Tout sur les SOCKETs

All About Sockets

Bachir Djafri

Dépt. Informatique
Université d'Évry Val d'Essonne
bachir.djafri@ibisc.univ-evry.fr

L3 Informatique
cours de TOA

Introduction

- URLs & URLConnections vs. Socket (high level, low level network communication)
- Applications client/serveur (sans perte d'info, dans le bon ordre, protocole)
- TCP : communication point-à-point

Introduction

- URLs & URLConnections vs. Socket (high level, low level network communication)
- Applications client/serveur (sans perte d'info, dans le bon ordre, protocole)
- TCP : communication point-à-point

Introduction

- URLs & URLConnections vs. Socket (high level, low level network communication)
- Applications client/serveur (sans perte d'info, dans le bon ordre, protocole)
- TCP : communication point-à-point

Objectifs du cours

- Les SOCKETs, c'est quoi ?
- Lecture à partir et écriture sur une SOCKET
- Exemple d'une application client/serveur

Références : https ://docs.oracle.com/javase/tutorial/networking/

Objectifs du cours

- Les SOCKETs, c'est quoi ?
- Lecture à partir et écriture sur une SOCKET
- Exemple d'une application client/serveur

Références : https ://docs.oracle.com/javase/tutorial/networking/

Bachir Djafri (Dépt. Informatique) Tout sur les SOCKETs Octobre 2018 3 / 8

Objectifs du cours

- Les SOCKETs, c'est quoi ?
- Lecture à partir et écriture sur une SOCKET
- Exemple d'une application client/serveur

Références : https ://docs.oracle.com/javase/tutorial/networking/

Objectifs du cours

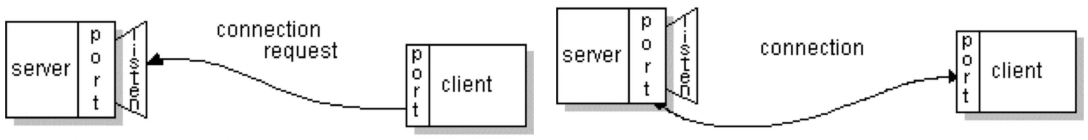
- Les SOCKETs, c'est quoi ?
- Lecture à partir et écriture sur une SOCKET
- Exemple d'une application client/serveur

Références : https ://docs.oracle.com/javase/tutorial/networking/

C'est quoi une Socket ?

Définition

A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent.



java.net Package

La classe `java.net.Socket` : coté client (et serveur)

La classe `java.net.ServerSocket` : coté serveur (écoute/accèpte/connecte)

Exemple : Client

EchoClient.java

```
import java.io.*;
import java.net.*;
public class EchoClient {
   public static void main(String[] args)
                              throws IOException {

        . . .

   }
}
```


Exemple : Client

EchoClient.java

```
public static void main(String[] args)
                              throws IOException {
   // partie déclaration

   Socket echoSocket = null;

   PrintWriter out = null;

   BufferedReader in = null;

        . . .
}
```

Exemple : Client

EchoClient.java

```
try {
    echoSocket = new Socket("eagles", 7);
    out = new PrintWriter(
          echoSocket.getOutputStream(), true);
    in = new BufferedReader(new
   InputStreamReader(echoSocket.getInputStream()))
} catch (UnknownHostException e) {
    System.err.println("Host eagles...");
    System.exit(1);
} catch (IOException e) {
    System.err.println("I/O error");
    System.exit(1);
```


Exemple : Client

EchoClient.java

```
...
BufferedReader stdIn = new BufferedReader(
new InputStreamReader(System.in));
String userInput;

while ((userInput = stdIn.readLine()) != null)
{
  out.println(userInput);
  System.out.println("echo: "+in.readLine());
}
```

Exemple : Client

EchoClient.java

```
...

// l'ordre est important

out.close();
in.close();
stdIn.close();
echoSocket.close();
```


Étapes à respecter

1. Création et ouverture d'une socket
2. Ouverture des flux en lecture et en écriture depuis et vers la socket
3. (selon les clients) Lecture et écriture sur les flux selon le protocole du serveur
4. Fermeture des flux
5. Fermeture de la socket

Étapes à respecter

1. Création et ouverture d'une socket
2. Ouverture des flux en lecture et en écriture depuis et vers la socket
3. (selon les clients) Lecture et écriture sur les flux selon le protocole du serveur
4. Fermeture des flux
5. Fermeture de la socket

Étapes à respecter

1. Création et ouverture d'une socket
2. Ouverture des flux en lecture et en écriture depuis et vers la socket
3. (selon les clients) Lecture et écriture sur les flux selon le protocole du serveur
4. Fermeture des flux
5. Fermeture de la socket

Étapes à respecter

1. Création et ouverture d'une socket
2. Ouverture des flux en lecture et en écriture depuis et vers la socket
3. (selon les clients) Lecture et écriture sur les flux selon le protocole du serveur
4. Fermeture des flux
5. Fermeture de la socket

Étapes à respecter

1. Création et ouverture d'une socket
2. Ouverture des flux en lecture et en écriture depuis et vers la socket
3. (selon les clients) Lecture et écriture sur les flux selon le protocole du serveur
4. Fermeture des flux
5. Fermeture de la socket

Bachir Djafri (Dépt. Informatique) Tout sur les SOCKETs Octobre 2018 7 / 8

Coté serveur

Création d'une socket serveur.

Serveur.java

```
try {

    serverSocket = new ServerSocket(4444);

} catch (IOException e) {

    System.out.println("Pb sur le port 4444");
    System.exit(-1);

}
```


Coté serveur

Acceptation d'un connextion.

Serveur.java

```
Socket clientSocket = null;

try {

    clientSocket = serverSocket.accept();
   // écoute, appel bloquant

} catch (IOException e) {
    System.out.println("echec de connection");
    System.exit(-1);
```

Coté serveur

Création des flux de communication.

Serveur.java

```
PrintWriter out = new PrintWriter(
      clientSocket.getOutputStream(), true);

BufferedReader in = new BufferedReader(
      new InputStreamReader(
            clientSocket.getInputStream()));
```


Coté serveur

La communication.

Serveur.java

```
String inputLine, outputLine;
TheProtocol proto = new TheProtocol();
outputLine = proto.processInput(null);
out.println(outputLine);
while ((inputLine = in.readLine()) != null) {
   outputLine = proto.processInput(inputLine);
   out.println(outputLine);
   if (outputLine.equals("Quit"))
      break;
}
```

Coté serveur

Pour finir.

Serveur.java

```
out.close();
in.close();
clientSocket.close();
serverSocket.close();
```


Coté serveur

Supporter plusieurs clients.

Serveur.java

```
while (true) {
   acceptation d'une connection ;
   création d'un thread pour dialoguer
       (traiter) avec le client ;
}
```