UNIVERSIDAD
Panamericana

Comunicación con Sockets en java: Nodo y cliente

Cómputo Distribuido

Jose Manuel Amador - 0250003

Facultad de Ingeniería, Universidad Panamericana

Proyecto parcial 1

Índice

Índice **2**
Introducción: **3**
Desarrollo: **4**
Código y Resultados: 4
Ejemplo de ejecución: 5
Conclusión: **6**

Introducción:

En este reporte se explicará brevemente cómo funciona y se establece la comunicación entre sockets mostrando de igual forma dos códigos de ejemplo junto con su ejecución para demostrar el funcionamiento de lo explicado.
La comunicación entre sockets es una de las técnicas más fundamentales en el desarrollo de aplicaciones distribuidas ya que esta permite la comunicación entre distintos nodos que necesitan compartir información.
Gracias a la comunicación entre sockets podemos disfrutar de servicios como lo pueden ser las páginas web, sistemas de pagos, sistemas de mensajería, entre otros.
El código a analizar será un código escrito en java que consta de dos programas que simulan la interacción entre cliente y servidor.

Desarrollo:

En el desarrollo de este programa se implementó un tipo servidor al que se le llamó “Nodo.java” en el cual se asegura de que este se encargue de esperar conexiones desde el puerto 31010, de esta forma cuando se detecta una conexión y se establece desde el otro programa al que se le llamo “Cliente.java” el servidor muestra una respuesta para confirmar la conexión establecida y posteriormente el programa que emula el cliente finaliza su ejecución.

Código y Resultados:

Nodo.java

```
import java.io.*;
import java.net.*;

public class Nodo {
   public static void main(String[] args) {
       int puerto = 31010;

       try (ServerSocket serverSocket = new ServerSocket(puerto)) {
           System.out.println("Nodo esperando conexiones en el puerto " + puerto +
"...");

           while (true) {
               Socket socket = serverSocket.accept();
               System.out.println("Cliente conectado desde " +
socket.getInetAddress());

               OutputStream output = socket.getOutputStream();
               PrintWriter writer = new PrintWriter(output, true);
               writer.println("Se ha establecido conexión al nodo");

               socket.close();
           }
       } catch (IOException e) {
           e.printStackTrace();
       }
   }
}
```

Cliente.java

```
import java.io.*;
import java.net.*;

public class Cliente {
   public static void main(String[] args) {
       String servidor = "127.0.0.1";
       int puerto = 31010;

       try (Socket socket = new Socket(servidor, puerto)) {
           InputStream input = socket.getInputStream();
           BufferedReader reader = new BufferedReader(new
InputStreamReader(input));

           String mensaje = reader.readLine();
           System.out.println("Mensaje recibido del Nodo: " + mensaje);
       } catch (IOException e) {
           e.printStackTrace();
       }
   }
}
```

Ejemplo de ejecución:

Primero que todo se inicia el código de nodo ya que para que exista una conexión nuestro "servidor" tiene que estar levantado, posteriormente se inicializa el código de cliente que permitirá la conexión, si todo funciona correctamente el output del nodo será el siguiente:

```
Nodo esperando conexiones en el puerto 31010...
Cliente conectado desde /127.0.0.1
```

y el del cliente será el siguiente:

```
Mensaje recibido del Nodo: Se ha establecido conexión al nodo

Process finished with exit code 0
```

Por si existe algún error se ha agregado en el código:

```
catch (IOException e) {
           e.printStackTrace();
```

Esto permitirá el poder troubleshootear algún problema en la conexión ya que esto se encargará de imprimir el problema relacionado con lo que evita que esta conexión no se pueda efectuar.

Conclusión:

En conclusión pudimos ver lo que es necesario para que una conexión se pueda realizar incluyendo desde poder manipular en que puerto quieres efectuar la conexión hasta en qué acción quieres que se ejecute desde el momento en el que la conexión es efectuada sin necesidad de hacer algo adicional aparte de lograr la conexión con el socket.

Este proyecto ha servido para entender de una mejor manera cómo es que se trabaja con la conexión de sockets e inclusive los problemas que se pueden llegar a tener al hacer dicha conexión.

Es importante recalcar que para una implementación mucho más limpia y poder escalar el proyecto es necesario siempre mantener un código limpio teniéndolo bien organizado y asegurarte que se hagan impresiones adecuadas de los errores que se tengan para poder darle un mantenimiento adecuado al proyecto cuando se tenga un problema con el mismo.