Comunicação via Socket em Java

Sham Vinicius Fiorin

Redes de Computadores

Professor Jean Carlos

Índice

Comunicação via Socket em Java....1
Link Github....1
Servidor....1
Classe Principal Servidor....1
Classe Servidor....2
Classe Tratar Cliente....4
Cliente....4
Classe Principal Cliente....4
Classe Cliente....5
Classe Recebedor....6
Bibliografia....7

Link Github

https://github.com/skatesham/Chat-Socket-Java

Servidor

Classe Principal Servidor

```
package socketserver;

import java.io.IOException;

/**
 *
 * @author shan
 */
public class SocketServer {
```

```
    /**
     * Função iniciar o servidor
     *
     * @param port
     */
    public static void conectarServidor(int port) {
        ServidorSocket server = new ServidorSocket(port);
        try {
            server.execute();
        } catch (IOException ioe) {
            System.err.println("Erro ao conectar servidor: " + ioe.getMessage());
        }
    }
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        //Executar servidor socket
        conectarServidor(12345);
    }
}
```

Classe Servidor

```
package socketserver;

import java.io.Closeable;
import java.io.IOException;
import java.io.PrintStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
import java.util.Scanner;

/**
 *
 * @author shan
 */
public class ServidorSocket {

    //private static final Logger log =
Logger.getLogger(LocalShell.class.getName());
    int PORT;

    ServerSocket servidor;
    Socket cliente;
    Scanner scaner;
    List<PrintStream> saidasClientes;

    public ServidorSocket(int port) {
```

```
        this.PORT = port;
        saidasClientes = new ArrayList<>();

    }

    public void execute() throws IOException {

        String ip;

        //Abrindo Porta Servidor
        servidor = new java.net.ServerSocket(PORT);
        System.out.println("Porta 12345 aberta!");

        while (true) {

            //Aceitando conexão cliente
            cliente = servidor.accept();
            //ip cliente
            ip = cliente.getInetAddress().getHostAddress();
            //aviso conexão
            System.out.println("Nova conexão com o cliente " + ip);

            //adicionar saida do cliente a lista de clientes
            PrintStream ps = new PrintStream(cliente.getOutputStream());
            saidasClientes.add(ps);

            //tratamento de clientes em uma nova Thread
            TrataCliente tc = new TrataCliente(ip, cliente.getInputStream(), this);
            new Thread(tc).start();
        }
    }
    public void distribuiMensagem(String msg) {
        for (PrintStream cli : saidasClientes) {
            cli.println(msg);
            System.out.println(msg);
        }
    }
    private void secureClose(final Closeable resource) {
        try {
            if (resource != null) {
                resource.close();
            }
        } catch (IOException ex) {
            System.out.println("Erro = " + ex.getMessage());
        }
    }
    public void encerrar() {
        secureClose(servidor);
        secureClose(cliente);
        secureClose(scaner);
        System.exit(2);
    }
```

```
}
```

Classe Tratar Cliente

```
package socketserver;

import java.io.InputStream;
import java.util.Scanner;
/**
 * @author shan
 */
public class TrataCliente implements Runnable {
    InputStream cliente;
    ServidorSocket servidor;

    public TrataCliente(String host, InputStream cliente, ServidorSocket servidor)
{
        this.cliente = cliente;
        this.servidor = servidor;
    }
    @Override
    public void run() {
        Scanner scan = new Scanner(this.cliente);
        while(scan.hasNextLine()){
            servidor.distribuiMensagem(scan.nextLine());
            //System.out.println(host+": "+scan.nextLine());
        }

    }
}
```

Cliente

Classe Principal Cliente

```
package socketclient;

import java.io.IOException;

/**
 *
 * @author shan
 */
public class SocketClient {
    /**
     * Iniciar Cliente para conexão com servidor
     *
     * @param host
     * @param port
```

```
     */
    public static void executarCliente(String host, int port) {
        ClienteSocket cliente = new ClienteSocket(host, port);
        try {
            cliente.executar();
        } catch (IOException ioe) {
            System.out.println("Problemas ao executar cliente: " +
ioe.getMessage());
        }
    }
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        //iniciando cliente
        executarCliente("192.168.1.15", 12345);
    }
}
```

Classe Cliente

```
package socketclient;

import java.io.Closeable;
import java.io.IOException;
import java.io.PrintStream;
import java.net.Socket;
import java.util.Scanner;

/**
 *
 * @author shan
 */
public class ClienteSocket {

    private int PORT = 12345;
    private String HOST = "192.168.1.15";
    private Socket cliente;
    private PrintStream saida;
    private Scanner scaner;
    private Recebedor r;

    public ClienteSocket(String host, int port) {
        this.HOST = host;
        this.PORT = port;
    }
    public void executar() throws IOException {
        String str;
        //Conectando o cliente
        cliente = new Socket(HOST, PORT);
        System.out.println("Conectado Servidor = " + HOST + ":" + PORT);
```

```
        //Tread receber Mensagem do servidor
        r = new Recebedor(cliente.getInputStream());
        new Thread(r).start();
        //Enviar Texto via PrintStream no outputStram do cliente
        scaner = new Scanner(System.in);
        saida = new PrintStream(cliente.getOutputStream());
        while (scaner.hasNextLine()) {
            str = scaner.nextLine();
            if (str.equalsIgnoreCase("sair")) {
                encerrar();
            } else {
                saida.println(cliente.getInetAddress().getHostAddress() + ": " +
str);
            }
        }
    }
    private void secureClose(final Closeable resource) {

        try {
            if (resource != null) {
                resource.close();
            }
        } catch (IOException ex) {
            System.out.println("Erro = " + ex.getMessage());
        }
    }
    public void encerrar() {
        secureClose(saida);
        secureClose(cliente);
        secureClose(scaner);
        System.exit(2);
    }
}
```

Classe Recebedor

```
package socketclient;

import java.io.InputStream;
import java.util.Scanner;

/**
 * @author shan
 */
public class Recebedor implements Runnable{
    InputStream servidor;

    public Recebedor(InputStream servidor) {
        this.servidor = servidor;
    }
    @Override
    public void run(){
```

```
        //Recebe Mesagens do servidor e imprime na tela
        Scanner s = new Scanner(this.servidor);
        while(s.hasNextLine()){
            System.out.println(s.nextLine());
        }
    }
}
```

Bibliografia

Caelum; Apêndice – Sockets; Apostila Java e Orientação a Objetos; Capítulo 19; Disponível em: https://www.caelum.com.br/apostila-java-orientacao-objetos/apendice-sockets/. Acesso em: 19 de junho de 2018.