Programación de servicios y procesos

Act3.2. Lectura remota de ficheros

Francisco José García Cutillas | 2FPGS_DAM

Índice

Ejercicio 1

Lectura remota de ficheros. Desarrollar una aplicación en Java que permita leer un fichero de texto ubicado en otro ordenador a través de sockets. Los pasos del proceso serán los siguientes:

- **Programa cliente: solicita al usuario el nombre de un fichero incluyendo su ruta completa dentro del sistema de archivos del servidor.**
- **Programa cliente: envía el nombre y la ruta del fichero al servidor.**
- **Programa servidor: lee el contenido del fichero y se lo envía al cliente.**
- **Programa cliente: muestra el contenido por la pantalla.**

Código del servidor

```java
package com.mycompany.garcia_cutillas_franciscojose_act3_2servidor;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;

/**
 *
 * @author Fran
 */
public class Garcia_Cutillas_FranciscoJose_Act3_2Servidor {

    public static void main(String[] args) {

        try {

            //Creamos un objeto de tipo SocketTCPServer con un puerto por el que nos vamos a conectar
            SocketTCPServer servidor = new SocketTCPServer( puerto: 49171);

            //Iniciamos la comunicación
            servidor.start();

            //Leemos la información pedida por el cliente
            InputStreamReader peticionCliente = new InputStreamReader( in: servidor.getIs());
            BufferedReader br = new BufferedReader( in: peticionCliente);
            String ruta = br.readLine();

            //Leemos la cadena del fichero
            String cadena = leeFichero(ruta);

            //Preparamos el mensaje para enviarlo al cliente
            OutputStream mensajeSalida = servidor.getSocket().getOutputStream();

            //Enviamos el mensaje
            mensajeSalida.write( b: cadena.getBytes());

            //Cerramos la comunicación
            mensajeSalida.close();
            br.close();
            peticionCliente.close();
            servidor.stop();

        } catch (IOException ex) {

            System.out.println("Error de conexión con el servidor: " + ex.getMessage());

        }

    }
```

```java

    //Método que lee una cadena de un fichero
    public static String leeFichero(String ruta){

        String salida = "";

        File fichero = new File( pathname: ruta);

        try{

            FileReader fr = new FileReader( file: fichero);
            int caracter = fr.read();
            while(caracter != -1){

                salida += (char)caracter;
                caracter = fr.read();

            }

            fr.close();

        } catch(IOException ex) {

            System.out.println("Error: " + ex.getMessage());

        }

        return salida;

    }

}

```

```java
package com.mycompany.garcia_cutillas_franciscojose_act3_2servidor;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

/**
 *
 * @author Fran
 */
public class SocketTCPServer {

    private ServerSocket serverSocket;
    private Socket socket;
    private InputStream is;
    private OutputStream os;
    private InputStreamReader isr;
    private BufferedReader br;
    private PrintWriter pw;

    //Constructor para crear un socket de tipo ServerSocket
    public SocketTCPServer(int puerto) throws IOException{

        serverSocket = new ServerSocket( port: puerto);

    }

    //Método para iniciar la comunicación
    public void start() throws IOException{

        System.out.println( x: "(Servidor) Esperando conexiones...");
        socket = serverSocket.accept();
        is = socket.getInputStream();
        os = socket.getOutputStream();
        System.out.println( x: "(Servidor) Conexión establecida.");

    }

    //Método para parar la comunicación
    public void stop() throws IOException{

        System.out.println( x: "(Servidor) Cerrando conexiones...");
        is.close();
        os.close();
        socket.close();
        serverSocket.close();
        System.out.println( x: "(Servidor) Conexiones cerradas.");

    }

    //Método para iniciar una comunicación con el cliente
    public void abrirComunicacion(){

        isr = new InputStreamReader( in: is);
        br = new BufferedReader( in: isr);
        pw = new PrintWriter( out: os,  autoFlush: true);

    }

```

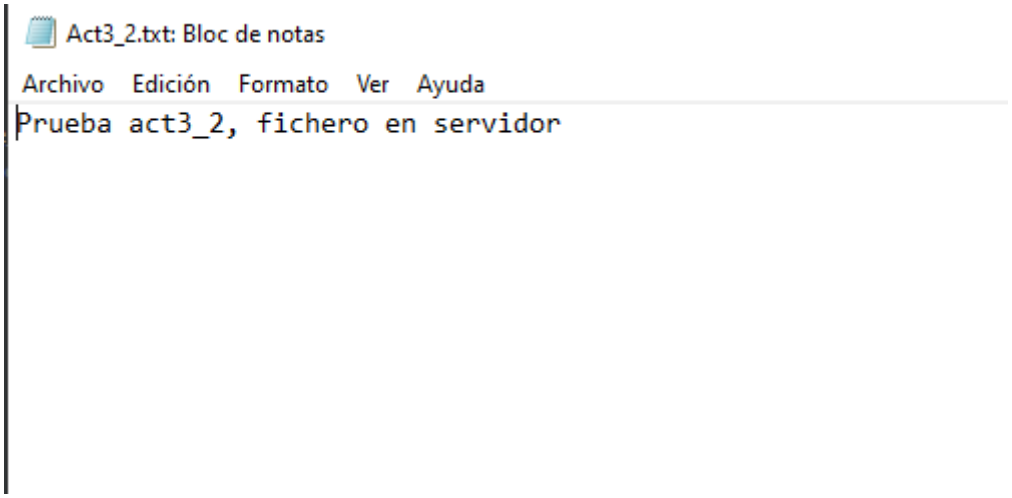
```java
    //Método para terminar la comunicación con el cliente
    public void cerrarComunicacion() throws IOException{

        br.close();
        isr.close();
        pw.close();

    }

    //Método para leer una cadena de texto enviada desde el cliente
    public String leerMensaje() throws IOException{

        String mensaje = br.readLine();
        return mensaje;

    }

    //Método para enviar una cadena de texto al cliente
    public void enviarMensaje(String mensaje){

        pw.println( x: mensaje);

    }

    //Métodos getter y setter para obtener información de la clase desde fuera
    public ServerSocket getServerSocket() {
        return serverSocket;
    }

    public void setServerSocket(ServerSocket serverSocket) {
        this.serverSocket = serverSocket;
    }

    public Socket getSocket() {
        return socket;
    }

    public void setSocket(Socket socket) {
        this.socket = socket;
    }

    public InputStream getIs() {
        return is;
    }

    public void setIs(InputStream is) {
        this.is = is;
    }

    public OutputStream getOs() {
        return os;
    }

    public void setOs(OutputStream os) {
        this.os = os;
    }

}
```

```java
package com.mycompany.garcia_cutillas_franciscojose_act3_2cliente;

import java.io.IOException;
import java.net.UnknownHostException;
import java.util.Scanner;

/**
 *
 * @author Fran
 */
public class Garcia_Cutillas_FranciscoJose_Act3_2Cliente {

    public static void main(String[] args) {

        SocketTCPClient cliente = new SocketTCPClient( serverIP: "localhost",  serverPort: 49171);

        try {

            //Inicio la comunicación
            cliente.start();
            cliente.abrirComunicacion();

            //Preparo y envio el mensaje al servidor con la ruta del fichero
            String mensajeEnviar = "";
            Scanner sc = new Scanner( source: System.in);
            System.out.print( s: "Escribe la ruta del fichero:");
            mensajeEnviar = sc.nextLine();
            cliente.enviarMensaje( mensaje: mensajeEnviar);

            //Muestro el mensaje recibido del servidor
            System.out.println( x: "Mensaje recibido del servidor:");
            String mensaje = cliente.leerMensaje();
            System.out.println( x: mensaje);

            //Cierro la conexión
            cliente.cerrarComunicacion();
            cliente.stop();

        } catch (UnknownHostException ex) {

            System.out.println("Error: " + ex.getMessage());

        } catch (IOException ex) {

            System.out.println("Error: " + ex.getMessage());

        }

    }
}
```

Código del cliente

```java
package com.mycompany.garcia_cutillas_franciscojose_act3_2cliente;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.UnknownHostException;

/**
 *
 * @author Fran
 */
public class SocketTCPClient {

    private String serverIP;
    private int serverPort;
    private Socket socket;
    private InputStream is;
    private OutputStream os;
    private InputStreamReader isr;
    private BufferedReader br;
    private PrintWriter pw;

    //Constructor que recibe la ip del servidor y el puerto de comunicación
    public SocketTCPClient (String serverIP, int serverPort){
        this.serverIP = serverIP;
        this.serverPort = serverPort;
    }

    //Método para iniciar la conexión con el servidor
    public void start() throws UnknownHostException, IOException{

        System.out.println(x:"(Cliente) Estableciendo conexion...");
        socket = new Socket(host:serverIP, port:serverPort);
        os = socket.getOutputStream();
        is = socket.getInputStream();
        System.out.println(x:"(Cliente) Conexión establecida.");

    }

    //Método para terminar la conexión con el servidor
    public void stop() throws IOException{

        System.out.println(x:"(Cliente) Cerrando conexión...");
        is.close();
        os.close();
        socket.close();
        System.out.println(x:"(Cliente) Conexiones cerradas.");

    }

    //Método para iniciar una comunicación con el servidor
    public void abrirComunicacion(){

        isr = new InputStreamReader(in:is);
        br = new BufferedReader(in:isr);
        pw = new PrintWriter(out:os, autoFlush:true);

    }

```

```java
    //Método para terminar la comunicación con el servidor
    public void cerrarComunicacion() throws IOException{

        br.close();
        isr.close();
        pw.close();

    }

    //Método para leer una cadena de texto enviada desde el servidor
    public String leerMensaje() throws IOException{

        String mensaje = br.readLine();
        return mensaje;

    }

    //Método para enviar una cadena de texto al servidor
    public void enviarMensaje(String mensaje){

        pw.println( x: mensaje);

    }

    //Métodos getter y settet para obtener los valores del socket desde fuera
    public String getServerIP() {
        return serverIP;
    }

    public void setServerIP(String serverIP) {
        this.serverIP = serverIP;
    }

    public int getServerPort() {
        return serverPort;
    }

    public void setServerPort(int serverPort) {
        this.serverPort = serverPort;
    }

    public Socket getSocket() {
        return socket;
    }

    public void setSocket(Socket socket) {
        this.socket = socket;
    }

    public InputStream getIs() {
        return is;
    }

    public void setIs(InputStream is) {
        this.is = is;
    }

    public OutputStream getOs() {
        return os;
    }

    public void setOs(OutputStream os) {
        this.os = os;
    }

}
```



Salida servidor

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ Garcia_Cutillas_FranciscoJose_Act3_2Servidor ---
(Servidor) Esperando conexiones...
(Servidor) Conexión establecida.
(Servidor) Cerrando conexiones...
(Servidor) Conexiones cerradas.
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  34.498 s
Finished at: 2023-12-09T18:33:03+01:00
------------------------------------------------------------------------
```

Salida cliente

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ Garcia_Cutillas_FranciscoJose_Act3_2Cliente ---
(Cliente) Estableciendo conexion...
(Cliente) Conexión establecida.
Escribe la ruta del fichero:C:\Users\Fran\OneDrive\Documents\2FPGS_DAM\Programacion_Servicios\Tareas\Tema3\Act3_2.txt
Mensaje recibido del servidor:
Prueba act3_2, fichero en servidor
(Cliente) Cerrando conexión...
(Cliente) Conexiones cerradas.
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  29.417 s
Finished at: 2023-12-09T18:33:03+01:00
------------------------------------------------------------------------
```