

Programación Orientada a Objetos

Práctica 6: Sockets Servidores



Alumno: Vargas Romero Erick Efraín

Profesor: Tecla Parra Roberto

Fecha: 16-05-2017

Grupo: 2CM4

**Introducción**

En la práctica anterior se trabajó principalmente con sockets clientes o bien, se hizo mayor hincapié en ello, pero también tenemos los sockets servidor, los cuales al igual que los sockets cliente son sencillos de utilizar si se conoce como es que funcionan.

**Desarrollo**

Primeramente, debemos considerar que se ha realizado el ChatBot, para lo cual debemos crear un servidor pero también un cliente, siendo así se han creado dos clases.

Primeramente, tenemos la clase client, en la cual tenemos 5 variables de instancia, la primera es un arreglo de Strings llamado Questions, en la cual tenemos las preguntas que se pueden hacer al servidor, posteriormente tenemos otra variable tipo String llamada Response, otra de tipo Socket (Cliente) y nuestros DataInputStream y DataInputStream con los cuales obtendremos los flujos de entrada y salida

String Questions**[]** **=** **{**"Como te llamas"**,**

"Cual es tu color favorito"**,**

"Cuantos anos tienes"**,**

"Cual es tu comida favorita"**,**

"Cual es tu lugar favorito"**,**

"Cual es tu musica favorita"**,**

"Tienes hermanos"**,**

"Como se llama tu papa"**,**

"Como se llama tu mama"**,**

"Que te gusta hacer"**},**

Response**;**

Socket socket**;**

DataOutputStream outputStream**;**

DataInputStream inputStream**;**

A continuación tenemos el constructor de nuestra clase client() en la cual tenemos un ciclo for() con el cual podremos imprimir en pantalla las preguntas que podrá hacer el cliente, además en el constructor se lee que pregunta desea el usuario que sea respondida como se muestra a continuación.

public client**(){**

**for** **(**int i**=**0**;** i**<**Questions**.**length**;** i**++)** **{**

System**.**out**.**println**(**"\n"**+** **(**i**+**1**)** **+** "." **+** Questions**[**i**]** **+** "?"**);**

**}**

Scanner scanner **=** **new** Scanner**(**System**.**in**);**

Response **=** scanner**.**nextLine**();**

**}**

Dentro de nuestra clase client tenemos otro método llamado connectClient el cual nos sirve para conectar a nuestro cliente, primetamente debemos utilizar un try – catch, dentro del try lo primero que debemos realizar es crear un objeto tipo Socket el cual recibe una ip o bien localhost y el puerto al cual será conectado, posteriormente obtenemos nuestros flujo de datos de salida y enviamos esta respuesta utilizando el método writeUTF(), acontinuación obtenemos nuestros flujos de entrada e imprimimos en consola la respuesta del servidor utilizando el método readUTF(), finalmente cerramos la conexión del Sockets cliente, esto se realiza como ya fue mencionado en el bloque try. En el caso del catch solo mostramos la Excepción que ocurra, en el caso de que sea de tipo IOExeption

public void connectClient**(){**

**try{**

socket **=** **new** Socket**(**"localhost"**,** 5000**);**

outputStream **= new** DataOutputStream**(**socket**.**getOutputStream**());**

outputStream**.**writeUTF**(**Response**);**

inputStream **=** **new** DataInputStream**(**socket**.**getInputStream**())**

System**.**out**.**println**(**"Server says: " **+** inputStream**.**readUTF**());**

socket**.**close**();**

**}** **catch(**IOException e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

Para finalizar con esta clase debemos ejecutar la aplicación esto se hace en el método main. Dentro del main tenemos un ciclo do – while() infinito y dentro de este ciclo creamos un objeto cliente y utilizamos el método connectClient()

public static void main**(**String**[]** args**){**

**do{**

**new** client**().**connectClient**();**

**}** **while(true);**

**}**

Ahora tenemos la clase server, en la cual toda nuestra implementación está dentro del método main en la cual tenemos primeramente nuestras variables de tipo: Socket, DataInputStream, DataOutputStream, ServerSocket un arreglo de Strings llamado Questions, que es idéntico al de nuestra clase client y otro arreglo de tipo Answers en el que están contenidas las respuestas de nuestras preguntas.

public class server **{**

public static void main**(**String args**[])** **{**

Socket client**;**

DataOutputStream outputSream**;**

DataInputStream inputStream**;**

ServerSocket serverSocket**;**

String Questions**[]** **=** **{**"Como te llamas"**,**

"Cual es tu color favorito"**,**

"Cuantos anos tienes"**,**

"Cual es tu comida favorita"**,**

"Cual es tu lugar favorito"**,**

"Cual es tu musica favorita"**,**

"Tienes hermanos"**,**

"Como se llama tu papa"**,**

"Como se llama tu mama"**,**

"Que te gusta hacer"**},**

Answers**[]** **=** **{**"Erick Efrain"**,**

"Negro"**,** "19"**,** "Los tacos"**,** "Mi habitacion"**,**

"Rock"**,** "Si"**,** "Efrain"**,** "Mireya"**,**  "En mi tiempo libre suelo dormir"**};**

**...**

**}**

**}**

Posteriormente utilizamos al igual que en nuestra clase client un bloque try en el cual creamos un nuevo objeto de tipo ServerSocket el cual recibe su constructor el puerto en el cual estará conectado, posteriormente un ciclo while el cual es infinito y dentro de este ciclo se obtiene un objeto tipo Socket el cual es el enlace entre nuestro servidor y nuestro cliente esto se realiza usando el método accept() el cual está contenido en la clase ServerSocket. Una vez hecho lo anterior obtenemos nuestros flujos de entrada y salida, para después utilizar una variable tipo String llamada request en la cual leemos lo que hay en nuestro objeto que contiene al flujo de entrada (DataInputStream) esto se hace utilizando el método readUTF().

Continuando tenemos un ciclo for que tiene como propósito recorrer el arreglo que contiene a nuestras preguntas y dentro de este ciclo debemos comparar lo que se ha obtenido en nuestro flujo de entrada con lo que hay en el arreglo que contiene a nuestras preguntas y utilizando el método writeUTF() que esta contenido en la clase DataOutputStream además el argumento que le pasaremos a writeUTF() será una cadena de nuestro arreglo de Strings llamado Answers.

Posteriormente cerramos la conexión de nuestro Socket cliente y finalmente si ha habido algún error tenemos el bloque catch el cual mostrará en caso de que sea necesario la excepción que se ha atrapado.

public class server **{**

public static void main**(**String args**[])** **{**

...

**try{**

serverSocket **=** **new** ServerSocket**(**5000**);**

System**.**out**.**println**(**"Waiting for a connection"**);**

**while(true){**

client **=** serverSocket**.**accept**();**

System**.**out**.**println**(**"Client connected"**);**

outputSream **=** **new** DataOutputStream**(**client**.**getOutputStream**());**

inputStream **=** **new** DataInputStream**(**client**.**getInputStream**());**

String request **=** inputStream**.**readUTF**();**

**for** **(**int i**=**0**;** i**<**Questions**.**length**;** i**++)** **{**

**if** **(**request**.**equals**(**Questions**[**i**]))**

outputSream**.**writeUTF**(**Answers**[**i**]);**

**}**

System**.**out**.**println**(**"Closing connection"**);**

client**.**close**();**

**}**

**}** **catch(**Exception e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

**}**

**Conclusión:**

Al igual que en los sockets clientes podemos realizar la conexión de programas diferentes en diferentes máquinas, pero al conectarlas a un servidor podemos enviar y recibir datos de diferentes computadoras, procesar estos datos y enviar respuestas ya sea a una máquina en específico o bien a diferentes computadoras.