura 1 Componentes de un sistema RMI. 3

Figura 2 Arquitectura RMI: modelo de cuatro capas. 4

Código 1 Conexión con el servidor..... 6

Código 2 Conexión con el Cliente 6

Código 3 Intersección de respuestas 7

INTRODUCCIÓN

En este tema presentaremos los principales elementos de RMI, un mecanismo que proporciona comunicación remota entre programas escritos en lenguaje Java. En primer lugar, explicaremos los componentes de la arquitectura, explicando los conceptos más relevantes de la invocación remota de objetos. A continuación, detallaremos el funcionamiento del servicio de nombres RMI, necesario para hacer uso de las llamadas remotas. Finalmente comentaremos algunas cuestiones sobre el paso de parámetros a través de la red.

Arquitectura RMI

RMI (*Remote Method Invocation*) es un mecanismo que permite realizar llamadas a métodos de objetos remotos situados en distintas (o la misma) máquinas virtuales Java, compartiendo así recursos y carga de procesamiento a través de varios sistemas.

RMI permite exportar objetos como objetos remotos para que otro proceso remoto pueda acceder directamente como un objeto Java. Todos los objetos de una aplicación distribuida basada en RMI deben ser implementados en Java. Esta es una de las principales ventajas de RMI, ya que RMI forma parte del API de Java, con lo que la integración de objetos remotos en aplicaciones distribuidas se realiza sin necesidad de usar recursos adicionales (como por ejemplo un lenguaje de descripción de interfaces o IDL). De hecho, se utiliza la misma sintaxis para una llamada a un objeto remoto o un objeto local.

La Figura 1 ilustra los componentes implicados en un sistema RMI: un interfaz remoto, un cliente, y uno o más servidores (objetos remotos) residentes en un host.

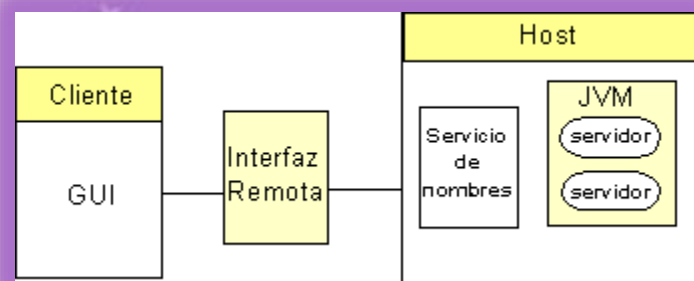


Figura 1 Componentes de un sistema RMI.

El cliente invoca a los objetos remotos mediante la interfaz remota. Un **servicio de nombres** (registro RMI) reside en el *host* proporcionando el mecanismo que el cliente usa para encontrar uno más servidores iniciales RMI.

La interacción con el objeto remoto se lleva a cabo a través de la **interfaz remota**. Esencialmente, ésta describe los métodos que pueden ser invocados de forma remota, y que el objeto remoto implementa. Cuando se obtiene una referencia a un objeto remoto, el objeto no se envía a través de la red al cliente que lo solicita. En su lugar se genera un objeto *proxy* o *stub* que constituye el *proxy* de la parte del cliente del objeto remoto. Todas las interacciones del cliente se realizarán con esta clase *stub*, la cual es responsable de gestionar los datos entre el sistema local y el remoto. Muchos clientes pueden tener referencias a un único objeto remoto. Cada cliente tiene su propio objeto *stub* que representa al objeto remoto, pero dicho objeto remoto NO se replica.

En la parte del servidor, una clase *skeleton* es la responsable de gestionar las llamadas al método y los datos enviados al objeto real referenciado. Éste es el *proxy* de la parte del servidor para el objeto remoto. El sistema completo puede verse como un modelo de cuatro capas, tal y como se ilustra en la Figura 2.

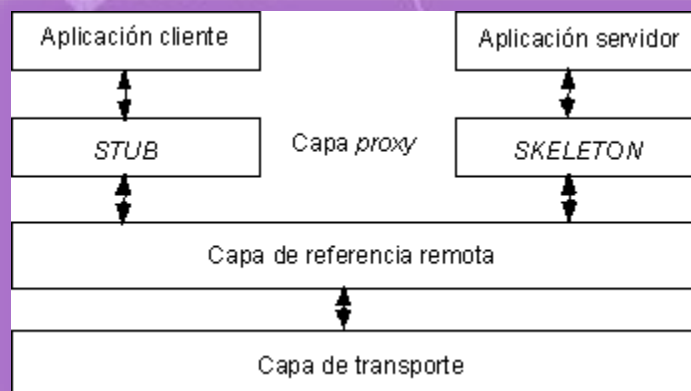



Figura 2 Arquitectura RMI: modelo de cuatro capas.

La primera capa es la de **aplicación**, y se corresponde con la implementación real de las aplicaciones cliente y servidor. Aquí tienen lugar las llamadas a alto nivel para acceder y exportar objetos remotos. Cualquier aplicación que quiera que sus métodos estén disponibles para su acceso por clientes remotos debe declarar dichos métodos en una interfaz que extienda `java.rmi.Remote`. Dicha interfaz se usa básicamente para "marcar" un objeto como remotamente accesible. Una vez que los métodos han sido implementados, el objeto debe ser exportado. Esto puede hacerse de forma implícita si el objeto extiende la clase `UnicastRemoteObject` (paquete `java.rmi.server`), o puede hacerse de forma explícita con una llamada al método `exportObject()` del mismo paquete.

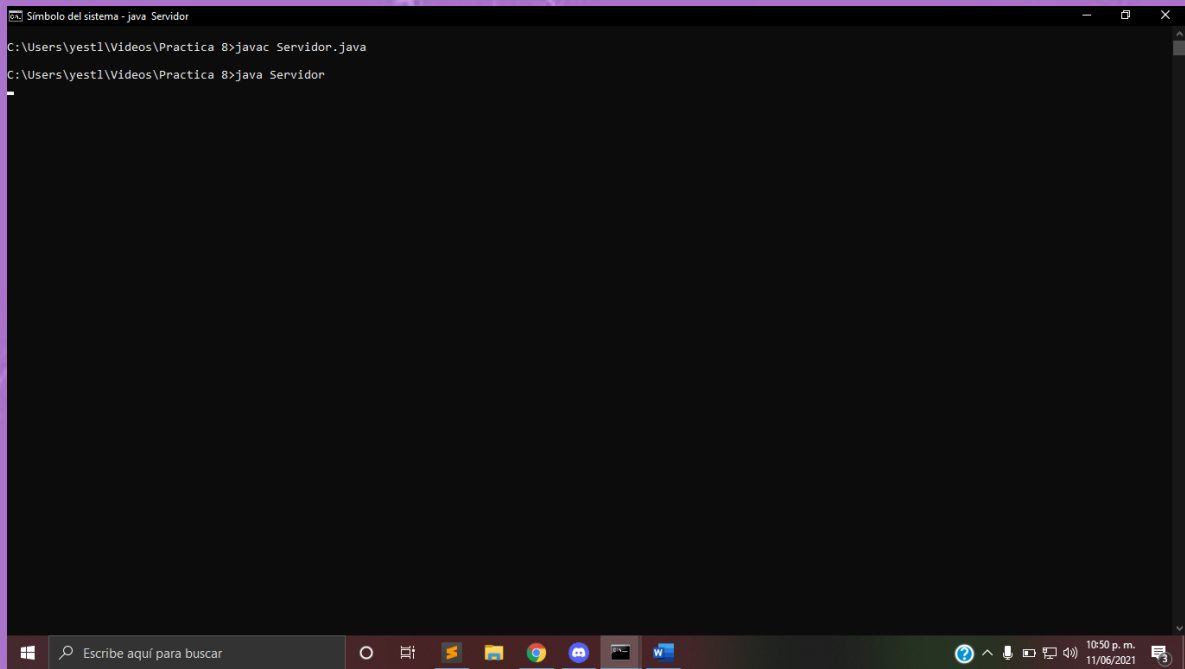


La capa 2 es la capa **proxy**, o capa stub-skeleton. Esta capa es la que interactúa directamente con la capa de aplicación. Todas las llamadas a objetos remotos y acciones sobre sus parámetros y retorno de objetos tienen lugar en esta capa.

La capa 3 es la de **referencia remota**, es responsable del manejo de la parte semántica de las invocaciones remotas. También es responsable de la gestión de la replicación de objetos y realización de tareas específicas de la implementación con los objetos remotos, como el establecimiento de las persistencias semánticas y estrategias adecuadas para la recuperación de conexiones perdidas. En esta capa se espera una conexión de tipo *stream* (*stream-oriented connection*) desde la capa de transporte.

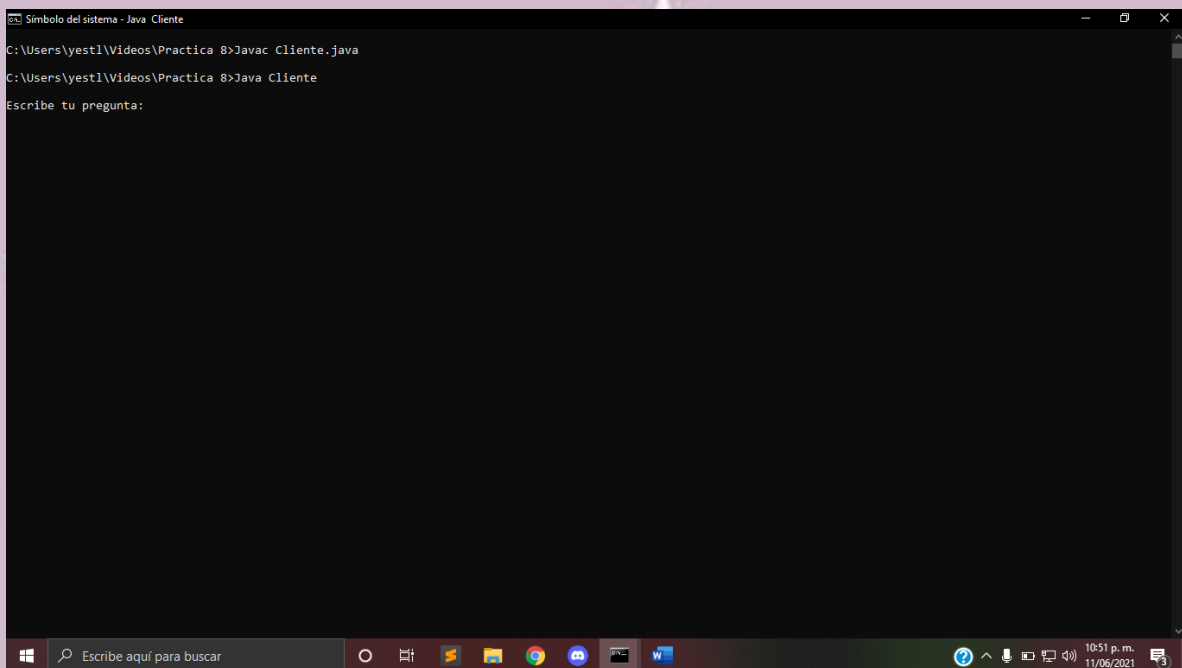
La capa 4 es la de **transporte**. Es la responsable de realizar las conexiones necesarias y manejo del transporte de los datos de una máquina a otra. El protocolo de transporte subyacente para RMI es JRMP (*Java Remote Method Protocol*), que solamente es "comprendido" por programas Java.

DESARROLLO



```
Símbolo del sistema - java Servidor
C:\Users\yestl\Videos\Practica 8>javac Servidor.java
C:\Users\yestl\Videos\Practica 8>java Servidor
```

Código 1 Conexión con el servidor



```
Símbolo del sistema - Java Cliente
C:\Users\yestl\Videos\Practica 8>Javac Cliente.java
C:\Users\yestl\Videos\Practica 8>Java Cliente
Escribe tu pregunta:
```

Código 2 Conexión con el Cliente

```
Simbolo del sistema - Java Cliente
C:\Users\yestl\Videos\Practica 8>javac Cliente.java
C:\Users\yestl\Videos\Practica 8>Java Cliente
Escribe tu pregunta: COMO TE LLAMAS?
Respuesta: Ayer te dije que mi humana me dice Stormy
Escribe tu pregunta: _
```

Código 3 Intersección de respuestas

CONCLUSIÓN

La practica numero 8 se me hizo muy similar a la práctica numero 5, sino me equivoco, ya que se debía contar con un servidor que se conectara con un cliente para poder recibir la pregunta y dar la respuesta, en este caso el programa debía contar un chiste si era solicitado y se tenían guardadas preguntas, así como también el programa debe darte la hora.

Como se muestra en las capturas de pantalla del desarrollo de la práctica, se muestra como se hace la conexión con el servidor y después se compila y ejecuta el cliente, este recibe una pregunta, por ejemplo Como te llamas? Y la respuesta predefinida ya esta en el programa, por lo que debería de aparecer "stormy" como se muestra en las capturas de pantalla.

Lo diferente de la practica es que era necesario utilizar RMI