SCIENCES SORBONNE UNIVERSITÉ

SRCS : Systèmes Répartis Client/Serveur

TME 3 : Protocole HTTP

Objectifs pédagogiques

- Socket Java
- Protocole applicatif
- Communication réseau

L'objectif de ce TP est dans un premier temps de comprendre les informations qui transitent entre un serveur et un client HTTP et dans un deuxième temps de programmer un mini-serveur web.

Exercice 1 – Contenu des messages

Le but de cet exercice est de comprendre les informations véhiculées par le protocole HTTP entre un client et un serveur. Soit la classe `Serveur` et soit l'interface `RequestProcessor` suivantes qui serviront de base pour afficher les requêtes clientes : le serveur ouvre une socket d'écoute, appelle la fonction de traitement de l'interface et ferme la socket.

src/srcs/http/Serveur.java

```
1  package srcs.http;
2
3  import java.io.IOException;
4  import java.net.ServerSocket;
5  import java.net.Socket;
6
7  public class Serveur {
8
9    private final int port;
10   private final RequestProcessor processor;
11
12   public Serveur(int port, RequestProcessor processor) {
13     this.port=port;
14     this.processor=processor;
15   }
16
17   public void start() {
18     try(ServerSocket ss = new ServerSocket(port)){
19       int i=0;
20         while ( true ) {
21         try(Socket client = ss.accept()){
22           System.out.println("Debut de requête "+i);
23           processor.process(client);
24         }
25         System.out.println("Fin de requête "+i);
26         System.out.println("**********************\n");
27         i++;
28       }
29     } catch (IOException e) {
30       e.printStackTrace();
```

```
31      }
32    }
33  }
```

src/srcs/http/RequestProcessor.java

```
1  package srcs.http;
2
3  import java.net.Socket;
4
5  public interface RequestProcessor {
6
7    public void process(Socket connexion);
8  }
```

Question 1
Écrire une classe qui implante l'interface `RequestProcessor` et qui envoie tout octet lu sur la socket vers le flux de sortie standard (il est possible d'utiliser la méthode `bind` vue en TD) et ajouter une méthode `main` permettant d'instancier et de démarrer le serveur avec cette classe.

Question 2
Pour tester le serveur, le lancer et ouvrir un navigateur. Dans la navigateur taper dans la barre de recherche `http://localhost:`*`le_port_choisi`*`/index.html`. Quelle sont les informations transmises ? Quel est leur format ? Comment peut-on savoir que le client ai fini de transmettre ses informations ?

Question 3
On peut remarquer que la requête n'est pas terminée car l'affichage de fin ne s'est pas produit. Pourquoi ? Comment peut-on débloquer la situation sans toucher au serveur ?

Question 4
Écrire une nouvelle classe qui implante l'interface `RequestProcessor` mais qui, cette fois-ci, lit ligne par ligne le flux d'entrée et s'arrête jusqu'à lire une ligne vide. Tester cette version. Que remarque-t-on ?

Question 5
Pour voir les informations transmises d'un serveur HTTP à un client, programmer un client qui interroge l'url `http://www.google.fr/index.html` (pour rappel un serveur http écoute sur le port 80) et qui envoie tout octet reçu sur la socket vers le flux de sortie standard. Pour simplifier on n'enverra pas d'en-tête optionnel. Quel est le format de la réponse du serveur ?

Question 6
Refaire la même chose avec l'url `http://www.google.fr/toto.html`. Quelle est la réponse du serveur ?

Question 7
Réinterroger le serveur en introduisant volontairement une erreur dans la requête (exemple : indiquer une version d'HTTP qui n'existe pas). Quelle est la réponse du serveur ?

Exercice 2 – Serveur HTTP

Question 1
Écrire une classe implantant `RequestProcessor` et qui traite les commandes *GET* :

- Vous prendrez soin de consommer tout le flux d'entrée (c'est à dire lire les en-têtes) avant de renvoyer la réponse.
- Tout autre commande que GET devra mener à une erreur "400 Bad Request"
- Si le fichier demandé n'existe pas ou si c'est un répertoire, le serveur enverra l'erreur "404 Not Found" (voir la doc de la classe `java.io.File`)
- N'oubliez pas que tout message HTTP se compose d'une ligne vide à la fin de son entête.
- Dans l'en-tête de la réponse on ne précisera pas d'option
- Pour rappel la racine de la requête HTTP correspond au répertoire courant du serveur. Vous pouvez connaître ce répertoire avec `System.getProperty("user.dir")`. Sous Eclipse, le répertoire courant de votre programme est la racine du projet Eclipse.

Question 2

Tester le serveur avec un navigateur en requêtant le fichier `hello.html` dont le contenu est le suivant :

```
<html>
  <head> <title>Hello Page</title> </head>
  <body> Bonjour !<br/> </body>
</html>
```