

Linguagem de Programação Orientada a Objetos II

Sockets

Prof. Tales Bitelo Viegas

<https://fb.com/ProfessorTalesViegas>



Sockets

- ▶ Um Socket representa uma conexão entre dois hosts.
- ▶ Através dele é possível realizar as seguintes operações :
 - Conectar a uma máquina remota
 - Enviar dados
 - Receber dados
 - Fechar uma conexão
 - Conectar a uma porta
 - Escutar dados que estão chegando por uma porta
 - Aceitar conexões de máquinas remotas em uma determinada porta



Introdução

- ▶ O pacote **java.net** contém classes para implementar comunicação através da rede
- ▶ A comunicação via protocolo **TCP** (Transfer Control Protocol):
 - Socket (soquete de dados)
 - ServerSocket (soquete do servidor).
- ▶ A comunicação via **UDP** (Unreliable Datagram Protocol) :
 - DatagramSocket (soquete de dados UDP),
 - DatagramPacket (pacote UDP)
 - MulticastSocket (soquete UDP para difusão).



A classe Socket

- ▶ Utilizada tanto por clientes, quanto por servidores para realizar as seguintes operações :
 - Conectar a uma máquina remota
 - Enviar dados
 - Receber dados
 - Fechar uma conexão



A classe Socket

- ▶ Ciclo de vida:

- 1) Cria um socket usando, por exemplo, o construtor `Socket(String host, int port)`
- 2) O socket tenta conectar ao servidor
- 3) Assim que a conexão é estabelecida, cliente e servidor trocam dados através de input e output streams
- 4) Encerrada a troca de dados, a conexão é encerrada.



Criando Cliente Socket

- ▶ Construtor:
 - **Socket**(String host, int port) throws **UnknownHostException**, **IOException**.

```
Socket sock = new Socket("gravatai.ulbra.tche.br", 80)
```



A classe Socket

▶ Principais métodos:

- Métodos destinados à manipulação dos fluxos de entrada e saída:
 - `getInputStream()`
 - `getOutputStream()`
 - `close()`
- `setSoTimeout(int timeout)` : define tempo limite (milissegundos) de espera. Quando o tempo limite for atingido será gerado um `InterruptedException`.



Exemplo de Cliente

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

public class ClienteSocket {

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

            // Conecta no servidor
            Socket s = new Socket("www.ulbra.br", 80);

            // Busca streams de E/S
            Scanner entrada = new Scanner(new InputStreamReader(s.getInputStream()));
            PrintWriter saida = new PrintWriter(s.getOutputStream());

            // Envia dados atraves do Stream
            saida.println("GET / HTTP/1.1\nHost: www.ulbra.br\n");
            saida.flush();

            // Imprime retorno
            String linha;
            while ((linha = entrada.nextLine()) != null) {
                System.out.println(linha);
            }

            // Encerra recursos
            entrada.close();
            saida.close();
            s.close();

        } catch (UnknownHostException ex) {
            System.out.println("Host desconhecido");
        } catch (IOException ex) {
            System.out.println("Erro na conexao: " + ex.getMessage());
        }

    }

}
```



A classe ServerSocket

- ▶ Contém os métodos necessários para trabalhar com servidores.
- ▶ Realiza as seguintes operações:
 - Conectar a uma porta
 - Escutar dados que estão chegando por uma porta
 - Aceitar conexões de máquinas remotas em uma determinada porta



A classe ServerSocket

- ▶ Ciclo de vida:

- 1) Cria um ServerSocket em uma porta da rede, usando o construtor `ServerSocket(int port)`
- 2) O ServerSocket fica a espera de uma conexão através da porta especificada através do método `accept()` que retorna um objeto `Socket`, que conecta o cliente e o servidor.
- 3) Utiliza os métodos `getInputStream()` ou `getOutputStream()`, da classe `Socket`, para se comunicar com o cliente



A classe ServerSocket

- ▶ Ciclo de vida:

- 4) O servidor e o cliente trocam informações até o encerramento da conexão.
- 5) O servidor retorna ao passo 2, aguardando por nova conexão.



A classe ServerSocket

▶ Principais construtores:

- **ServerSocket(int port)** : cria um ServerSocket que fica escutando a porta, aguardando por um cliente.
 - Se `port==0` então usa qualquer porta livre.



A classe ServerSocket

- ▶ Principais métodos:
 - `accept()`: aguarda por uma conexão. Retorna um socket através do qual o programa poderá se comunicar com o cliente.
 - `close()` : encerra a comunicação



Exemplo de Servidor

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

public class ServidorSocket {

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

            // Instancia o servidor
            ServerSocket server = new ServerSocket(8080);

            // Conecta no servidor
            Socket s = server.accept();

            // Busca streams de E/S
            Scanner entrada = new Scanner(new InputStreamReader(s.getInputStream()));
            PrintWriter saida = new PrintWriter(s.getOutputStream());

            // Imprime requisicao
            String linha;

            while ((linha = entrada.nextLine()) != null) {
                if (linha.length() == 0) {
                    break;
                }
                System.out.println(linha);

            }

            // Envia dados atraves do Stream
            saida.println("Recebi requisicao\n");
            saida.flush();

            // Encerra recursos
            entrada.close();
            saida.close();
            s.close();

        } catch (UnknownHostException ex) {
            System.out.println("Host desconhecido");
        } catch (IOException ex) {
            System.out.println("Erro na conexao: " + ex.getMessage());
        }

    }
}
```



Curiosidades

- ▶ TELNET – porta 23
- ▶ FTP – porta 20 e 21
- ▶ SMTP – porta 25
- ▶ NNTP (Network News Transfer Protocol)– porta 119
- ▶ HTTP – porta 80

