Remote Method Invocation (RMI) y Socket en JAVA

Ortiz Montiel, Pedro Antonio - Ramírez Jiménez, Andrés Camilo
Universidad de Cartagena
Facultad de Ingeniería

Resumen— RMI (Remote Method Invocation) es un mecanismo que permite realizar llamadas a métodos de objetos remotos situados en distintas (o la misma) máquinas virtuales de Java, compartiendo así recursos y carga de procesamiento a través de varios sistemas. Sockets, al igual que RMI, permite establecer un enlace entre dos programas que se ejecutan independientes el uno del otro (generalmente un programa cliente y un programa servidor).

Abstract— RMI (Remote Method Invocation) is a mechanism that allows calls to methods of remote objects located in different (or the same) Java virtual machines, thus sharing resources and processing load through several systems. Sockets, like RMI, allows you to establish a link between two programs that are running independently of one another (usually a client program and a server program).

I. INTRODUCCIÓN

El presente artículo tiene como finalidad, dar a conocer las características principales de RMI (Remote Method Invocation) y socket. Teniendo encuentra que son mecanismos que permiten comunicaciones entre un cliente y servidor en la misma red. Además, se presentan las ventajas e inconvenientes de cada uno de ellos.

II. RMI (REMOTE METHOD INVOCATION)

A. Definición

RMI es un paquete de JAVA que permite manejar objetos (y sus respectivos métodos) de manera remota, para utilizar los recursos de un servidor de manera transparente para el usuario local.

B. Objetivos de RMI

Proporcionar un middelware[1]para el desarrollo de aplicaciones distribuida manteniendo un estilo Java puro y ortodoxo:

- Facilita la interacción de objetos instanciados en diferente JVM mediante el paradigma de invocación de métodos de los objetos.
- Integra el modelo de objetos distribuidos en el lenguaje Java de una forma natural y manteniendo la semántica que le es propia.
- Capacita para escribir aplicaciones distribuidas tan simple como sea posible.
- Introduce los niveles de seguridad necesarios para garantizar la integridad de las aplicaciones distribuidas. de la conferencia de publicaciones.

C. Componentes

- Clientes: Conducen el flujo de la aplicación. Localizan e invocan métodos ofertados como remotos por los servidores.
- Servidores: Conjunto de objetos que ofrecen interfaces remotas públicas cuyos métodos pueden ser invocados clientes de cualquier procesador de la plataforma.
- Registro: Servicio estático que se establece en cada nudo, en el que se registran los servidores con un nombre, y donde los clientes los localizan por él.

D. Ventajas

- Es realmente fácil de usar si ya se conoce JAVA.
- Portable a través de plataformas con soporte JAVA.
- Bajo costo al convertir sistema existente.

[1] lógica de intercambio de información entre aplicaciones

- Soporta paso de objetos por referencia y/o valor.
- Permite distribuir una aplicación de forma muy transparente, es decir, sin que el programador tenga que modificar apenas el código.

E. Desventajas

- El paso de parámetros por valor implica tiempo para hacer la serialización, enviar los objetos serializados a través de la red y luego volver a recomponer los objetos en el destino.
- A veces, no es tan intuitivo.
- No soportado por otros lenguajes.
- Disminuye el rendimiento con el crecimiento del sistema.

III. SOCKET

Los sockets son un sistema de comunicación entre procesos de diferentes máquinas de una red. Más exactamente, un socket es un punto de comunicación por el cual un proceso puede emitir o recibir información. Fueron popularizados por Berckley Software Distribution, de la universidad norteamericana de Berkley[4].

A. FUNCIONAMIENTO

- Normalmente, un servidor se ejecuta sobre una computadora específica y tiene un socket que responde en un puerto específico.
- El **servidor** únicamente espera, escuchando a través del socket a que un cliente haga una petición. En el lado del cliente: el cliente conoce el nombre de host de la máquina en la cual el servidor se encuentra ejecutando y el número de puerto en el cual el servidor está conectado. Para realizar una petición de conexión, el cliente intenta encontrar al servidor en la máquina servidora en el puerto especificado
- Por la parte del **cliente**, si la conexión es aceptada, un socket se crea de forma satisfactoria y puede usarlo para comunicarse con el servidor. El cliente y el servidor pueden comunicarse escribiendo o leyendo en o desde sus respectivos sockets.

B. MODELO DE COMUNICACIÓN

El modelo de sockets más simple es:

1. El servidor establece un puerto y espera durante un cierto tiempo (timeout segundos), a que el cliente establezca la conexión. Cuando el cliente solicite una conexión,
2. El servidor abrirá la conexión socket con el método accept(). ·
3. El cliente establece una conexión con la máquina host a través del puerto que se designe en puerto
4. El cliente y el servidor se comunican con manejadores InputStream y OutputStream

C. VENTAJAS

- Su principal ventaja radica en que son muy eficientes a la hora de enviar un número elevado de mensajes y datos.
- Compatible con casi todos los lenguajes de Programación
- Disponible en casi todos los sistemas operativos (Windows, Unix, MacOs)

IV. SOCKET Y RMI

Table 1 RMI y socket

SOCKET	RMI
Permite intercambiar flujos de datos	
Una invocación remota a un método, y este es pasado por copia en vez de referencia	Los clientes de objetos remotos pueden interactuar con las interfaces remotas
Hay menos posibilidades de que haya fallas	Se debe manejar una gran cantidad de excepciones
Poca seguridad	Un poco más seguro que socket
Fácil implementación	
Utiliza cliente servidor	

V. REFERENCIAS

[1] JAVA RMI, The Java Remote Method Invocation http://profesores.elo.utfsm.cl/~agv/elo330/2s05/projects/CesarVasquez/sitio_web/comparacion02.html
[2] PROGRAMACION CONCURRENTE Y DISTRIBUIDA, José M. Drake http://www.ctr.unican.es/asignaturas/procodis_3_II/Doc/Procodis_7_01.pdf
[3] Lesson 8: Remote Method Invocation, Oracle http://www.oracle.com/technetwork/java/rmi-141556.html
[4] Socket https://www.ecured.cu/Socket
[5] Socket en JAVA ftp://soporte.uson.mx/PUBLICO/02_ING.SISTEMAS.DE.INFORMACION/TPA/JSockets.pdf