
Java Avancé

Cours 6 : Programmation Réseau

Arsène Lapostolet

15 Février 2024

Introduction

Socket

Thread : flux d'entrée/sortie réseau.

- Même interface que les autres entrées/sortie (fichiers, console)
- Utilise les couches réseau TCP ou UDP pour communiquer avec d'autres machines
- Le cours couvre le cas des sockets TCP

Socket Java

Modèle client-serveur.

Classes :

- `ServerSocket` : se lie à un port sur la machine serveur
- `Socket` : se connecte à un port sur une IP

`ServerSocket.accept` bloque jusqu'à la connexion d'un client et retourne une `Socket` ouverte vers le client à chaque connexion.

Coté Serveur

Attente d'un client et ouverture des flux :

```
1 final var serverSocket = new ServerSocket(3000);
2 final var socket = serverSocket.accept();
3
4 final var streamFromClient = new BufferedReader(
5     new InputStreamReader(socket.getInputStream())
6 );
7
8 final var streamToClient = new
  PrintWriter(socket.getOutputStream(), true);
```

Java

Coté Client

Connexion au serveur et ouverture des flux :

```
1 final var clientSocket = new Socket(3000, "localhost");  
2  
3 final var streamFromClient = new BufferedReader(  
4     new InputStreamReader(clientSocket.getInputStream())  
5 );  
6  
7 final var streamToClient = new PrintWriter(  
8     clientSocket.getOutputStream(),  
9     true  
10 );
```

Java

Socket et threads

`println` et `readLine` sont bloquants :

- Un `println` côté client bloque jusqu'au prochain `readLine` côté serveur
- Un `readLine` côté client bloque jusqu'au prochain `println` côté serveur

Et vice-versa.

Il faut utiliser des threads !

Exemple : serveur de chat

```
1 public class ChatServer {
2     private final int port;
3     private final List<PrintWriter> connectedClients;
4
5     public ChatServer(int port) {
6         this.port = port;
7         this.connectedClients = new ArrayList<>();
8     }
9
10
11
12
```

Java


```
13 public void start() {  
14     final var serverSocket = new ServerSocket(port);  
15  
16     while (true) {  
17         final var client = serverSocket.accept();  
18  
19         connectedClients.add(  
20             new PrintWriter(client.getOutputStream(), true)  
21         );  
22  
23         new Thread(() -> listenForMessage(client))  
24             .start();  
25     }  
26 }
```

```
27 private void listenForMessage(Socket client) {
28     final BufferedReader messages = new BufferedReader(
29         new InputStreamReader(client.getInputStream())
30     );
31
32     while (true) {
33         final var message = messages.readLine();
34
35         for (final var connectedClient : connectedClients) {
36             connectedClient.println(messageFromClient);
37         }
38     }
39 }
```

Exemple : client de chat

```
1 public class ChatServer {
2     private final int port;
3     private final String address;
4     private BufferedReader messagesFromServerStream;
5     private PrintWriter messagesToServerStream, output;
6
7     public ChatClient(int port, String address, PrintWriter out) {
8         this.port = port;
9         this.address = address;
10        this.output = out;
11    }
12
```

Java

```
13 public void connect() {
14     final var socket = new Socket(address, port);
15     messagesFromServerStream = new BufferedReader(
16         new InputStreamReader(socket.getInputStream())
17     );
18
19     messagesToServerStream = new PrintWriter(
20         socket.getOutputStream(),
21         true
22     );
23
24     new Thread(this::listenForMessages).start();
25 }
26
```

```
27 private void listenForMessages() {
28     while (true) {
29         final var message = messagesFromServerStream.readLine();
30         output.println(message);
31     }
32 }
33
34 public void sendMessage(String message) {
35     messagesToServerStream.println(message);
36 }
37
38 }
```