UNIVERSIDAD DE GRANADA

Universidad de Granada

Desarrollo de Sistemas Distribuidos: Práctica 3

Leire Requena Garcia

16 de mayo de 2021

Índice general

Parte 1:
Ejemplos RMI

Para compilar los ejemplos tenemos que:

- Tener un archivo `server.policy`
- Haber ejecutado `rmiregistry` en una terminal, como es un proceso bloqueante se puede añadir el símborlo & para ejecutarlo en segundo plano
- Compilar los ficheros `.java` con `javac`.
- Usar la siguiente orden seguida del nombre del fichero para ejecutar desde donde este el fichero `server.policy`:

```
1       java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=
        localhost -Djava.security.policy=server.policy "fichero a ejecutar"
```

§1.1: Ejemplo 1

En este ejemplo se envía un mensaje simple desde el cliente para invocar al servidor. Despues de compilar lo ejecutaríamos así:

```
1   java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost
     -Djava.security.policy=server.policy ejemplo1.Ejemplo
2   java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost
     -Djava.security.policy=server.policy ejemplo1.Cliente_Ejemplo localhost 1554
```

El número que añadimos al cliente es la hebra que va a invocar al objeto remoto del servidos, cada una de las órdenes anteriores debe estar en una terminal.

```
DSD/P3 on  main [?] took 44s
→ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.polic
y ejemplo1.Cliente_Ejemplo localhost 1337
Buscando el objeto remoto.
Invocando el objeto remoto.

DSD/P3 on  main [?]
→ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.polic
y ejemplo1.Cliente_Ejemplo localhost 1337
Buscando el objeto remoto.
Invocando el objeto remoto.

DSD/P3 on  main [?]
→ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.polic
y ejemplo1.Cliente_Ejemplo localhost 1338
Buscando el objeto remoto.
Invocando el objeto remoto.
```

```
DSD/P3 on  main [?] took 58s
→ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.polic
y ejemplo1.Ejemplo
Ejemplo bound
Recibida petición de proceso: 1337

Hebra 1337
Recibida petición de proceso: 1337

Hebra 1337
Recibida petición de proceso: 1338

Hebra 1338
```

Si además el número fuese 0 el servidor "dormiría" durante 5 segundos:

```
DSD/P3 on  main [?] took 31s
➜ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.polic
y ejemplo1.Ejemplo
Ejemplo bound
Recibida petición de proceso: 0
Empezamos a dormir.
Terminamos de dormir.

Hebra 0
```

```
DSD/P3 on  main [?]
➜ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.polic
y ejemplo1.Cliente_Ejemplo localhost 0
Buscando el objeto remoto.
Invocando el objeto remoto.

DSD/P3 on  main [?] took 6s
➜
```

§1.2: Ejemplo 2

Este ejemplo es parecido al anterior, solo que en este caso desde el cliente se especifica un número de hebras que llamarán al objeto remoto del servidor.

Para ejecutarlo usamos:

```
1  java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost
    -Djava.security.policy=server.policy ejemplo2.Ejemplo
2  java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost
    -Djava.security.policy=server.policy ejemplo2.Cliente_Ejemplo_Multi_Thread
   localhost 5
```

Y el resultado es el siguiente:

```
DSD/P3 on  main [?]
➜ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.polic
y ejemplo2.Cliente_Ejemplo_Multi_Thread localhost 5
Buscando el objeto remoto
Buscando el objeto remoto
Buscando el objeto remoto
Buscando el objeto remoto
Buscando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto

DSD/P3 on  main [?] took 5s
➜
```

```
DSD/P3 on  main [?]
➜ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.polic
y ejemplo2.Ejemplo
Ejemplo bound

Entra Hebra Cliente 0

Entra Hebra Cliente 2

Entra Hebra Cliente 4

Entra Hebra Cliente 3
Empezamos a dormir
Sale Hebra Cliente 2
Sale Hebra Cliente 3
Sale Hebra Cliente 4

Entra Hebra Cliente 1
Sale Hebra Cliente 1
Terminamos de dormir
Sale Hebra Cliente 0
```

§1.3: Ejemplo 3

Ejecutamos el cliente y servidor con las siguientes órdenes:

```
1  java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost
    -Djava.security.policy=server.policy ejemplo3.Servidor
2  java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost
    -Djava.security.policy=server.policy ejemplo3.Cliente
```

En este ejemplo se llama varias veces a un objeto remoto `Contador` desde su interfaz `Contador_I` incrementando un valor local y devuelve el tiempo que ha tardado como resultado de hacer las peticiones indicadas en el código.

```
DSD/P3 on  main [?]
➜ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.polic
y ejemplo2.Cliente_Ejemplo_Multi_Thread localhost 5
Buscando el objeto remoto
Buscando el objeto remoto
Buscando el objeto remoto
Buscando el objeto remoto
Buscando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto

DSD/P3 on  main [?] took 5s
➜ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.polic
y ejemplo3.Cliente
Poniendo contador a 0
Incrementando ...
Media de las RMI realizadas = 0.114 msegs
RMI realizadas = 1000

DSD/P3 on  main [?]
➜ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.polic
y ejemplo3.Cliente
Poniendo contador a 0
Incrementando ...
Media de las RMI realizadas = 0.086 msegs
RMI realizadas = 1000
```

Parte 2:
Ejercicio Servidores Replicados

En este ejercicio debemos gestionar los registros de los clientes y sus donaciones entre dos réplicas distintas.

En la clase `Servidor` tendremos la estructura de este y se instanciarán el número máximo de réplicas que se desee, con la clase `Replicas`.

He modificado el código para que el máximo de réplicas sean 4, cuando se intente registrar un nuevo cliente se procederá de la siguiente forma:

```
1  public int registrarCliente(int id_cliente) throws RemoteException, NotBoundException
    {
2     int rep_min_registrados = id_replica;
3     int min_total_registrados = getTotalClientes();
4     boolean registrado = false;
5     Replicas_I candidata = (Replicas_I) this;
6
7     for (int i = 0; i < Replicas.total_replicas; i++) {
8        if (i != this.id_replica) {
9           candidata = (Replicas_I) registry.lookup("Replica" + i);
10
11          registrado = candidata.getRegistrado(id_cliente);
12
13          //Buscar la réplica que tiene menor número de clientes registrados
14          int total_registrados_candidata = candidata.getTotalClientes();
15          if(total_registrados_candidata < min_total_registrados) {
16             rep_min_registrados = i;
17             min_total_registrados = total_registrados_candidata;
18          }
19       }
20    }
21
22    if(!registrado) {
23       if (rep_min_registrados != id_replica) {
24          candidata = (Replicas_I) registry.lookup("Replica" + rep_min_registrados);
25          candidata.registrarCliente(id_cliente);
26       }
27       else {
28          System.out.println("Cliente " + id_cliente + " registrado en réplica " +
            this.id_replica);
29          this.clientes_registrados.put(id_cliente, 0);
30       }
31    }
32
33    return rep_min_registrados;
34 }
```

Se escoge la réplica actual como candidata y se cicla entre todas las réplicas que hay en el registro mientras no estemos buscando la actual. Si en la réplica encontrada hay menos clientes que en la candidata se actualiza esta para que la réplica i sea la nueva candidata. Una vez acabado el bucle registramos el cliente en la réplica candidata.

Las donaciones de los clientes se guardan en un `HashMap` donde la clave es el id de cliente y el valor la cantidad total que ha donado ese cliente, para recuperar el total donado guardado entre todas las réplicas se vuelve a cliclar entre ellas sumando los subtotales de cada una.

```java
public int getTotalDonaciones(int id_cliente) throws RemoteException,
NotBoundException {
    int total = -1;

    if(getRegistrado(id_cliente) && haDonado(id_cliente)) {
        total = getSubtotalDonaciones();
        for (int i = 0; i < Replicas.total_replicas; i++)
            if(i != id_replica)
                total +=((Replicas_I) registry.lookup("Replica" + i)).
                getSubtotalDonaciones();
    }

    return total;
}
```

Para ejecutar tendremos que usar:

```
    java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost
     -Djava.security.policy=server.policy practica3.Servidor
    java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost
     -Djava.security.policy=server.policy practica3.Cliente
```

Un ejemplo de ejecución:

```
→ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.polic
y practica3.Cliente
Selecciona una operación:
1 → Registrar nuevo cliente.
2 → Donar
3 → Consultar total de donaciones
4 → Finalizar ejecución
1
Su id de cliente es: 0 y su réplica asignada es: 0

Selecciona una operación:
1 → Registrar nuevo cliente.
2 → Donar
3 → Consultar total de donaciones
4 → Finalizar ejecución
1
Su id de cliente es: 1 y su réplica asignada es: 1

Selecciona una operación:
1 → Registrar nuevo cliente.
2 → Donar
3 → Consultar total de donaciones
4 → Finalizar ejecución
1
Su id de cliente es: 2 y su réplica asignada es: 2

Selecciona una operación:
1 → Registrar nuevo cliente.
2 → Donar
3 → Consultar total de donaciones
4 → Finalizar ejecución
1
Su id de cliente es: 3 y su réplica asignada es: 3
```

```
DSD/P3 on  main [?] took 1m 0s
→ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.polic
y practica3.Servidor
Desplegada réplica: 0
Desplegada réplica: 1
Desplegada réplica: 2
Desplegada réplica: 3
Cliente 0 registrado en réplica 0
Cliente 1 registrado en réplica 1
Cliente 2 registrado en réplica 2
Cliente 3 registrado en réplica 3
Cliente 4 registrado en réplica 0
Cliente 5 registrado en réplica 1
Cliente 6 registrado en réplica 2
Cliente 7 registrado en réplica 3
```