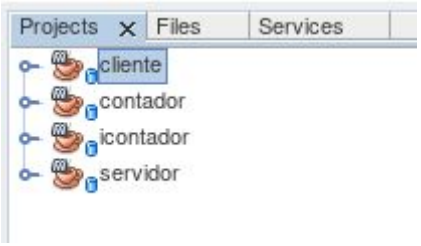
Practica 3: Servidor Replicado

Por Javier Béjar Méndez

Entorno

Se ha realizado en netbeans siguiendo los pasos de la documentación, que nos deja los siguientes proyectos:



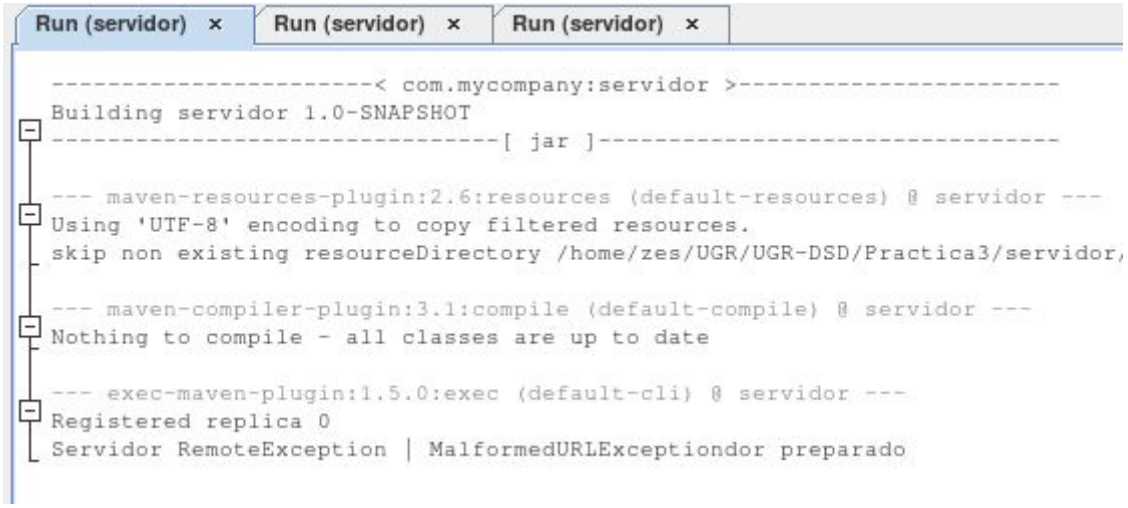
Dónde cliente depende de la interfaz icontador y servidor de contador e icontador. Se genera los permisos para cliente y servidor:

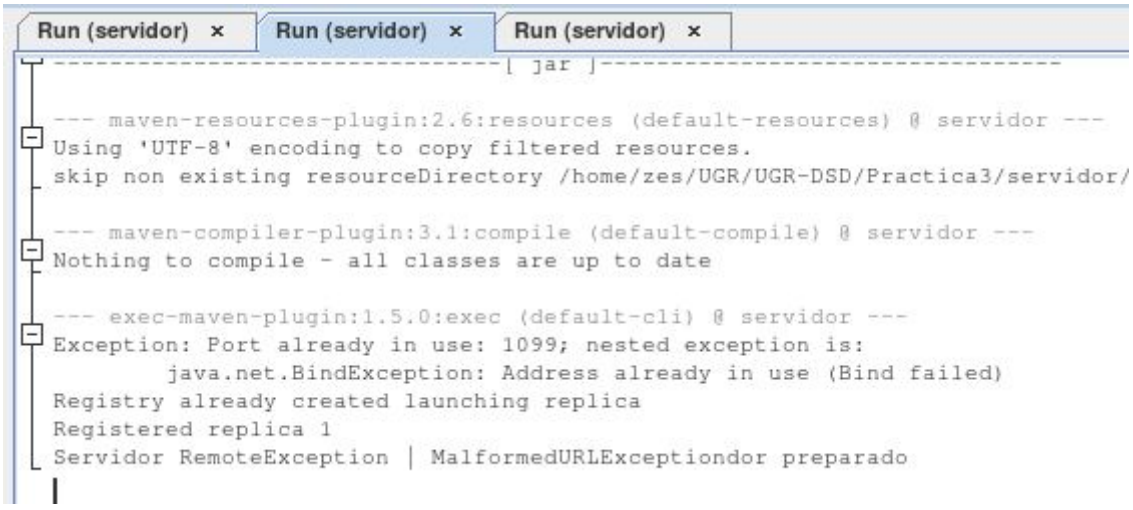
```
// client.policy
grant {
permission java.security.AllPermission;
};
```

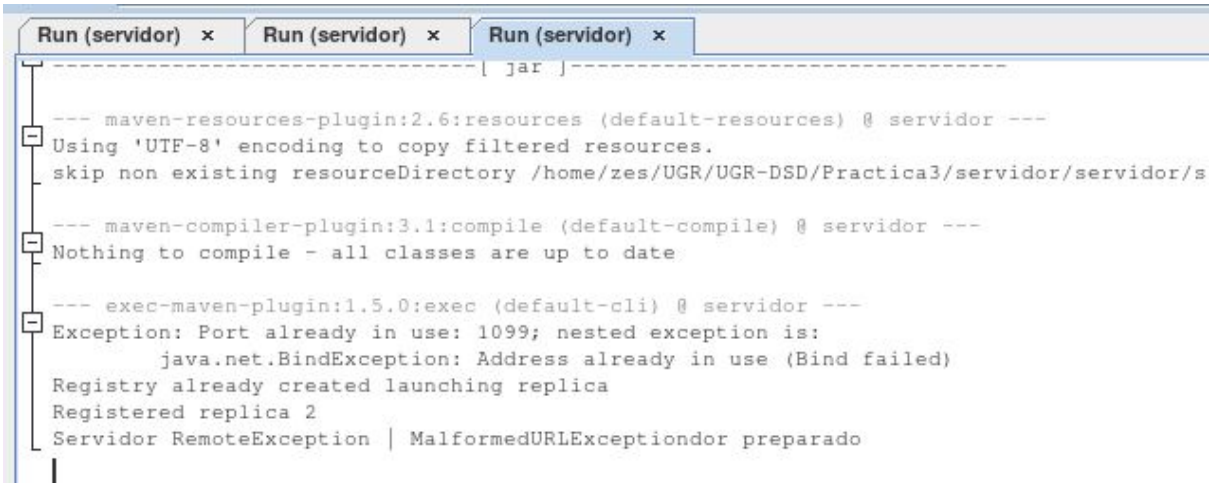
```
// server.policy
grant {
permission java.security.AllPermission;
};
```

Prueba

Soporte de hasta 10 servidores replicados, cumpliendo las restricciones indicadas en la descripción de la práctica, ejemplo con 3 servidores réplica y normal funcionamiento:

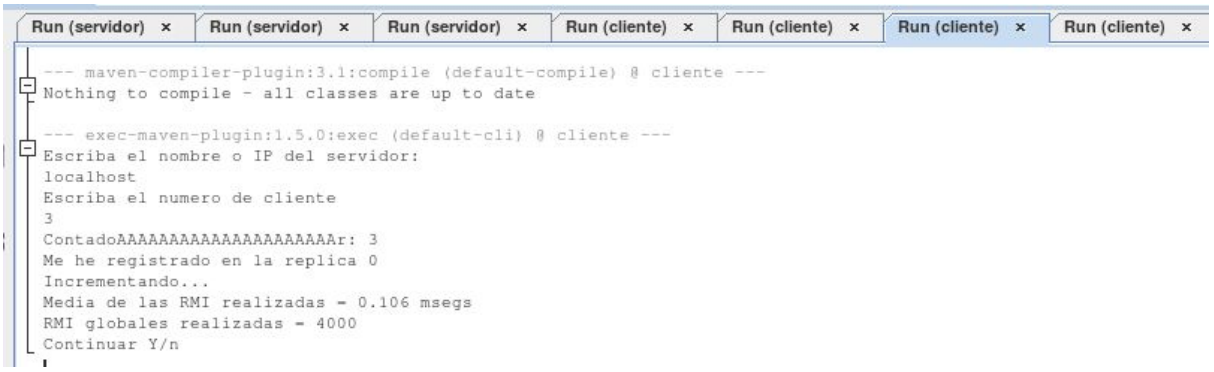






Ahora registramos 4 clientes:
El primer servidor registrará el primero y el último, los demas solo 1

Y el último cliente registrado (3) mostrará las donaciones totales acumuladas por las réplicas:



Si intentamos registrar ahora el 2 otra vez no podriamos:

Código

```java
// icontador.java
package icontador;
import java.rmi.NotBoundException;
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.rmi.registry.Registry;
import java.util.ArrayList;
public interface icontador extends Remote {
int sumar(int cliente, Registry reg) throws RemoteException, NotBoundException;
void sumar (int cliente, int valor) throws RemoteException;
public int incrementar(int cliente) throws RemoteException;
public int registrarCliente(int cliente, Registry reg) throws RemoteException, NotBoundException;
public boolean isRegistered(int cliente) throws RemoteException;
public ArrayList<Integer> getClientes() throws RemoteException;
public int sumar() throws RemoteException;
public void regCliente(int cliente) throws RemoteException;
```

En icontador.java declaramos las funciones necesarias para realizar las donaciones y registros de los clientes.

```java
// contador.java
package contador;
import icontador.icontador;
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
import java.net.MalformedURLException;
import java.util.ArrayList;
import java.rmi.registry.Registry;

public class contador extends UnicastRemoteObject implements icontador {
private int suma;
private ArrayList<Integer> clientes;
private int nReplica;

public contador(int n) throws RemoteException{
    suma = 0;
    clientes = new ArrayList();
    nReplica = n;
}
```

En la clase contador.java que usa el servidor definimos dichas funciones, además tenemos una lista de clientes registrados, el número(nombre) de réplica en el registry y la suma local de la réplica.

El funcionamiento de las funciones es, si el cliente está registrado realizar la operación, si la operación necesita de comunicarse con otras réplicas, mediante el Registry pasado por parámetros obtiene la información necesaria de las otras réplicas.

La clase cliente.java, primero obtiene la lista del registro y realiza la petición de registrarse en un servidor réplica, es atendida por el servidor primero de la lista, el cual lo registrará en el servidor con menor número de clientes, devolviendo el número de la réplica al cliente para que se comunique con ella:

```
String host = "";
 Scanner teclado = new Scanner (System.in);
 System.out.println ("Escriba el nombre o IP del servidor: ");
 host = teclado.nextLine();

 System.out.println ("Escriba el numero de cliente");
 int nCliente = -1;
 nCliente = Integer.parseInt(teclado.nextLine());
// Crea e instala el gestor de seguridad
if (System.getSecurityManager() == null) {System.setSecurityManager(new SecurityManager());}
try {
// Crea el stub para el cliente especificando el nombre del servidor
Registry mireg = LocateRegistry.getRegistry(host, 1099);
int replica;
//icontador micontador = (icontador)mireg.lookup("1");
icontador micontador =(icontador) mireg.lookup(mireg.list()[0]);
replica = micontador.registrarCliente(nCliente, mireg);
if(replica != -1){
    micontador =(icontador) mireg.lookup(mireg.list()[replica]);
    System.out.println("Me he registrado en la replica "+replica);
    boolean seguir = true;
    while(seguir){
        // Obtiene hora de comienzo
        long horacomienzo = System.currentTimeMillis();
        // Incrementa 1000 veces
        System.out.println("Incrementando...");
        for (int i = 0 ; i < 1000 ; i++ ) {
        micontador.incrementar(nCliente);
        }
        // Obtiene hora final, realiza e imprime calculos
        long horafin = System.currentTimeMillis();
        System.out.println("Media de las RMI realizadas = "+ ((horafin - horacomienzo)/1000f)
         + " msegs");
        System.out.println("RMI globales realizadas = " + micontador.sumar(nCliente, mireg));

        //continuar
        System.out.println ("Continuar Y/n ");
        String control = "Y";
        control = teclado.nextLine();
```