# Cours Applications Web

**TP1**

Adil El Ghali

3 Fvrier 2009

## 1 Premiers pas avec HTTP

Comme nous l'avons vu en cours HTTP est un protocole en mode texte, ce qui veut dire que l'on peut envoyer des requêtes aux serveurs HTTP sans passer par un navigateur. Le but de cette première partie de TP est de vous familiariser avec le format des requêtes et des réponses HTTP[1].

### 1.1 Client HTTP

Vous allez envoyer des requêtes entièrement écrites à la main au serveur, en d'autres termes vous allez simuler un client HTTP.

Pour réaliser cette première étape, vous aurez besoin d'un terminal (xterm, gnome-terminal ou konsole). Lancez le terminal de votre choix et au prompt taper la commande :

telnet www.unice.fr 80 (1)

Le serveur vous répondra:

```
# telnet www.unice.fr 80
Trying 134.59.1.71...
Connected to www.unice.fr.
Escape character is '^]'.
```

Vous pouvez alors commencer à envoyer des requêtes au serveur, par exemple:

```
GET / HTTP/1.0
Accept: www/source
Accept: text/html
```

[1]Pendant cette séance vos références principales seront les RFCs du protocole HTTP:
**RFC1945**: `http://www.w3.org/Protocols/rfc1945/rfc1945` et
**RFC2616**: `http://www.w3.org/Protocols/rfc2616/rfc2616.html`

Le serveur vous va vous retourner une réponse et se déconnecter. Pour envoyer d'autres commandes, vous devez vous reconnecter (1).

**Question 1** Quelles sont les caractéristiques du serveur web de l'université?

**Question 2** Quelles informations tirez-vous de la première réponse du serveur?

**Question 3** Essayez la requête suivante sur `portail.unice.fr` :

```
GET /jahia/jsp/index.jsp HTTP/1.0
Accept: www/source
Accept: text/html
Host: portail.unice.fr
```

**Question 4** Que vous apprennent les en-têtes de la réponse du serveur?

**Question 5** Essayez de provoquer sur le serveur de l'université des réponses ayant les différents status possibles du protocole HTTP?

### 1.2 Serveur HTTP

On va maintenant changer de rôle et jouer au serveur HTTP, pour cela on va utiliser la commande nc (nc pour netcat: un cat sur le réseau ;-)[2]:

```
usage: nc [-46DdhklnrStUuvzC] [-i interval] [-p source_port]
          [-s source_ip_address] [-T ToS] [-w timeout] [-X proxy_version]
          [-x proxy_address[:port]] [hostname] [port[s]]
```

Nc vous permet de simuler un serveur en utilisant l'option -l, par exemple pour lancer un serveur qui écouter sur le port 12345, vous pouvez utiliser la commande suivante:

$$\text{nc} \ -\text{l localhost 12345} \tag{2}$$

et pour vous connectez á ce serveur vous pouvez utiliser la commande:

$$\text{telnet localhost 12345} \tag{3}$$

ou utiliser un navigateur `http://localhost:12345` ...

**Question 6** Connectez vous au serveur en utilisant le navigateur de votre choix, quelles sont les informations que votre navigateur envoie aux serveurs web sur lesquels vous vous connectez?

**Question 7** Répondez à la requête du serveur de sorte à afficher une page comme dans la figure 1

**Question 8** Positionnez un Cookie sur le client dans votre réponse suivante.

**Question 9** Envoyez des erreurs avec différents codes de status au client.

[2]Pour plus d'informations sur nc man nc

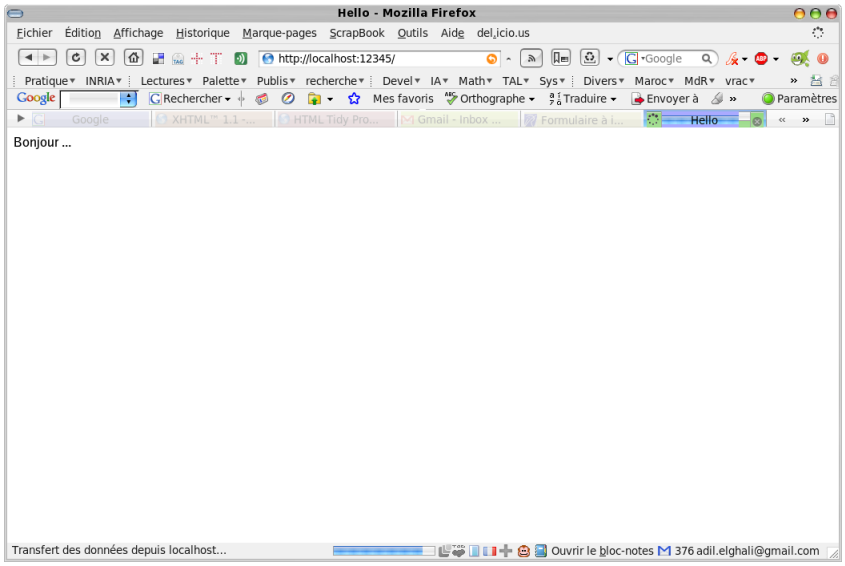


Figure 1: Exemple de page renvoyée par le serveur

## 2 Client et serveur HTTP en Java

Les fichiers Client.java, Connection.java et Server.java, qui vous sont fournis implementent un client et un serveur HTTP simples.

**Question 10** Compilez et testez ces programmes.

**Question 11** Ajoutez une fonctionnalité permettant de positionner des Cookies.

**Question 12** Ajoutez une fonctionnalité permettant de renvoyer des codes d'erreurs.

**Question 13** Ajoutez une fonctionnalité qui fait que le serveur ne répond qu'à des requêtes de votre navigateur préféré.