Aide mémoire HTTP

(structure des entêtes)

Fabrice HARROUET

É*cole* **N***ationale d'***I***ngénieurs de* **B***rest*

Structure générale d'une requête

Requête HTTP

méthode ressource HTTP/1.*X* *(1.0 ou 1.1)*
option1 : *valeur1*
...
optionN : *valeurN*
(ligne vide → fin de l'entête)
... *données éventuelles à transmettre* ...

Méthodes

- `GET` : réclamer le contenu de la *ressource* (fichier ...)
- `POST` : fournir des données à la *ressource* (*cgi*, *php* ...)
- `CONNECT` : établir une connexion via un *proxy* (*ssl* ...)
- `HEAD` : réclamer juste l'entête de la *ressource* (taille, date ...)
- `PUT`, `DELETE`, `TRACE` : ...

Structure générale d'une réponse

Réponse HTTP

```
HTTP/1.X statut description                          (1.0 ou 1.1)
option1 : valeur1
...
optionN : valeurN
                                  (ligne vide → fin de l'entête)
...  données à transmettre ...                    (contenu utile)
```

Statut/descriptions

- 1*XX* : information
- 2*XX* : succès (`200 OK`, `201 Created` ...)
- 3*XX* : redirection (`301 Moved Permanently` ...)
- 4*XX* : erreur du *client* (`403 Forbidden`, `404 Not Found` ...)
- 5*XX* : erreur du *serveur* (`501 Not Implemented` ...)

Structure générale d'une requête/réponse

Les informations optionnelles

`Host:`	*machine* `:`*port* du serveur à atteindre
`User-Agent:`	identification/version du client émettant la requête
`Server:`	identification/version du serveur délivrant la réponse
`Content-Length:`	taille des données (en décimal) après l'entête
`Content-Type:`	nature des données transmises (type *mime*) (`text/html`, `text/plain`, `image/gif`, `application/pdf` ...)
`Connection:`	gestion de la connexion (`close`, `keep-alive`, ...)
`Expires:`	date d'expiration des données (gestion du cache)
...	

Exemple de requête GET

Le client veut consulter http://www.ietf.org/rfc.html

Il se connecte à www.ietf.org:80 et envoie la requête :

```
GET /rfc.html HTTP/1.1
Host: www.ietf.org:80
Connection: close                          (fermer après la réponse)
                                  (ligne vide → fin de l'entête)
```

Le serveur accepte la connexion, analyse la requête et répond :

```
HTTP/1.0 200 OK
Server: Apache/2.2.4 (Linux/SUSE) ...
...                              (autres options non indiquées ici)
Content-Length: 13671
Content-Type: text/html
Connection: close
                                  (ligne vide → fin de l'entête)
<HTML> ... </HTML>                      (contenu de la ressource)
                                  (le serveur ferme la connexion)
```

Exemple de requête POST

Méthode souvent utilisée pour les formulaires

```
POST /cgi-bin/login.cgi HTTP/1.1
Host: localhost:80
Content-Type: application/x-www-form-urlencoded
Content-Length: 28
Connection: close
```

(ligne vide → fin de l'entête)

```
login=myname&password=mypass
```

(données POST)

Terminaison des lignes de l'entête

En théorie `\n` ou `\r\n`

Il faut s'attendre à recevoir l'une ou l'autre

En pratique certains logiciels se croient les seuls au monde !

Il vaut donc mieux terminer par `\r\n` afin que ces logiciels fonctionnent

Gestion de la connexion en HTTP/1.1

`Connection: close` (par défaut en HTTP/1.0)

La connexion ne sert qu'à une seule paire requête/réponse
Le client lit simplement la réponse jusqu'à la fin de fichier

`Connection: keep-alive` (par défaut en HTTP/1.1)

La connexion sert pour plusieurs paires requête/réponse successives

- Si la réponse contient `Content-Length:` *nb*
  Le client lit exactement *nb* octets
- Si la réponse contient `Transfer-Encoding: chunked`
  Le client lit en boucle :
  - Une ligne de texte contenant une quantité en hexadécimal
  - La quantité d'octets annoncée

  Jusqu'à ce que cette quantité soit nulle

Gestion de la connexion en HTTP/1.1

Requêtes/réponses minimales, fermeture de la connexion

```
 ---------------------------------- connect()
<| GET /this HTTP/1.1               | <-- beginning of request header
<| Host: www.lost-there.org:8080    |
<| Connection: close                |
<|                                  | <-- empty line: end of request header
 ----------------------------------
>| HTTP/1.1 200 OK                  | <-- beginning of reply header
>| Content-Type: text/plain         |
>| Connection: close                |
>|                                  | <-- empty line: end of reply header
>| Here is a 29-byte-long reply!    | <-- reply content
 ---------------------------------- EOF
 ---------------------------------- connect()
<| POST /that HTTP/1.1              | <-- beginning of request header
<| Host: www.lost-there.org:8080    |
<| Connection: close                |
<| Content-Type: text/plain         |
<| Content-Length: 31               |
<|                                  | <-- empty line: end of request header
<| Here is a 31-byte-long request!  | <-- 31 bytes of POST request content
 ----------------------------------
>| HTTP/1.1 200 OK                  | <-- beginning of reply header
>| Content-Type: text/plain         |
>| Connection: close                |
>|                                  | <-- empty line: end of reply header
>| Here is a 29-byte-long reply!    | <-- reply content
 ---------------------------------- EOF
```

Gestion de la connexion en HTTP/1.1

Requêtes/réponses minimales, réutilisation de la connexion

```
 ---------------------------------- connect()
<| GET /this HTTP/1.1               | <-- beginning of request header
<| Host: www.lost-there.org:8080    |
<|                                  | <-- empty line: end of request header
 ----------------------------------
>| HTTP/1.1 200 OK                  | <-- beginning of reply header
>| Content-Type: text/plain         |
>| Connection: keep-alive           |
>| Content-Length: 29               |
>|                                  | <-- empty line: end of reply header
>| Here is a 29-byte-long reply!    | <-- 29 bytes of reply content
 ---------------------------------- (still connected)
<| POST /that HTTP/1.1              | <-- beginning of request header
<| Host: www.lost-there.org:8080    |
<| Content-Type: text/plain         |
<| Content-Length: 31               |
<|                                  | <-- empty line: end of request header
<| Here is a 31-byte-long request!  | <-- 31 bytes of POST request content
 ----------------------------------
>| HTTP/1.1 200 OK                  | <-- beginning of reply header
>| Content-Type: text/plain         |
>| Connection: keep-alive           |
>| Content-Length: 29               |
>|                                  | <-- empty line: end of reply header
>| Here is a 29-byte-long reply!    | <-- 29 bytes of reply content
 ---------------------------------- (still connected)
```

Gestion de la connexion en HTTP/1.1

Requêtes/réponses minimales, réutilisation de la connexion par *chunks*

```
 ---------------------------------- connect()
<| GET /this HTTP/1.1               | <-- beginning of request header
<| Host: www.lost-there.org:8080    |
<|                                  | <-- empty line: end of request header
 ----------------------------------
>| HTTP/1.1 200 OK                  | <-- beginning of reply header
>| Content-Type: text/plain         |
>| Connection: keep-alive           |
>| Transfer-Encoding: chunked       |
>|                                  | <-- empty line: end of reply header
>| 1d                               | <-- hexa. text line: 0x1d = 29
>| Here is a 29-byte-long chunk!    | <-- 29 bytes of reply content
>| 17                               | <-- hexa. text line: 0x17 = 23
>| And a 23-byte-long one!          | <-- 23 more bytes of reply content
>| 0                                | <-- hexa. text line: 0 --> no more chunk
 ---------------------------------- (still connected)
<| POST /that HTTP/1.1              | <-- beginning of request header
<| Host: www.lost-there.org:8080    |
<| Content-Type: text/plain         |
<| Content-Length: 31               |
<|                                  | <-- empty line: end of request header
<| Here is a 31-byte-long request!  | <-- 31 bytes of POST request content
 ----------------------------------
>| HTTP/1.1 200 OK                  | <-- beginning of reply header
>| Content-Type: text/plain         |
>| Connection: keep-alive           |
>| Transfer-Encoding: chunked       |
>|                                  | <-- empty line: end of reply header
>| 1d                               | <-- hexa. text line: 0x1d = 29
>| Here is a 29-byte-long chunk!    | <-- 29 bytes of reply content
>| 17                               | <-- hexa. text line: 0x17 = 23
>| And a 23-byte-long one!          | <-- 23 more bytes of reply content
>| 0                                | <-- hexa. text line: 0 --> no more chunk
 ---------------------------------- (still connected)
```

Utilisation d'un proxy

Client → Proxy

- Le client se connecte toujours au *proxy* (ex: `proxy.enib.fr:3128`)
- La première ligne contient de la requête contient l'*uri* complète (ex : `GET http://www.ietf.org/rfc.html HTTP/1.1` au lieu de `GET /rfc.html HTTP/1.1`)

Proxy → Serveur

- Le *proxy* se connecte au serveur désigné dans la requête
- Il s'adresse au serveur comme le ferait un client (ex : `GET /rfc.html HTTP/1.1`)

Serveur → Proxy → Client

- La structure des réponses est semblable au fonctionnement direct
- Seules quelques options indiquent l'usage du *proxy*

Communication sécurisée en HTTPS

Littéralement : dialogue HTTP dans une connexion SSL

- Connexion entre client et serveur + chiffrement des échanges par SSL
- Le dialogue HTTP est le même que lorsqu'il n'y a pas de chiffrement

Connexion à travers un *proxy*

- Le client demande *"en clair"* la connexion au *proxy*

```
CONNECT imap.enib.fr:443 HTTP/1.1
Host: imap.enib.fr:443
```

(ligne vide → fin de l'entête)

- Le *proxy* se connecte au serveur indiqué et répond *"en clair"* au client

```
HTTP/1.1 200 OK
```

(ligne vide → fin de l'entête)

- Le *proxy* relaie aveuglément les données entre client et serveur
- Le dialogue chiffré est le même que s'il n'y avait pas de *proxy*