SCIENCES SORBONNE UNIVERSITÉ

SRCS : Systèmes Répartis Client/Serveur

TME 3 : Serveur Web

Objectifs pédagogiques :

- **Socket Java**
- **Protocole applicatif**
- **Communication réseau**

L'objectif de ce TP est de mettre en œuvre l'API socket du langage Java au travers la programmation d'un serveur Web. Vous respecterez le protocole HTTP dont la RFC a été donné en TD. Pour les requêtes de type GET, il sera inutile de coder un client dans ce TME, car c'est votre navigateur web habituel qui jouera pleinement ce rôle.

Exercice 1 – Serveur générique

Dans un premier temps nous allons construire un serveur générique (adaptable à n'importe quel protocole ou service) permettant de traiter les requêtes de ses clients. L'idée ici est de factoriser le code permettant de créer la socket d'écoute et de créer un objet traitant la requête à chaque nouvelle connexion. Ceci vous facilitera la tâche par la suite car vous aurez juste à coder la partie spécifique de l'exercice.

Question 1

Écrire une interface `Request` qui propose une méthode `void execute(Socket connexion)` qui définira le traitement de la requête pour une connexion de client donnée.

Question 2

Écrire une classe `Serveur` qui définira le code du serveur. Cette classe sera composée :

- de deux attributs déclarés `final` :
  - ◇ le port d'écoute (type `int`)
  - ◇ la classe concrète permettant de traiter une requête, c'est-à-dire une classe qui implante `Request`. On utilisera ici la réflexivité de java.
- un constructeur permettant d'initialiser les deux attributs
- une méthode `listen()` qui :

1. instanciera la socket d'écoute
2. se mettra en attente sur la socket d'écoute,
3. à la réception d'une requête, instanciera l'objet typé par `Request` qui traitera la requête.
4. appellera la méthode `execute`
5. et reviendra à l'étape 2. (boucle infinie)

Exercice 2 – Affichage d'une requête cliente HTTP

Question 1

Écrire une classe `PrintRequestHTTP` qui implante `Request` et qui permet d'afficher sur la sortie standard le contenu complet de la requête HTTP provenant du client.

Question 2

Dans la classe `PrintRequestHTTP` ajoutez la méthode `main` suivante :

```
public static void main (String args[]) throws Exception {
       new Serveur(4343, PrintRequestHTTP.class).listen();
}
```

Question 3

Pour tester votre code, ouvrez votre navigateur et indiquez *localhost :4343* dans la barre de recherche.

Exercice 3 – Traitement de requêtes *Get*

Question 1

Écrire une classe `ProcessRequestGet` qui étend `Request` et qui permet de traiter 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"
- N'oubliez pas que tout message HTTP se compose d'une ligne blanche à la fin de son entête.
- 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

Testez votre serveur depuis votre navigateur en requêtant le fichier `coucou.html` dont le contenu est le suivant :

```
<html>
  <head> <title>Coucou</title> </head>
  <body> Bonjour !<br/> </body>
</html>
```

Exercice 4 – Serveur Web avec commandes *GET* et *PUT*

Question 1

Reprendre la classe `ProcessRequestGet` pour écrire une classe `ProcessRequestGetPut` afin de pouvoir traiter maintenant les requêtes *PUT*.

Question 2

Écrire un client permettant de tester la fonctionnalité *PUT* de votre serveur.