http://clx.asso.fr/spip/?Introduction-au-protocole-HTTP

# Introduction au protocole HTTP

- Documentations - Technique -

Date de mise en ligne : jeudi 18 juillet 2002

Date de parution : 1er janvier 0

**De nos jours, "surfer" sur le Web est un acte entré dans le quotidien de bons nombre de personnes.**
**Nous sommes de plus en plus nombreux à utiliser ce service par le biais de logiciels appelés "navigateurs", "butineurs" ou encore "browsers" pour les anglicistes.**
**Et ceci bien sûr avec grand plaisir.**

**Malheureusement,(d'aucuns diront heureusement) peu d'entre-nous savent ce qui se cache sous les clics de la souris.**

**Aussi c'est dans une suite d'articles à la technicité croissante que je me propose de vous faire découvrir la face cachée du Web.**

**Cet article, le premier volet de la série présente les bases nécessaires pour aller plus en avant.**

# INTRODUCTION

Le protocole HTTP est parmi tous les protocoles applicatifs sûrement celui dont l'utilisation est la plus courante chez les néophytes.
Un peu comme Mr Jourdain qui fait de la prose sans le savoir, tous ceux qui "surfent" sur le web utilisent le protocole HTTP.

Cet article a pour but d'entrer plus profondement dans le fonctionnement de ce fameux protocole HTTP afin de vous faire découvrir ce que fait votre navigateur préféré dans votre dos...

# PRESENTATION

Nous allons par la suite utiliser le terme *"client"* pour parler de la machine sur laquelle s'exécute le navigateur et de *"serveur"* pour la machine qui renvoie les documents voulus (par exemple `clx.anet.fr`).

De façon basique le *client* demande un document au *serveur*.
Ce document est identifié via son *URL* [1].

**Apprendre à lire une URL**

Voyons comment "lire" une URL :

– Voici un exemple simple d'URL :
`http://10function.kicks-ass.org/back.jpg`

Un URL peut être décomposée en un certain nombre de champs comme suit :

– `http` désigne le protocole utilisé. On peut aussi trouver des valeurs comme ftp, https, gopher ...
Seul http nous intéresse pour l'instant.
– `://` est un séparateur qui permet de différencier le nom du protocole de la suite de l'URL.
– `10function.kicks-ass.org` désigne le nom d'hôte de la machine sur laquelle se trouve le document requis.
Remarquez que le nom du serveur Web ou serveur HTTP peut aussi être identifié par une adresse IP comme `80.118.2.10`.
– `/` est un nouveau séparateur qui indique la fin du nom d'hôte du serveur.
– On trouve ensuite la localistation du document sur le serveur.
Ici `back.jpg` donne le nom du document.
On peut voir rapidement qu'il s'agit d'un fichier nommé back.jpg, et de par son extension qu'il s'agit sans doute d'une image.

Il arrive que l'on ait une URL un peu plus complexe comme :
`http://10function.kicks-ass.org:80/books/ruhacker.txt`
On retrouve une syntaxe proche de la précédente avec quelques "suppléments" :

– `:80` Indique le port TCP utilisé pour communiquer avec le serveur. Dans 95% des cas il s'agit du port 80 qui est le port standard pour un serveur HTTP.
Remarquez que ne pas mentionner ce numero de port indique implicitement le port 80.
– `/books/ruhacker.txt` Est le nom et l'emplacement du document voulu sur le serveur.
Il est ici un peu plus complexe que précédement car il s'agit d'un chemin complet (un path) ; il faut donc "lire" que le document est 'ruhacker.txt' et qu'il se trouve dans le répertoire /books/.

Si ce le nom du document n'est pas indiqué, et que seul le répertoire est mentionné, ce qui est très courant, le serveur renvoie un document défini comme étant le document par défaut dans son fichier de configuration [2]

# FONCTIONNEMENT

Lorsque vous voulez "naviguer" sur le web. Vous commencez en général par saisir 'l'adresse du site' [3]. (ex : `http://clx.anet.fr`).
Pour surfer, l'URL est la seule chose que vous ayez à fournir à votre navigateur dans la zone de saisie (la barre d'adresse).
Quand vous cliquez sur un lien vous remarquerez que le navigateur remplit cette zone à votre place.
Un lien n'est en fait ni plus ni moins qu'une URL vers une autre page.
Après quelques instants, le navigateur affiche le document demandé.

Mais voici ce qui se passe en réalité... en "coulisse".

– Le navigateur (ou client) analyse (on dit *parse* en franglais informatique) l'URL qui lui a été fournie et tente d'établir un session TCP [4] avec l'hôte mentionné, sur le port indiqué, et via le protocole choisi.
– Si la connexion s'effectue, le client envoie une requête HTTP vers le serveur (Le nom du document est inclu dans cette requête).
– Le serveur reçoit et analyse cette requête puis recherche le document correspondant et s'il le possède renvoie

 

l'ensemble du document précédé d'une "en-tête" de son cru vers le client.
Il ferme ensuite la session TCP.
– Le navigateur interprète (c'est à dire traduit) le document reçu sous une forme simple pour nous autres humains.
*S'il s'agit d'une image, il l'affiche ;
*S'il s'agit d'un texte, il l'affiche également ;
*S'il s'agit d'une page HTML (ou page Web), il recherche les URLs désignant les documents inclus dans la page (images,sons,frames...) et reprend le processus au point **1**) pour chacune de ces URL.
*S'il s'agit d'un son (MP3/WAV/OGG VORBIS...), il lance le logiciel associé (ce dernier étant installé sur votre système).
– Vous regardez ébahi, le résultat (de même pour cet article).

# PRATIQUE

Fort bien. Après la théorie, passons à la pratique.
Le protocole HTTP est un protocole assez ancien, l'échange se fait donc en mode texte. Il n'est ainsi pas nécessaire de coder des structures particulières pour communiquer avec ce protocole.
En fait un simple client comme *telnet* suffit.

Allez, c'est parti, nous allons simuler à la main le fonctionnement d'un navigateur.

1) On va établir la connection avec le serveur sur le port 80 :

```
lucif3r:~$ telnet clx.anet.fr 80 Trying 80.118.2.10... Connected to gaia.anet.fr. Escape character is '^]'.
```

2) A partir de ce moment, le serveur attend une requête de notre part. Une requête très simple serait

```
GET / \n \n
```

*le caractère '\n' correspond à un retour à la ligne, c'est à dire un appui sur la touche Entrée*

3) A la réception de cette requête, le serveur vous renvoie le contenu du document par défaut. Puis clot la connexion :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN"> [ COUPE CAR TROP LONG ] Connection closed by foreign host.
lucif3r:~$
```

La phase 4) est en fait une mise en page du document HTML qui apparait sous sa forme brute (le source) dans la réponse donnée par telnet.

Easy non ?
Les requêtes peuvent évidement être beaucoup plus complexes.
Voyons de plus près ce qu'il est possible de faire :

# LES REQUETES HTTP

Pour simplifier, nous allons nous intéresser aux trois requêtes les plus courantes.

– GET : Comme nous venons de le voir GET permet de récupérer un document sur le serveur.
– HEAD : Fonctionne comme GET, mais ne retourne qu'un en-tête e HTTP sans le corps du document. Ce type de requête peut être utile pour connaitre la taille du document par exemple.
– POST : Permet d'envoyer des données au serveur dans le corps de la requête. En effet jusqu'à maintenant nous avons vu que les données étaient toujours envoyées par le serveur.

Mais il arrive que le navigateur doive lui aussi faire parvenir des données au serveur. Le cas le plus courant est un formulaire où l'on vous demande vos coordonnées par exemple.

Donc la requête commence en général par une commande GET, HEAD ou POST suivie du chemin du document à chercher.
– / : Permet d'aller chercher le document par défaut qui est en général la page d'accueil du site.
Mais on peut très bien mentionner un chemin complet :

par exemple :

```
lucif3r:~# telnet 10function.kicks-ass.org 80 Trying 80.13.111.104... Connected to 10function.kicks-ass.org.
Escape character is '^]'. GET /books/ruhacker.txt \n \n
```

Permet d'obtenir la prose de ReDragon.(Je rappelle que \n signifie entrée).

En réalité les navigateurs ne se contentent pas de requêtes aussi simples, ils fournissent (parfois à votre insu) d'autres informations à la suite du GET.
Il peut aussi s'agir de négocier certaines options comme la langue ou les tables de caractères supportées.
Il s'agit de ce que nous appelerons une en-tête HTTP.

# L'entête HTTP (Envoyée par le client)

L'entête est généralement une liste de variables et de valeurs associées. Voici un tableau recensant ces variables :

| | |
|---|---|
| Date | Date et heure de génération de la requête |
| Authorization | Permet d'accéder à des ressources protégées par un mot de passe |
| From | Adresse e-mail de l'utilisateur effectuant la requête |
| If-Modified-Since | Spécifie une date permettant d'introduire une condition (GET conditionnel) |
| Referer | Pointe sur l'URL de la page à partir de laquelle le document est demandé. |
| User-Agent | Identifiant logiciel de navigation employé.(w3M RuleZ) |
| Content-Type | Type de données contenues dans le corps de la requête (Dans le cas d'un POST) |
| Content-Length | Longueur du corps éventuelle de la requête |
| Content-Encoding | Indique un codage supplémentaire associé à la ressource accédée. |
| Connection | Comportement désiré au niveau des connections persistantes. |

| | |
|---|---|
| Accept | Indique la liste des types de données supportées par le client |
| Accept-Charset | Enumère la ou les tables de caractères supportées |
| Accept-Encoding | Précise une spécification du type d'encodage accepté pour la réponse |
| Accept-Language | Spécifie la liste des langues préférés de l'utilisateur |
| Host | Indique le nom du serveur(+no de port) à partir duquel le document doit être récupéré.(cf VirtualHost) |

Tout d'abord, il faut savoir que pour utiliser ce type d'enttête, il faut mentionner à la suite de la requête la version du protocole utilisée. (en général HTTP 1.0 ou HTTP/1.0 voire HTTP 1.1 ou HTTP/1.1)

Parmis toutes ces variables celle qui me parait la plus importante est 'Host'.
En effet la plupart des serveurs HTTP - pour ne pas dire tous - n'hébergent pas qu'un seul site, pour des raisons d'économie d'adresses IP. On fait alors appel au *virtual-host* pour les distinguer les uns des autres.
En pratique la valeur fixée avec Host permet d'arriver sur une racine de site différente, et d'associer plusieurs noms de sites à une seule et même adresse IP.
Par exemple :

```
lucif3r:~$ telnet clx.anet.fr 80 Trying 80.118.2.10... Connected to gaia.anet.fr. Escape character is '^]'.
GET / HTTP 1.0\n Host:clx.anet.fr\n  HTTP/1.1 200 OK Date: Sat, 13 Jul 2002 22:07:28 GMT Server: Apache/1.3.9
(Unix) Debian/GNU PHP/3.0.18 mod_ssl/2.4.10 OpenSSL/0.9.4 mod_perl/1.21_03-dev Last-Modified: Sun, 12 May
2002 11:52:42 GMT ETag: "7a81-7f-3cde578a" Accept-Ranges: bytes Content-Length: 127 Connection: close
Content-Type: text/html; charset=iso-8859-1 Connection closed by foreign host.
```

Nous permet de demander la page de garde du clx.
Remarquons dès lors que le serveur a répondu en insérant lui aussi une en-tête HTTP.
Nous allons donc détailler un peu plus l'en-tête HTTP retournée.

# L'en tête HTTP (Renvoyée par le serveur)

| | | | | |
|---|---|---|---|---|
| Accept-Range | Permet au serveur d'indiquer s'il accepte l'accès à des portions de documents et, si oui,avec quelle granularité(au niveau de l'octet par exemple) | | | |
| Content-length | Taille du flux de sortie(en octets), considéré comme étant un binaire | | | |
| Content-ty | Type MIME du flux de sortie | | | |
| Date | Date et heure de génération de la réponse | | Last-Modified | Date de dernière modification |
| Expires | Date et heure de fin de validité du document, lequel doit alors être chargé à nouveau par le serveur. | | | |
| Pragma | Document caché / non caché | | Status | Etat de la requête (ne doit en aucun cas figurer dans un intitulé complet) |

| | | | |
|---|---|---|---|
| Server | Logiciel et version du serveur HTTP | Location | Redirection du serveur (ne doit en aucun cas apparaître dans un intitulé complet.)(Codes de statut 301 et 302) |
| WWW-Authenticate | Demande l'authentification de l'utilisateur(Réponse de type 401) | | |
| Refresh | Le client doit charger à nouveau le document mentionné | | |
| Set-Cookie | Le client stocke les données spécifiées, qui peuvent alors être conservées en mémoire entre deux requêtes | | |

Avec un peu d'attention, vous aurez sans doute remarqué que le serveur débute l'en-tête de la réponse par une chaîne du type :

```
HTTP/1.1 200 OK
```

**HTTP/1.1** est la version du protocole utilisé par le serveur.

**200** est un code de retour dont nous allons détailler tout de suite les valeurs.

**OK** est le message associé à la valeur de retour.

# Code de retour

| | |
|---|---|
| 200 | Réussite |
| 201 | Réussite et création d'un nouveau document (POST) |
| 202 | Requête acceptée, traitement en cours |
| 204 | Aucune Réponse (corps vide) |
| 301 | Document déplacé permanent |
| 302 | Document déplacé temporairement |
| 304 | La ressource demandée n'a pas été modifiée (GET conditionnel) |
| 400 | Erreur de syntaxe dans la requête envoyée |
| 401 | Autorisation non valable |
| 403 | Accès interdit |
| 404 | Introuvable |
| 500 | Erreur système |

| | |
|---|---|
| 501 | Opération non implémentée |
| 502 | Le serveur, utilisé comme intermédiaire, a reçu une réponse invalide du système accédé pour le compte du client |
| 503 | Service indisponible |

Voilà, c'est tout pour aujourd'hui...
Bon test, et à bientôt !

---

[1] Uniform Ressource Locator

[2] En général, le document par défaut se nomme index.html, index.htm ou index.php (sur un serveur apache) et default.htm, default.htm sur un serveur propriétaire Microsoft IIS

[3] *ie.* l'URL

[4] Une session TCP est une connexion avec un serveur distant