Instituto Politécnico Nacional

Escuela Superior de Cómputo

Aplicaciones para Comunicaciones de Red

Práctica 2

Sopa de Letras

Miembros:

Benítez Ramírez Sergio

Monteros Cervantes Miguel Angel

3CM15

Introducción.

RMI (*Java Remote Method Invocation*) es un mecanismo ofrecido por Java para invocar un método de manera remota. Forma parte del entorno estándar de ejecución de Java y proporciona un mecanismo simple para la comunicación de servidores en aplicaciones distribuidas basadas exclusivamente en Java. Si se requiere comunicación entre otras tecnologías debe utilizarse CORBA o SOAP en lugar de RMI.

RMI se caracteriza por la facilidad de su uso en la programación por estar específicamente diseñado para Java; proporciona paso de objetos por referencia (no permitido por SOAP), recolección de basura distribuida (Garbage Collector distribuido) y paso de tipos arbitrarios (funcionalidad no provista por CORBA).

A través de RMI, un programa Java puede exportar un objeto, con lo que dicho objeto estará accesible a través de la red y el programa permanece a la espera de peticiones en un puerto TCP. A partir de ese momento, un cliente puede conectarse e invocar los métodos proporcionados por el objeto.

La invocación se compone de los siguientes pasos:

- Encapsulado (marshalling) de los parámetros (utilizando la funcionalidad de serialización de Java).
- Invocación del método (del cliente sobre el servidor). El invocador se queda esperando una respuesta.
- Al terminar la ejecución, el servidor serializa el valor de retorno (si lo hay) y lo envía al cliente.
- El código cliente recibe la respuesta y continúa como si la invocación hubiera sido local.

Desarrollo.

Caso 1:

El archivo solo se encuentra en un servidor

```
Escriba el nombre del archivo que desea buscar entre los servidores:
colibri
Se encontraron 1 archivos que coinciden con ese nombre
Archivo 0 :
        Nombre: C:\Users\migue\Documents\NetBeansProjects\Practica7\DB1\colibri.jpg
        Hash: 64fe9daa4ce9bce0a8701d0ba15d1a2b

Escriba el numero del archivo que quiere descargar: 0

La descarga se dividira en 1 servidores
```

Caso 2:

El archivo se encuentra en distintos servidores repetido, por lo tanto, se puede acelerar la descarga

```
Escriba el nombre del archivo que desea buscar entre los servidores:
cancion
Se encontraron 2 archivos que coinciden con ese nombre
Archivo 0 :
        Nombre: C:\Users\migue\Documents\NetBeansProjects\Practica7\DB1\cancion.mp3
        Hash: 6ffe554705819c5589ef4e75fb501153
Archivo 1 :
        Nombre: C:\Users\migue\Documents\NetBeansProjects\Practica7\DB2\cancion2.mp3
        Hash: 6ffe554705819c5589ef4e75fb501153

Escriba el numero del archivo que quiere descargar: 0

La descarga se dividira en 2 servidores
```

Conclusiones

Con esta práctica realizada podemos llegar a la conclusión de que con una arquitectura básica de conexión con sockets podemos implementar una aplicación completa y totalmente productiva, que podrá permitir crear un juego clásico y además incentivar a la gente mayor a ejercitar su cerebro para evitar problemas de memoria en el futuro.

Referencias

https://www.tlm.unavarra.es/~daniel/docencia/rc_itig/rc_itig04_05/slides/clase12-SocketsUDP.pdf

http://www.it.uc3m.es/celeste/docencia/cr/2003/PracticaSocketsUDP/

https://mate.uprh.edu/~jse/cursos/4097/notas/java/javaEspanol/JavaTut/Cap9/socket.html