Prática - Sockets 2

Sistemas Distribuídos e Mobile

Prof. Me. Gustavo Torres Custódio
gustavo.custodio@anhembi.br

Exemplo de Conexão TCP Persistente

```java
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;

public class Conexao {

    public static String receber(Socket socket) throws IOException {
        InputStream in = socket.getInputStream();
        byte infoBytes[] = new byte[100];
        int bytesLidos = in.read(infoBytes);

        if (bytesLidos > 0) {
            return new String(infoBytes);
        }else {
            return "";
        }
    }

    public static void enviar(Socket socket, String textoRequisicao) throws
         IOException {
        OutputStream out = socket.getOutputStream();
        out.write(textoRequisicao.getBytes());
        out.flush(); // limpando a saida de dados
    }
}
```

Exemplo de Conexão TCP Persistente

```java
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;

public class Server {
    Socket socketClient;
    ServerSocket serversocket;

    public boolean connect() {
        try {
            socketClient = serversocket.accept(); // fase de conexao
            return true;
        }catch (IOException e) {
            System.err.println("Nao fez conexao" + e.getMessage());
            return false;
        }
    }

    public static void main(String[] args) {
        try {
            Server servidor = new Server();
            servidor.rodarServidor();
        }catch(Exception e){
            e.printStackTrace();
        }
    }
```

Exemplo de Conexão TCP Persistente

```
    public void rodarServidor() throws Exception {
        String textoRecebido = "";
        String textoEnviado = "Olá, Cliente";
        Scanner input = new Scanner(System.in);

        serversocket = new ServerSocket(9600);
        System.out.println("Servidor iniciado!");

        boolean conectado = connect();

        while (conectado && !textoRecebido.trim().equals("fim")) {

            textoRecebido = Conexao.receber(socketClient);

            System.out.println("Cliente enviou: " + textoRecebido);
            System.out.print("\nDigite a sua mensagem: ");// fase de dados
            textoEnviado = input.nextLine();
            Conexao.enviar(socketClient, textoEnviado);
        }
        socketClient.close();
    }

}
```

Exemplo de Conexão TCP Persistente

```java
import java.io.InputStream;
import java.io.OutputStream;
import java.io.IOException;
import java.net.Socket;
import java.util.Scanner;

public class Client {
    Socket socket;

    public void comunicarComServidor() throws Exception {
        String textoRequisicao = "Conexao estabelecida";
        String textoRecebido = "";
        socket = new Socket("localhost", 9600);

        Scanner input = new Scanner(System.in);
        while (!textoRecebido.trim().equals("fim")) {
            System.out.print("\nDigite a sua mensagem: ");
            textoRequisicao = input.nextLine();

            // Enviar mensagem para o servidor
            Conexao.enviar(socket, textoRequisicao);

            // Receber mensagem do servidor
            textoRecebido = Conexao.receber(socket);

            System.out.println("Servidor enviou: " + textoRecebido);
        }
    }
```

Protocolo de Consulta de Passagens Aéreas

- reqConsulta:
  - assento: int
  - codigoVoo: String
- respConsulta:
  - 0: vôo disponível
  - 1: assento indisponível
  - 2: assento inexistente
  - 3: vôo inexistente

Protocolo de Consulta de Passagens Aéreas

- O cliente envia a informação do vôo que está buscando, na forma do número do vôo e do assento;
- Para simplificar, vamos concatenar essas duas infomaçãoes usando ponto-e-vírgula.
  - Exemplo:
    - Código vôo: A2;
    - Assento: 10;
    - Resultado: “A2;10”

Protocolo de Consulta de Passagens Aéreas

- Agora vamos implementar esse protocolo:

Protocolo de Consulta de Passagens Aéreas

- Vamos criar 3 classes além do cliente e do servidor:
  - Assento;
  - Vôo;
  - Controlador.

Assento.java

```java
public class Assento {
    private int numero;
    private boolean disponivel;

    public Assento(int numero) {
        this.disponivel = true;
        this.numero = numero;
    }

    public int getNumero() {
        return numero;
    }

    public boolean getDisponivel() {
        return disponivel;
    }

    public String toString() {
        return "Assento número " + this.numero;
    }
}
```

Voo.java

```java
import java.util.ArrayList;

public class Voo {
    public final int NUM_ASSENTOS = 50;
    private String codigoVoo;
    private ArrayList<Assento> assentos = new ArrayList<>();

    public Voo(String codigoVoo) {
        this.codigoVoo = codigoVoo;
        // Adicionando 50 assentos
        for (int numero = 1; numero <= NUM_ASSENTOS; numero++) {
            Assento assento = new Assento(numero);
            assentos.add(assento);
        }
    }
```

Voo.java

```
    public String getCodigoVoo() {
        return codigoVoo;
    }

    public ArrayList getAssentos() {
        return assentos;
    }


    public Assento procurarAssento(int numeroAssento) {
        for (Assento a : assentos) {
            if (a.getNumero() == numeroAssento) {
                return a;
            }
        }
        return null;
    }
}
```

ControladorVoo.java

```java
import java.util.ArrayList;

public class ControladorVoo {
    private ArrayList<Voo> voos = new ArrayList<>();
    public final int NUM_VOOS = 10;

    public ControladorVoo() {
        for (int numero = 1; numero <= NUM_VOOS; numero++) {
            String codigoVoo = "A" + numero;
            Voo voo = new Voo(codigoVoo);
            voos.add(voo);
        }
    }
```

ControladorVoo.java

```
    public int verificarStatus(String codigoVoo, int numeroAssento) {
        // Se o voo não foi encontrado, codigo de erro 3
        Voo voo = procurarVoo(codigoVoo);
        if (voo == null) {
            return 3;
        }
        Assento assento = voo.procurarAssento(numeroAssento);
        if (assento == null) {
            return 2;
        }
        if (!assento.getDisponivel()) {
            return 1;
        }
        return 0;
    }

    public Voo procurarVoo(String codigoVoo) {
        // procura o voo enviado, se nada for encontrado, retorna null
        for(Voo v: voos) {
            // Verifica se o codigo bate com o que foi procurado
            if(v.getCodigoVoo().equals(codigoVoo)) {
                return v;
            }
        }
        return null;
    }
}
```

Conexao.java

```java
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;

public class Conexao {

    public static String receber(Socket socket) throws IOException {
        InputStream in = socket.getInputStream();
        byte infoBytes[] = new byte[100];
        int bytesLidos = in.read(infoBytes);

        if (bytesLidos > 0) {
            return new String(infoBytes);
        }else {
            return "";
        }
    }

    public static void enviar(Socket socket, String textoRequisicao) throws
        IOException {
        OutputStream out = socket.getOutputStream();
        out.write(textoRequisicao.getBytes());
    }
}
```

Client.java

```
public class Client {
    Socket socket;

    public void comunicarComServidor() throws Exception {
        String textoRequisicao = "Conexao estabelecida";
        String respostaServer = "";
        socket = new Socket("localhost", 9600);

        Scanner input = new Scanner(System.in);
        textoRequisicao = lerInformacaoPassagem(input);
        // Enviar mensagem para o servidor
        Conexao.enviar(socket, textoRequisicao);

        // Receber mensagem do servidor
        respostaServer = Conexao.receber(socket);
        int codigoResposta = Integer.parseInt(respostaServer.trim());
        mostrarResultado(codigoResposta);
    }

    public String lerInformacaoPassagem(Scanner input) {
        System.out.print("\nDigite o codigo do voo: ");
        String voo = input.nextLine();
        System.out.print("Digite o assento: ");
        int assento = input.nextInt();
        return voo + ";" + assento;
    }
```

Client.java

```
public void mostrarResultado(int respostaServer) {
    switch(respostaServer) {
        case 0:
            System.out.println("Voo disponível");
            break;
        case 1:
            System.out.println("Assento indisponível");
            break;
        case 2:
            System.out.println("Assento inexistente");
            break;
        default:
            System.out.println("Voo inexistente");
            break;
    }
}

public static void main(String[] args) {
    try {
        Client cliente = new Client();
        cliente.comunicarComServidor();
    }catch(Exception e){
        e.printStackTrace();
    }
}
```

Server.java

```
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;

public class Server {
    Socket socketClient;
    ServerSocket serversocket;
    ControladorVoo controlador = new ControladorVoo();

    public boolean connect() {
        try {
            socketClient = serversocket.accept(); // fase de conexao
            return true;
        }catch (IOException e) {
            System.err.println("Nao fez conexao" + e.getMessage());
            return false;
        }
    }

    public int calcularCodigoStatus(String textoRecebido) {
        String[] codigoEAssento = textoRecebido.split(";");
        String voo = codigoEAssento[0];
        int assento = Integer.parseInt(codigoEAssento[1].trim());
        return controlador.verificarStatus(voo, assento);
    }
```

Server.java

```
    public void rodarServidor() throws Exception {
        String textoRecebido = "";
        int codigoStatus = 0;
        Scanner input = new Scanner(System.in);

        serversocket = new ServerSocket(9600);
        System.out.println("Servidor iniciado!");

        while(true) {
            if (connect()) {
                textoRecebido = Conexao.receber(socketClient);
                System.out.println("Cliente enviou: " + textoRecebido);
                codigoStatus = calcularCodigoStatus(textoRecebido);

                Conexao.enviar(socketClient, Integer.toString(codigoStatus));
                socketClient.close();
            }
        }
    }
    public static void main(String[] args) {
        try {
            Server servidor = new Server();
            servidor.rodarServidor();
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

Protocolo de Consulta de Passagens Aéreas

- Agora modifique o programa para que o cliente continue conectado.

Conteúdo

https://gustavotcustodio.github.io/sdmobile.html

Referências

- Sistemas Distribuídos: Conceitos e Projetos - Coulouris
  - Cap 4.

Obrigado

gustavo.custodio@anhembi.br