Resumen parcial 1 DAD

Hilos

Un hilo es una secuencia única de control de flujo dentro de un programa

- Es la unidad de código más pequeña que se puede ejecutar
- Pensado para realizar tareas lentas.

Sistemas Distribuidos

Un sistema distribuido es un conjunto de computadores independientes interconectados a través de una red y que son capaces de colaborar con el fin de realizar una tarea

- Monoprocesador: única unidad central de proceso
- Sistema Distribuido: conjunto de ordenadores independientes
  - interconectados a través de red
  - Colaboran para un fin
- Computadores independientes: no comparten memoria ni espacio de ejecución
- Computación distribuida: computación que se lleva a cabo en un sistema dsitribuido:
  - Servicio de Red
  - Aplicación de red

Formas de colaboración

- Servicio de red: Servicio proporcionado por un tipo especial de programa, servidor en una red
- Aplicación de red: Aplicación para usuarios finales ejecutada en ordenadores conectados a través de la red

Ventajas

- Los computadores y el acceso a la red son económicos
- Compartición de recursos
- Escalabilidad
- Tolerancia a fallos

Desventajas

- Múltiples puntos de fallo
- Aspectos de seguridad

Formas de computación existentes

- Monolítica: un único computador sin conexión y un solo usuario
- Distribuida: Múltiples computadores conectados por red
- Paralela: más de un procesador simultáneamente para ejecutar un programa
- Cooperativa: dividir la computación entre ordenadores conectados para utilizar los ciclos de CPU excedentes

Sockets

Un socket es un punto de conexión entre 2 procesos e identificado por una IP y un puerto

Puntos clave

- Hilo: **Un hilo es una secuencia única de control de flujo dentro de un programa**
- Sistema Distribuido: **Un sistema distribuido es un conjunto de computadores independientes interconectados a través de una red y que son capaces de colaborar con el fin de realizar una tarea**
- Socket:**Un socket es un punto de conexión entre 2 procesos e identificado por una IP y un puerto**

Práctica

Se suele pedir implementar un protocolo, lo que se resume en un servidor que responde a comandos enviados desde un cliente. Generalmente para implementar esto necesitaremos 4 clases:

- Elemento: El objeto con el que trabajaremos o gestionaremos
- Cliente: Un cliente simple implementado por sockets
- Servidor: Un servidor implementado por sockets, almacenará los diferentes elementos en una lista declarada como static y llamará a un ServerThread.
- ServerThread: Hilo al que le pasaremos el socket del servidor y donde se implementará la principal funcionalidad de este.

A continuación se muestran implementaciones de ejemplo:

Elemento

```
public class Elemento {
    String nombre;
    double cantidad;
    public Elemento(String nombre, double cantidad) {
        this.nombre = nombre;
```

```
        this.cantidad = cantidad;
    }
}
```

Cliente

```
public class Cliente {
    Socket socket;
    public void ejecutar() {
        try {
            socket = new Socket("localhost", 5000);
            BufferedReader br = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            PrintWriter pw = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()));
            String linealeida ="";
            Scanner sc = new Scanner(System.in);
            do {
                pw.println(sc.nextLine());
                pw.flush();
                linealeida = br.readLine();
                System.out.println(linealeida);
            } while (linealeida.equals("cerrar") == false);
            sc.close();
            br.close();
            pw.close();
        } catch (IOException e) {
            System.out.println("Error en el Socket del Cliente: " + e);
        }
    }
    public static void main(String[] args) {
        (new Cliente()).ejecutar();
    }
}
```

Servidor

```
public class Server {
    ServerSocket serverSocket;
    Socket socket;
    static Hashtable<String, ArrayList<Elemento>> listaElementos = new
Hashtable<String, ArrayList<Elemento>>();
    public void ejecutar() {
        try {
            serverSocket = new ServerSocket(5000);
            while(true) {
            socket = serverSocket.accept();
```

```
            //implementamos ServerThread
            (new ServerThread(socket)).start();
            }
        } catch (IOException e) {
            System.out.println("Error en el socket servidor: "+e);
        }
    }
    public static void main(String[] args) {
        (new Server()).ejecutar();
    }
}
```

Server Thread

```
public class ServerThread extends Thread{
    Socket socket;
    public ServerThread(Socket socket) {
        this.socket = socket;
    }
    public void run() {
        try {
            BufferedReader br = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            PrintWriter pw = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()));

            String linealeida[];
            do {
                linealeida = br.readLine().split(" ");
                switch (linealeida[0]) {
                    case "ADD":
                        if(linealeida.length == 3) {
                            ArrayList<Elemento> listado;
                            listado =
Server.listaElementos.get(linealeida[1]);
                            Elemento elemento = new Elemento(linealeida[1],
Double.parseDouble(linealeida[2]));
                            if (listado == null) {
                                listado = new ArrayList<Elemento>();
                                Server.listaElementos.put(linealeida[1],
listado);
                            }
                            listado.add(elemento);
                            pw.println("Elemento insertado");
                            pw.flush();
                        }else {
                            pw.println("Uso incorrecto para ADD");
                            pw.flush();
                        }
```

```
                        break;
                    case "GET":
                        if(linealeida.length == 2) {
                            ArrayList<Elemento> listado = new
ArrayList<Elemento>();
                            listado =
Server.listaElementos.get(linealeida[1]);
                            if(listado == null){
                                pw.println("Ese elemento no exite en la
lista");
                                pw.flush();
                            }else {
                                double total=0;
                                for(Elemento elemento : listado) {
                                    total += elemento.cantidad;
                                }
                                pw.println("Total: " + total);
                                pw.flush();
                            }
                        }else {
                            pw.println("USo incorrecto del comando GET");
                            pw.flush();
                        }
                        break;
                    case "DELETE":
                        if(linealeida.length == 2) {
                            ArrayList<Elemento> listado = new
ArrayList<Elemento>();
                            listado =
Server.listaElementos.get(linealeida[1]);
                            if(listado == null){
                                pw.println("Ese elemento no existe");
                                pw.flush();
                            }else {
                                listado.remove(listado.size()-1);
                                pw.println("Elemento eliminado");
                                pw.flush();
                            }
                        }else {
                            pw.println("USo incorrecto del comando DELETE");
                            pw.flush();
                        }
                        break;
                    case "EXIT":
                        pw.println("cerrar");
                        pw.flush();
                        break;
                    default:
                        pw.println("Error: Comando inexistente");
                        pw.flush();
                        break;
```

```
                }
            }while(linealeida[0].equals("EXIT")==false);
            br.close();
            pw.close();
        } catch (IOException e) {
            System.out.println("Error en el Socket ServerThread: "+ e);
        }
    }
}
```

From:
https://knoppia.net/ - **Knoppia**

Permanent link:
https://knoppia.net/doku.php?id=dad:resumen_parcial1&rev=1700122333

Last update: **2023/11/16 08:12**