package ca.polymtl.inf4410.tp1.client;

import java.rmi.AccessException;

import java.rmi.NotBoundException;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.math.\*;

import java.util.Arrays;

import ca.polymtl.inf4410.tp1.shared.ServerInterface;

public class Client {

public static void main(String[] args) {

String distantHostname = null;

if (args.length > 0) {

distantHostname = args[0];

}

Client client = new Client(distantHostname);

client.run();

}

FakeServer localServer = null; // Pour tester la latence d'un appel de

// fonction normal.

private ServerInterface localServerStub = null;

private ServerInterface distantServerStub = null;

public Client(String distantServerHostname) {

super();

if (System.getSecurityManager() == null) {

System.setSecurityManager(new SecurityManager());

}

localServer = new FakeServer();

localServerStub = loadServerStub("127.0.0.1");

if (distantServerHostname != null) {

distantServerStub = loadServerStub(distantServerHostname);

}

}

private void run() {

for(int i = 1; i <= 7; i++){

System.out.println("i is: " + i);

byte [] x\_arg = new byte[(int)Math.pow(10,i)];

appelNormal(x\_arg);

if (localServerStub != null) {

appelRMILocal(x\_arg);

}

if (distantServerStub != null) {

appelRMIDistant(x\_arg);

}

System.out.print("\n\n ------------- \n\n");

}

}

private ServerInterface loadServerStub(String hostname) {

ServerInterface stub = null;

try {

Registry registry = LocateRegistry.getRegistry(hostname);

stub = (ServerInterface) registry.lookup("server");

} catch (NotBoundException e) {

System.out.println("Erreur: Le nom '" + e.getMessage()

+ "' n'est pas défini dans le registre.");

} catch (AccessException e) {

System.out.println("Erreur: " + e.getMessage());

} catch (RemoteException e) {

System.out.println("Erreur: " + e.getMessage());

}

return stub;

}

private void appelNormal(byte [] x) {

long start = System.nanoTime();

int result = localServer.execute2(x);

long end = System.nanoTime();

System.out.println("Temps écoulé appel normal: " + (end - start)

+ " ns");

System.out.println("Résultat appel normal: " + result);

}

private void appelRMILocal(byte [] x) {

try {

long start = System.nanoTime();

int result = localServerStub.execute2(x);

long end = System.nanoTime();

System.out.println("Temps écoulé appel RMI local: " + (end - start)

+ " ns");

System.out.println("Résultat appel RMI local: " + result);

} catch (RemoteException e) {

System.out.println("Erreur: " + e.getMessage());

}

}

private void appelRMIDistant(byte [] x) {

try {

long start = System.nanoTime();

int result = distantServerStub.execute2(x);

long end = System.nanoTime();

System.out.println("Temps écoulé appel RMI distant: "

+ (end - start) + " ns");

System.out.println("Résultat appel RMI distant: " + result);

} catch (RemoteException e) {

System.out.println("Erreur: " + e.getMessage());

}

}

}

package ca.polymtl.inf4410.tp1.client;

public class FakeServer {

int execute(int a, int b) {

return a + b;

}

int execute2(byte [] x) {

return 0;

}

}

package ca.polymtl.inf4410.tp1.server;

import java.rmi.ConnectException;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.rmi.server.UnicastRemoteObject;

import ca.polymtl.inf4410.tp1.shared.ServerInterface;

public class Server implements ServerInterface {

public static void main(String[] args) {

Server server = new Server();

server.run();

}

public Server() {

super();

}

private void run() {

if (System.getSecurityManager() == null) {

System.setSecurityManager(new SecurityManager());

}

try {

ServerInterface stub = (ServerInterface) UnicastRemoteObject

.exportObject(this, 0);

Registry registry = LocateRegistry.getRegistry();

registry.rebind("server", stub);

System.out.println("Server ready.");

} catch (ConnectException e) {

System.err

.println("Impossible de se connecter au registre RMI. Est-ce que rmiregistry est lancé ?");

System.err.println();

System.err.println("Erreur: " + e.getMessage());

} catch (Exception e) {

System.err.println("Erreur: " + e.getMessage());

}

}

/\*

\* Méthode accessible par RMI. Additionne les deux nombres passés en

\* paramètre.

\*/

@Override

public int execute(int a, int b) throws RemoteException {

return a + b;

}

@Override

public int execute2(byte [] x) throws RemoteException {

return 0;

}

}

package ca.polymtl.inf4410.tp1.shared;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface ServerInterface extends Remote {

int execute(int a, int b) throws RemoteException;

int execute2(byte [] x) throws RemoteException;

}