

Programación de servicios y procesos

Act3.2. Lectura remota de
ficheros

Francisco José García Cutillas | 2FPGS_DAM

Índice

Ejercicio 1	3
-------------------	---

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

```

1  package com.mycompany.garcia_cutillas_franciscojose_act3_2servidor;
2
3  import java.io.BufferedReader;
4  import java.io.File;
5  import java.io.FileReader;
6  import java.io.IOException;
7  import java.io.InputStreamReader;
8  import java.io.OutputStream;
9
10 /**
11  *
12  * @author Fran
13  */
14 public class Garcia_Cutillas_FranciscoJose_Act3_2Servidor {
15
16     public static void main(String[] args) {
17
18         try {
19
20             //Creamos un objeto de tipo SocketTCPServer con un puerto por el que nos vamos a conectar
21             SocketTCPServer servidor = new SocketTCPServer(puerto: 49171);
22
23             //Iniciamos la comunicación
24             servidor.start();
25
26             //Leemos la información pedida por el cliente
27             InputStreamReader petitionCliente = new InputStreamReader(is: servidor.getIs());
28             BufferedReader br = new BufferedReader(is: petitionCliente);
29             String ruta = br.readLine();
30
31             //Leemos la cadena del fichero
32             String cadena = leeFichero(ruta);
33
34             //Preparamos el mensaje para enviarlo al cliente
35             OutputStream mensajeSalida = servidor.getSocket().getOutputStream();
36
37             //Enviamos el mensaje
38             mensajeSalida.write(b: cadena.getBytes());
39
40             //Cerramos la comunicación
41             mensajeSalida.close();
42             br.close();
43             petitionCliente.close();
44             servidor.stop();
45
46         } catch (IOException ex) {
47
48             System.out.println("Error de conexión con el servidor: " + ex.getMessage());
49
50         }
51
52     }
53

```

```
53
54 //Método que lee una cadena de un fichero
55 public static String leeFichero(String ruta){
56
57     String salida = "";
58
59     File fichero = new File(pathname: ruta);
60
61     try{
62
63         FileReader fr = new FileReader(file:fichero);
64         int character = fr.read();
65         while(character != -1){
66
67             salida += (char)character;
68             character = fr.read();
69
70         }
71
72         fr.close();
73
74     } catch(IOException ex) {
75
76         System.out.println("Error: " + ex.getMessage());
77
78     }
79
80     return salida;
81
82 }
83
84 }
```

```

1 package com.mycompany.garcia_cutillas_franciscojose_act3_2servidor;
2
3 import java.io.BufferedReader;
4 import java.io.IOException;
5 import java.io.InputStream;
6 import java.io.InputStreamReader;
7 import java.io.OutputStream;
8 import java.io.PrintWriter;
9 import java.net.ServerSocket;
10 import java.net.Socket;
11
12 /**
13  *
14  * @author Fran
15  */
16 public class SocketTCPServer {
17
18     private ServerSocket serverSocket;
19     private Socket socket;
20     private InputStream is;
21     private OutputStream os;
22     private InputStreamReader isr;
23     private BufferedReader br;
24     private PrintWriter pw;
25
26     //Constructor para crear un socket de tipo ServerSocket
27     public SocketTCPServer(int puerto) throws IOException{
28
29         serverSocket = new ServerSocket(puerto);
30
31     }
32
33     //Método para iniciar la comunicación
34     public void start() throws IOException{
35
36         System.out.println("(Servidor) Esperando conexiones...");
37         socket = serverSocket.accept();
38         is = socket.getInputStream();
39         os = socket.getOutputStream();
40         System.out.println("(Servidor) Conexión establecida.");
41
42     }
43
44     //Método para parar la comunicación
45     public void stop() throws IOException{
46
47         System.out.println("(Servidor) Cerrando conexiones...");
48         is.close();
49         os.close();
50         socket.close();
51         serverSocket.close();
52         System.out.println("(Servidor) Conexiones cerradas.");
53
54     }
55
56     //Método para iniciar una comunicación con el cliente
57     public void abrirComunicacion(){
58
59         isr = new InputStreamReader(is);
60         br = new BufferedReader(isr);
61         pw = new PrintWriter(os, true);
62
63     }
64

```

```

65 //Método para terminar la comunicación con el cliente
66 public void cerrarComunicacion() throws IOException{
67
68     br.close();
69     isr.close();
70     pw.close();
71
72 }
73
74 //Método para leer una cadena de texto enviada desde el cliente
75 public String leerMensaje() throws IOException{
76
77     String mensaje = br.readLine();
78     return mensaje;
79
80 }
81
82 //Método para enviar una cadena de texto al cliente
83 public void enviarMensaje(String mensaje){
84
85     pw.println( ":" + mensaje);
86
87 }
88
89 //Métodos getter y setter para obtener información de la clase desde fuera
90 public ServerSocket getServerSocket() {
91     return serverSocket;
92 }
93
94 public void setServerSocket(ServerSocket serverSocket) {
95     this.serverSocket = serverSocket;
96 }
97
98 public Socket getSocket() {
99     return socket;
100 }
101
102 public void setSocket(Socket socket) {
103     this.socket = socket;
104 }
105
106 public InputStream getIs() {
107     return isr;
108 }
109
110 public void setIs(InputStream isr) {
111     this.isr = isr;
112 }
113
114 public OutputStream getOs() {
115     return os;
116 }
117
118 public void setOs(OutputStream os) {
119     this.os = os;
120 }
121
122 }
123

```

```
1 package com.myccompany.garcia_cutillas_franciscojose_act3_2cliente;
2
3 import java.io.IOException;
4 import java.net.UnknownHostException;
5 import java.util.Scanner;
6
7 /**
8  *
9  * @author Fran
10  */
11 public class Garcia_Cutillas_FranciscoJose_Act3_2Cliente {
12
13     public static void main(String[] args) {
14
15         SocketTCPClient cliente = new SocketTCPClient("localhost", 49171);
16
17         try {
18
19             //Inicio la comunicación
20             cliente.start();
21             cliente.abrirComunicacion();
22
23             //Preparo y envío el mensaje al servidor con la ruta del fichero
24             String mensajeEnviar = "";
25             Scanner sc = new Scanner(System.in);
26             System.out.print("Escribe la ruta del fichero:");
27             mensajeEnviar = sc.nextLine();
28             cliente.enviarMensaje(mensajeEnviar);
29
30             //Muestro el mensaje recibido del servidor
31             System.out.println("Mensaje recibido del servidor:");
32             String mensaje = cliente.leerMensaje();
33             System.out.println(mensaje);
34
35             //Cierro la conexión
36             cliente.cerrarComunicacion();
37             cliente.stop();
38
39         } catch (UnknownHostException ex) {
40
41             System.out.println("Error: " + ex.getMessage());
42
43         } catch (IOException ex) {
44
45             System.out.println("Error: " + ex.getMessage());
46
47         }
48     }
49 }
50
51 }
```

Código del cliente

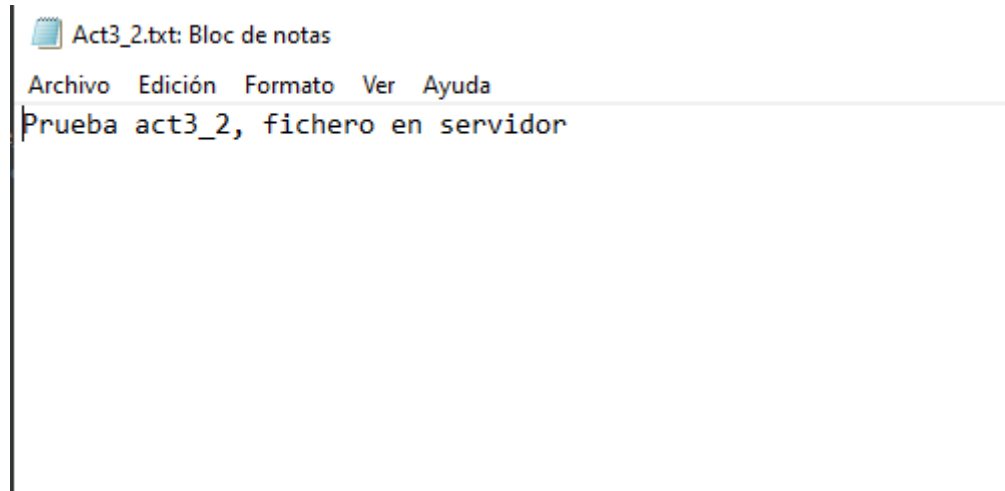
```

1  package com.mycompany.garcia_cutillas_franciscojose_act3_2cliente;
2
3  import java.io.BufferedReader;
4  import java.io.IOException;
5  import java.io.InputStream;
6  import java.io.InputStreamReader;
7  import java.io.OutputStream;
8  import java.io.PrintWriter;
9  import java.net.Socket;
10 import java.net.UnknownHostException;
11
12 /**
13  *
14  * @author Fran
15  */
16 public class SocketTCPClient {
17
18     private String serverIP;
19     private int serverPort;
20     private Socket socket;
21     private InputStream is;
22     private OutputStream os;
23     private InputStreamReader isr;
24     private BufferedReader br;
25     private PrintWriter pw;
26
27     //Constructor que recibe la ip del servidor y el puerto de comunicación
28     public SocketTCPClient (String serverIP, int serverPort){
29         this.serverIP = serverIP;
30         this.serverPort = serverPort;
31     }
32
33     //Método para iniciar la conexión con el servidor
34     public void start() throws UnknownHostException, IOException{
35
36         System.out.println(":(Cliente) Estableciendo conexión...");
37         socket = new Socket(host: serverIP, port: serverPort);
38         os = socket.getOutputStream();
39         is = socket.getInputStream();
40         System.out.println(":(Cliente) Conexión establecida.");
41     }
42
43     //Método para terminar la conexión con el servidor
44     public void stop() throws IOException{
45
46         System.out.println(":(Cliente) Cerrando conexión...");
47         is.close();
48         os.close();
49         socket.close();
50         System.out.println(":(Cliente) Conexiones cerradas.");
51     }
52
53     //Método para iniciar una comunicación con el servidor
54     public void abrirComunicacion(){
55
56         isr = new InputStreamReader(in: is);
57         br = new BufferedReader(in: isr);
58         pw = new PrintWriter(out: os, autoFlush: true);
59     }
60
61 }
62
63

```



```
64 //Método para terminar la comunicación con el servidor
65 public void cerrarComunicacion() throws IOException{
66
67     br.close();
68     isr.close();
69     pw.close();
70
71 }
72
73 //Método para leer una cadena de texto enviada desde el servidor
74 public String leerMensaje() throws IOException{
75
76     String mensaje = br.readLine();
77     return mensaje;
78
79 }
80
81 //Método para enviar una cadena de texto al servidor
82 public void enviarMensaje(String mensaje){
83
84     pw.println( ":" + mensaje);
85
86 }
87
88 //Métodos getter y settet para obtener los valores del socket desde fuera
89 public String getServerIP() {
90     return serverIP;
91 }
92
93 public void setServerIP(String serverIP) {
94     this.serverIP = serverIP;
95 }
96
97 public int getServerPort() {
98     return serverPort;
99 }
100
101 public void setServerPort(int serverPort) {
102     this.serverPort = serverPort;
103 }
104
105 public Socket getSocket() {
106     return socket;
107 }
108
109 public void setSocket(Socket socket) {
110     this.socket = socket;
111 }
112
113 public InputStream getIs() {
114     return isr;
115 }
116
117 public void setIs(InputStream isr) {
118     this.isr = isr;
119 }
120
121 public OutputStream getOs() {
122     return os;
123 }
124
125 public void setOs(OutputStream os) {
126     this.os = os;
127 }
128
129 }
130
```



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
-----
```