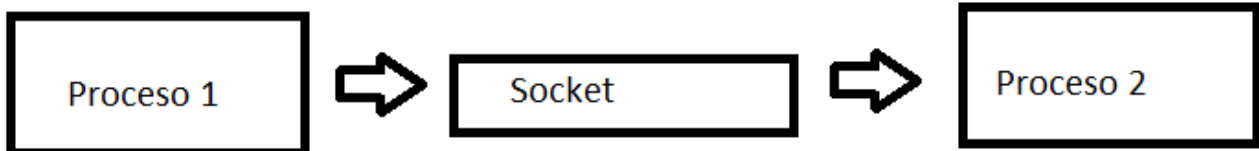


# Sockets

Un Socket es un punto de conexión (Una Tubería o un Canal) entre 2 procesos e identificado por una IP y un Puerto. En este caso serán 2 procesos ejecutándose simultáneamente conectados por un socket.



Un socket funciona de una forma similar a la de los ficheros.

## Cliente

```
Socket socket = new Socket("127.0.0.1", 5000);

BufferedReader br = null;
PrintWriter pw = null;

br = new BufferedReader(new InputStreamReader(socket.getInputStream()))//Buffer de lectura
pw = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()))//Buffer de Escritura

br.readLine();
pw.println("texto a enviar");//Texto a mandar por el Socket
pw.flush();//Similar al fflush stdin de C OJO
pw.push();//
```

## Servidor

```
//Primero creamos el socket
ServerSocket serverSocket = new ServerSocket(2013)//Indicamos el puerto de escucha, en este caso 2013

//esperamos una petición
Socket socket = ServerSocket.accept();//Esto para la ejecución y nos devuelve un socket

BufferedReader br = null;
PrintWriter pw = null;

br = new BufferedReader(new InputStreamReader(socket.getInputStream()))//Buffer de lectura
pw = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()))//Buffer de Escritura

br.readLine();
pw.println("texto a enviar");//Texto a mandar por el Socket
pw.flush();//Similar al fflush stdin de C OJO
pw.push();
```

## Implementación Cliente

```
1 package jelouda;
2
3 import java.io.BufferedReader;
10
11 public class Cliente {
12     public void ejecutar() {
13         try {
14
15
16             System.out.print("Lanzando conexión...");
17
18             Socket socket = new Socket ("127.0.0.1", Servidor.PUERTO);//Conectamos al servidor
19
20             System.out.println("[OK]");
21
22
23             BufferedReader br = new BufferedReader(new InputStreamReader(socket.getInputStream()));//Bufere de lectura
24             PrintWriter pw = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()));//Buffer de escritura
25
26             String cadenaRecibida = br.readLine();
27             System.out.println(cadenaRecibida);
28
29             pw.println("cadena recibida" + cadenaRecibida);
30             pw.flush();
31
32             System.out.println("Fin del Cliente");
33
34             }catch(IOException e){
35                 e.printStackTrace();
36             }
37     }
38
39     public static void main(String[] args) {
40         Cliente client = new Cliente();
41         client.ejecutar();
42     }
43 }
44 }
```

## Implementación Servidor

```

1 package jelouda;
2
3 import java.io.BufferedReader;
4 import java.io.IOException;
5 import java.io.InputStreamReader;
6 import java.io.OutputStreamWriter;
7 import java.io.PrintWriter;
8 import java.net.ServerSocket;
9 import java.net.Socket;
10
11 public class Servidor {
12     public static int PUERTO = 5000; //Indicamos el puerto del servidor
13     public void ejecutar() {
14         try {
15
16             System.out.println("Lanzando Servidor...");
17             ServerSocket serverSocket = new ServerSocket(Servidor.PUERTO);
18
19             Socket socket = serverSocket.accept();
20
21             BufferedReader br = new BufferedReader(new InputStreamReader(socket.getInputStream())); //Buffer entrada
22             PrintWriter pw = new PrintWriter(new OutputStreamWriter(socket.getOutputStream())); //Buffer salida
23
24             String cadenaRecibida = br.readLine();
25
26             pw.print(cadenaRecibida);
27             pw.flush();
28             System.out.println("Fin del Servidor");
29         } catch (IOException e) {
30             e.printStackTrace();
31         }
32     }
33
34     public static void main(String[] args) {
35         Servidor server = new Servidor();
36         server.ejecutar();
37     }
38 }
39

```

```
package jelouda;
```

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;
```

```
public class Servidor {
    public static int PUERTO = 5000; //Indicamos el puerto del servidor
    public void ejecutar() {
        try {
            System.out.println("Lanzando Servidor...");
            ServerSocket serverSocket = new ServerSocket(Servidor.PUERTO);
            Socket socket = serverSocket.accept();
            BufferedReader br = new BufferedReader(new
InputStreamReader(socket.getInputStream())); //Buffer entrada
            PrintWriter pw = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream())); //Buffer salida
            String cadenaRecibida = br.readLine();
            pw.print(cadenaRecibida);
            pw.flush();

```

```
        System.out.println("Fin del Servidor");
    }catch(IOException e){
        e.printStackTrace();
    }
}
public static void main(String[] args) {
    Servidor server = new Servidor();
    server.ejecutar();
}
}
```

## Implementación Servidor continuo

Añadimos un nuevo método al que llamaremos ejecutar infinito:

From:

<https://www.knoppia.net/> - Knoppia

Permanent link:

<https://www.knoppia.net/doku.php?id=dad:sockets&rev=1696500522>

Last update: **2023/10/05 10:08**

