Protocole HTTP

Table des matières

L'HyperText Transfer Protocol, plus connu sous l'abréviation HTTP — littéralement « protocole de transfert hypertexte » — est un protocole de communication client-serveur développé pour le World Wide Web. HTTPS (avec S pour secured, soit « sécurisé ») est la variante du HTTP sécurisée par l'usage des protocoles SSL ou TLS. C'est un protocole texte s'appuyant les protocoles plus bas-niveau TCP et IP.

1. Introduction

Les versions du protocole HTTP les plus utilisées actuellement sont les versions 1.0 et 1.1, l'évolution suivante étant le XHTML qui apporte avant tout un peu plus de rigueur dans la syntaxe et un peu plus de cohérence avec le XML, mais le principe reste exactement le même.

Une page est en principe définie par un URI (Unified Ressource Identifier). Par exemple : http://www.debian.org/doc/manuals/reference/ch-preface.fr.html. Nous savons que dans cet URI, il y a trois morceaux :

1. http:

   qui indique quel protocole nous allons employer ;

2. //www.debian.org

   Qui est sensé représenter le serveur web contenant l'information que l'on recherche ;

3. /doc/manuals/reference/ch-preface.fr.html

   qui représente le chemin complet de la page visitée dans l'arborescence du serveur concerné.

Pourtant, si nous indiquons comme URI http://www.debian.org , il manque toute la partie correspondant au nom de la page. Cependant, nous arrivons bien à visualiser quelque chose… Tout simplement, parce que le serveur est paramétré pour ajouter /index.html s'il n'y a pas le nom de la page. Ainsi, http://www.debian.org est équivalent à http://www.debian.org/index.html.

Avec le protocole HTTP la communication entre un navigateur et un serveur Web est assez simple :

1. une URL telle que http://www.debian.org/fichier.html est donnée au navigateur par un internaute
2. le navigateur en extrait le nom de domaine 'www.debian.org' et à partir de cette information sait comment trouver le serveur Web distant (grâce à une opération dite de DNS lookup qui résout un nom de domaine en une adresse IP)
3. à partir de là une connexion (basée sur les protocoles TCP/IP) est établie entre le navigateur et le serveur Web distant
4. une requête HTTP demandant la ressource '/fichier.html' est alors transmise par le navigateur
5. le serveur Web trouve la ressource correspondante et en renvoie le contenu dans une réponse HTTP
6. le navigateur est désormais capable d'afficher le fichier HTML à l'internaute

2. Historique

HTTP a été inventé par Tim Berners-Lee avec les adresses Web et le langage HTML pour créer le World Wide Web. À cette époque, le File Transfer Protocol (FTP) était déjà disponible pour transférer des fichiers, mais il ne supportait pas la notion de format de données telle qu'introduite par Multipurpose Internet Mail Extensions (MIME). La première version de HTTP était très élémentaire, mais prévoyait déjà le support d'en-têtes MIME pour décrire les données transmises. Cette première version reste encore partiellement utilisable de nos jours, connue sous le nom de HTTP/0.9.

- En mai 1996, HTTP/1.0 voit le jour et est décrit dans la RFC 1945. Cette version supporte

les serveurs HTTP virtuels, la gestion de cache et l'identification.

- En janvier 1997, HTTP/1.1 devient finalement standard de l'IETF. Cette version ajoute le support du transfert en pipeline (ou pipelinage) et la négociation de type de contenu (format de données, langue).
- En mars 2012, les travaux à propos de HTTP/2.0 démarrent à l'IETF adoptant SPDY comme matériel de départ.
- En février 2014, la spécification de HTTP 1.1 a été republiée. Elle a été éclatée en plusieurs RFC et corrigée pour toutes ses imprécisions.

3. La commande GET

Ouvrons une session telnet sur le serveur HTTP d'un réseau privé dont le nom d'hôte est linux.maison.mrs (un serveur web écoute par convention sur le port 80).

```
telnet linux.maison.mrs 80

Trying 192.168.0.253...
Connected to linux.maison.mrs (192.168.0.253)
Escape character is '^]'.
```

La session est ouverte. Utilisons maintenant la commande GET…

```
GET / HTTP/1.0
```

Voici la réponse du serveur. Cette partie constitue l'en-tête, systématiquement envoyée par le serveur :

```
HTTP/1.1 200 OK
Date: Wed, 08 May 2002 14:26:57 GMT
Server: Apache-AdvancedExtranetServer/1.3.23
(Mandrake Linux/4mdk) auth_ldap/1.6.0 mod_ssl/2.8.7 OpenSSL/0.9.6c PHP/4.1.2
Last-Modified: Sun, 14 Apr 2002 09:29:32 GMT
ETag: "57d44-116-3cb94bfc"
Accept-Ranges: bytes
Content-Length: 278
Connection: close
Content-Type: text/html
```

Le serveur connaît le protocole HTTP 1.1. C'est un Apache version 1.3.23. Il annonce également ce qu'il sait faire :

- Authentification par annuaire LDAP
- Secure Socket Layer en OpenSSL version 0.9.6c
- Interprétation de code PHP version 4.1.2

Il indique également la date GMT de dernière modification du document demandé, qu'il mettra fin à la connexion TCP à la fin de l'envoi, et que le document fourni est au format MIME : text/html

Et voici le document proprement dit :

```
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv= "Content-Type" content= "text/html; charset=iso-8859-1">
</head>
```

```
<body bgcolor= »#FFFFFF » text= »#000000 »>
<p>Hello world.</p>
<p><img src="images/tux.gif" width= "97" height= "115"></p>
</body>
</html>
```

Le message Connection closed by foreign host., issu de Telnet, indique que, par défaut, la connexion est interrompue par le serveur à la fin de l'envoi du document. L'option Connection: Keep-Alive aurait évité cette déconnexion (nous parlons de la connexion TCP et en aucune façon d'une « session » au niveau de l'application).

Le document HTML indique le nom d'une image, essayons de l'avoir…

```
telnet linux.maison.mrs 80

Trying 192.168.0.253...
Connected to linux.maison.mrs (192.168.0.253).
Escape character is '^]'.

GET /images/tux.gif HTTP/1.0
Trying 192.168.0.253...

Connected to linux.maison.mrs (192.168.0.253).
Escape character is '^]'.

HTTP/1.1 200 OK
Date: Wed, 08 May 2002 14:44:17 GMT
Server: Apache-AdvancedExtranetServer/1.3.23
(Mandrake Linux/4mdk) auth_ldap/1.6.0 mod_ssl/2.8.7 OpenSSL/0.9.6c PHP/4.1.2
Last-Modified: Sun, 14 Apr 2002 09:29:32 GMT
ETag: « cf98a-a20-3cb94bfc »
Accept-Ranges: bytes
Content-Length: 2592
Connection: close

Content-Type: image/gif
GIF89aasæ????/IHG£HIíÈ?Å?2□?°??gp©:<f?×¥?ÄHL§v?■èü?©?°□»¹ÅØ ?ÉÈ×rI?
('}y|□?Ë8:µ}?ÿ}ÁX_À□?□hnÎ??N4?è×?XXZò»?W????98;ýýûhgj°x□□?ôÖ?«Y^ZXbE?
©¶ÙÙëà³L□hÓ?NE?4ÜÌÃXYªyÜ<?Ç?ã©??□..
...
Connection closed by foreign host.
```

Avec une console en mode texte, on a bien reçu l'image. Le type MIME (image/gif) est bien signalé, mais il n'est pas possible de la visualiser avec telnet.

4. Comment travaille le navigateur

Le navigateur fait exactement la même chose que telnet :

- Il appelle la page d'accueil GET / HTTP/1.0 (éventuellement HTTP/1.1, mais alors, suivant la définition de cette version HTTP, il devra au moins envoyer aussi le nom d'hôte du serveur interrogé).
- Une fois la page reçue, il va chercher dedans tous les URI, ici celui de l'image, et va les appeler avec un GET. Dans le cas de l'image, il devra la recomposer, ce qui lui est possible parce qu'il connaît le format d'encodage gif.
- Il affiche alors la page, dans son intégralité.

Nous appelons la page avec Internet Explorer avec un sniffeur qui va enregistrer ce qu'il se passe. Les trames surlignées sont celles qui sont propres à HTTP. Mais n'oublions pas que HTTP s'appuie sur TCP, raison pour laquelle les autres trames existent :

```
No.  Time     Source        Destination   Proto Info
  1 0.000000 192.168.0.10  192.168.0.253 TCP   1282 > 80 [SYN]
  2 0.000163 192.168.0.253 192.168.0.10  TCP   80 > 1282 [SYN, ACK]
  3 0.000565 192.168.0.10  192.168.0.253 TCP   1282 > 80 [ACK]
  4 0.001410 192.168.0.10  192.168.0.253 HTTP  GET / HTTP/1.1
  5 0.001487 192.168.0.253 192.168.0.10  TCP   80 > 1282 [ACK]
  6 0.068550 192.168.0.253 192.168.0.10  HTTP  HTTP/1.1 200 OK
  7 0.098435 192.168.0.10  192.168.0.253 HTTP  GET /images/tux.gif HTTP/1.1
  8 0.098593 192.168.0.253 192.168.0.10  TCP   80 > 1282 [ACK]
  9 0.099450 192.168.0.253 192.168.0.10  HTTP  HTTP/1.1 200 OK
 10 0.099724 192.168.0.253 192.168.0.10  HTTP  Continuation
 11 0.102794 192.168.0.10  192.168.0.253 TCP   1282 > 80 [ACK]
 12 0.102915 192.168.0.253 192.168.0.10  HTTP  Continuation
 13 0.280331 192.168.0.10  192.168.0.253 TCP   1282 > 80 [ACK]
```

- La trame 4 représente la première requête du client
- La trame 6 renvoie le document demandé, c'est à dire la page d'accueil du site.
- La trame 7 indique une requête supplémentaire pour l'image
- Les trames 9, 10 et 12 représentent l'envoi par le serveur de l'image demandée. Les données sont trop volumineuses pour entrer dans une seule trame. Ici, il en faut trois.

5. La première requête HTTP

Pour cette première analyse, on laisse volontairement la totalité de la trame, afin de bien montrer que HTTP est un protocole « application », qui s'appuie sur TCP/IP, lui-même s'appuyant sur Ethernet dans cet exemple (avec une connexion PPPoE, on aurait une couche supplémentaire introduite par PPP).

```
Frame 4 (380 on wire, 380 captured)
    Arrival Time: Apr 13, 2002 16:07:10.266248000
    Time delta from previous packet: 0.000861000 seconds
    Time relative to first packet: 0.001408000 seconds
    Frame Number: 4
    Packet Length: 380 bytes
    Capture Length: 380 bytes

Ethernet II
    Destination: 00:00:b4:bb:5d:ee (00:00:b4:bb:5d:ee)
    Source: 00:20:18:b9:49:37 (00:20:18:b9:49:37)
    Type: IP (0x0800)

Internet Protocol, Src Addr: 192.168.0.10, Dst Addr: 192.168.0.253
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
```

```
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 366
    Identification: 0xc99c
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (0x06)
    Header checksum: 0xad95 (correct)
    Source: 192.168.0.10 (192.168.0.10)
    Destination: 192.168.0.253 (192.168.0.253)

Transmission Control Protocol, Src Port: 2752, Dst Port: 80
    Source port: 2752 (2752)
    Destination port: 80 (80)
    Sequence number: 1135843004
    Next sequence number: 1135843330
    Acknowledgement number: 2485285689
    Header length: 20 bytes
    Flags: 0x0018 (PSH, ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 1... = Push: Set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 17520
    Checksum: 0x6ec7 (correct)

Hypertext Transfer Protocol
    GET / HTTP/1.1\r\n        La version du protocole HTTP utilisé.
                              Suivent les informations envoyées par le client
    Accept: image/gif,                    Les images gif...
            image/x-xbitmap,              Les images bitmap (bmp par exemple)
            image/jpeg,                   Les images jpeg...
            image/pjpeg,
            application/vnd.ms-powerpoint,     Les trois lignes qui suivent
            application/vnd.ms-excel,     Représentent des informations dont
l'intérêt
            application/msword,           peut paraître contestable...
            */*\r\n
    Accept-Language: fr\r\n               Nous parlons français...
    Accept-Encoding: gzip, deflate\r\n
    User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)\r\n
    Host: linux.maison.mrs\r\n      Celle-ci est indispensable au protocole v1.1
    Connection: Keep-Alive\r\n      Notez qu'IE demande à garder la connexion
    \r\n
```

Si IE se permet d'annoncer au monde entier que les composants principaux de MS Office sont présents sur ma machine, ce n'est pas dans le but de « moucharder ». Si vous désirez télécharger des documents .doc, .xls ou .ppt, ces documents devront être transférés par le serveur au format MIME, comme une image gif ou jpg. Internet Explorer informe qu'il accepte ce type de documents. Dans la pratique, tous les navigateurs les acceptent, mais vous proposeront seulement de les

enregistrer en tant que fichiers. Ici, IE indique qu'il est capable de les afficher lui-même.

6. La page d'accueil

```
Frame 6 (710 on wire, 710 captured)
...
Hypertext Transfer Protocol
    HTTP/1.1 200 OK\r\n
    Date: Sun, 14 Apr 2002 09:30:12 GMT\r\n
    Server: Apache-AdvancedExtranetServer/1.3.23 (Mandrake Linux/4mdk)
            auth_ldap/1.6.0
            mod_ssl/2.8.7
            OpenSSL/0.9.6c
            PHP/4.1.2\r\n
    Last-Modified: Sun, 14 Apr 2002 09:29:32 GMT\r\n
    ETag: « 57d44-116-3cb94bfc »\r\n
    Accept-Ranges: bytes\r\n
    Content-Length: 278\r\n
    Keep-Alive: timeout=15, max=100\r\n
    Connection: Keep-Alive\r\n
**    Content-Type: text/html\r\n
**    \r\n

    Data (278 bytes)
0000  3c 68 74 6d 6c 3e 0d 0a 3c 68 65 61 64 3e 0d 0a   <html>..<head>..
0010  3c 74 69 74 6c 65 3e 55 6e 74 69 74 6c 65 64 20   <title>Untitled
0020  44 6f 63 75 6d 65 6e 74 3c 2f 74 69 74 6c 65 3e   Document</title>
0030  0d 0a 3c 6d 65 74 61 20 68 74 74 70 2d 65 71 75   ..<meta http-equ
0040  69 76 3d 22 43 6f 6e 74 65 6e 74 2d 54 79 70 65   iv="Content-Type
0050  22 20 63 6f 6e 74 65 6e 74 3d 22 74 65 78 74 2f   "content="text/
0060  68 74 6d 6c 3b 20 63 68 61 72 73 65 74 3d 69 73   html; charset=is
0070  6f 2d 38 38 35 39 2d 31 22 3e 0d 0a 3c 2f 68 65   o-8859-1">..</he
0080  61 64 3e 0d 0a 0d 0a 3c 62 6f 64 79 20 62 67 63   ad>....<body bgc
0090  6f 6c 6f 72 3d 22 23 46 46 46 46 46 46 22 20 74   olor="#FFFFFF" t
00a0  65 78 74 3d 22 23 30 30 30 30 30 30 22 3e 0d 0a   ext="#000000">..
00b0  3c 70 3e 48 65 6c 6c 6f 20 77 6f 72 6c 64 2e 0d   <p>Hello world..
00c0  0a 3c 2f 70 3e 0d 0a 3c 70 3e 3c 69 6d 67 20 73   .</p>..<p><img s
00d0  72 63 3d 22 69 6d 61 67 65 73 2f 74 75 78 2e 67   rc="images/tux.g
00e0  69 66 22 20 77 69 64 74 68 3d 22 39 37 22 20 68   if" width="97" h
00f0  65 69 67 68 74 3d 22 31 31 35 22 3e 0d 0a 3c 2f   eight="115">..</
0100  70 3e 0d 0a 3c 2f 62 6f 64 79 3e 0d 0a 3c 2f 68   p>..</body>..</h
0110  74 6d 6c 3e 0d 0a                                 tml>..
```

C'est au navigateur de se débrouiller pour aller chercher les données de cette image, à partir des références fournies. Ceci justifie la présence de la requête de la trame 7 :

```
Frame 7 (298 on wire, 298 captured)
...
Hypertext Transfer Protocol
  GET /images/tux.gif HTTP/1.1\r\n        Appel d'une référence relative
  Accept: */*\r\n
  Referer: http://linux.maison.mrs\r\n    Depuis la page indiquée, ce qui aboutit
                                          à la référence absolue:
                                          http://linux.maison.mrs/images/tux.gif
  Accept-Language: fr\r\n
  Accept-Encoding: gzip, deflate\r\n
```

```
   User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)\r\n
   Host: linux.maison.mrs\r\n
   Connection: Keep-Alive\r\n
   \r\n
```

Le serveur envoie les données à partir de la trame 9 :

```
Frame 9 (1514 on wire, 1514 captured)
....
Hypertext Transfer Protocol
   HTTP/1.1 200 OK\r\n
   Date: Sun, 14 Apr 2002 09:30:12 GMT\r\n
   Server: Apache-AdvancedExtranetServer/1.3.23 (Mandrake Linux/4mdk)
         auth_ldap/1.6.0
         mod_ssl/2.8.7
         OpenSSL/0.9.6c
         PHP/4.1.2\r\n
   Last-Modified: Sun, 14 Apr 2002 09:29:32 GMT\r\n
   ETag: « cf98a-a20-3cb94bfc »\r\n
   Accept-Ranges: bytes\r\n
   Content-Length: 2592\r\n
   Keep-Alive: timeout=15, max=99\r\n
   Connection: Keep-Alive\r\n
   Content-Type: image/gif\r\n
   \r\n
    Data (1082 bytes)

0000  47 49 46 38 39 61 61 00 73 00 e6 00 00 04 04 05   GIF89aa.s.......
0010  a4 85 2f 49 48 47 a3 48 49 ed c8 10 c5 a4 32 9b   ../IHG.HI.....2.
0020  69 07 ba 85 0e be 67 70 a9 3a 3c 85 66 1d d7 a5   i.....gp.:<.f...
0030  15 c4 48 4c a7 76 0c 8b 88 8d e8 e8 fc b8 88 94   ..HL.v..........
...
```

7. Le cache

L'internaute ferme son navigateur. Quelques instants plus tard, il l'ouvre à nouveau et réclame la même page :

```
No.  Time     Source        Destination   Proto  Info
  1  0.000000 192.168.0.10  192.168.0.253 TCP    2632 > 80 [SYN]
  2  0.000144 192.168.0.253 192.168.0.10  TCP    80 > 2632 [SYN, ACK]
  3  0.000540 192.168.0.10  192.168.0.253 TCP    2632 > 80 [ACK]
  4  0.001342 192.168.0.10  192.168.0.253 HTTP   GET / HTTP/1.1
  5  0.001461 192.168.0.253 192.168.0.10  TCP    80 > 2632 [ACK]
  6  0.004186 192.168.0.253 192.168.0.10  HTTP   HTTP/1.1 304 Not Modified
  7  0.200392 192.168.0.10  192.168.0.253 TCP    2632 > 80 [ACK]
```

La réponse n'est pas la même que dans le cas précédent.

7.1. La requête

```
Frame 4 (336 on wire, 336 captured)
...
```

```
Hypertext Transfer Protocol
    GET / HTTP/1.1\r\n
    Accept: */*\r\n
    Accept-Language: fr\r\n
    Accept-Encoding: gzip, deflate\r\n
    If-Modified-Since: Sat, 13 Apr 2002 13:40:06 GMT\r\n
                  Cette ligne n'apparaissait pas dans la requête précédente...

    If-None-Match: "57d44-d2-3cb83536"\r\n
    User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)\r\n
    Host: linux.maison.mrs\r\n
    Connection: Keep-Alive\r\n
    \r\n
```

Internet Exlorer demande au serveur la page, si elle a été modifiée depuis la date de son précédent chargement, tout simplement parce qu'il a conservé en cache cette page qui a été demandée il n'y a pas si longtemps.

7.2. La réponse

```
Frame 6 (327 on wire, 327 captured)
...
Hypertext Transfer Protocol
    HTTP/1.1 304 Not Modified\r\n
    Date: Sat, 13 Apr 2002 13:53:52 GMT\r\n
    Server: Apache-AdvancedExtranetServer/1.3.23 (Mandrake Linux/4mdk)
            auth_ldap/1.6.0 mod_ssl/2.8.7 OpenSSL/0.9.6c PHP/4.1.2\r\n
    Connection: Keep-Alive\r\n
    Keep-Alive: timeout=15, max=100\r\n
    ETag:  "57d44-d2-3cb83536"\r\n
    \r\n
```

Le serveur s'est contenté de répondre que la page n'a pas été modifiée…

Le navigateur va donc réafficher la page qu'il a conservée en cache. Cette méthode de travail présente deux particularités :

- Le temps d'affichage est considérablement raccourci lorsque l'on navigue dans un site, puisque les pages déjà chargées ne le sont généralement plus si l'on revient dessus.
- L'espace requis pour le cache gonfle considérablement et peut occuper jusqu'à plusieurs dizaines de Mo sur le disque…

Remarque : il existe de nombreux cas où le contrôle sur la modification n'est pas efficace. Le navigateur affiche alors une version obsolète de la page.

7.3. Liste des codes HTTP

Voici une description des principales valeurs à connaître :

Code HTTP	**Explication**	
Code HTTP 200	Ok	Tout va bien, le serveur Web renvoie le contenu d'une ressource dans le corps (body) de la réponse

Code HTTP 301	Moved Permanently	C'est une redirection permanente vers une URL indiquée dans la réponse (champs 'Location'). Cette redirection est considérée comme 'Google Friendly' et n'affecte pas votre référencement naturel.
Code HTTP 302	Moved temporarily	C'est une redirection temporaire vers une URL indiquée. C'est moins apprécié par Google
Code HTTP 400	Bad request	La requête reçue par le serveur Web ne respecte pas le format défini par le protocole HTTP
Code HTTP 401	Unauthorized	L'accès à l'URL est sécurisée. Le serveur Web demande ainsi de lui indiquer un login/mot de passe
Code HTTP 403	Forbidden	Un login/mot de passe invalide a été donnée pour accéder à une URL sécurisée
Code HTTP 404	File not found	Le serveur Web n'a pas trouvé de ressource correspondant à l'URL indiquée
Code HTTP 500	Internal Error	Le serveur Web n'a pas été capable de traiter la requête HTTP. Cela peut indiquer un problème très sérieux.

7.4. Liste des champs HTTP

Pour les requêtes :

Nom du champ	**Explication**
Host	c'est le nom du domaine visé. Un serveur Web peut gérer plusieurs domaines en même temps sur la même machine. On dit alors qu'il gère des 'virtual hosts'.
User-Agent	ce champ est utilisé par les navigateurs pour indiquer leur nom. Les crawlers des moteurs de recherches utilise aussi ce champ pour se faire reconnaître (ex: Googlebot pour Google)
Referer	quand un utilisateur clique sur un lien externe à partir d'une page, le navigateur indique dans ce champ l'URL de la source. Cela peut être très utile pour analyser la source de notre trafic.

Pour les réponses :

Content-Type	le serveur Web indique le type ('MIME type') de la ressource qu'il renvoie (image, html, pdf...). Il peut même indiquer parfois l'encodage des caractères du contenu. Par exemple: text/html; charset=UTF-8
Server	le serveur Web indique sa signature dans ce champ.
Set-Cookie	le serveur Web (souvent guidé par les lignes de code du développeur d'un site Web) peut indiquer au navigateur des valeurs de cookies. Ces valeurs seront renvoyés par le navigateur dans les requêtes futures.
Location	c'est le champ utilisé lors des redirections pour indiquer la nouvelle URL cible.

8. Les commandes HTTP

Dans le protocole HTTP, une méthode est une Commande spécifiant un type de requête, c'est-à-dire qu'elle demande au serveur d'effectuer une action. En général l'action concerne une ressource identifiée par l'URL qui suit le nom de la méthode.

Il existe de nombreuses méthodes, les plus courantes étant GET, HEAD et POST :

Méthode	**Explication**
GET	Cette méthode est la plus courante, il s'agit normalement d'une simple requête de téléchargement d'un document. Deux requêtes GET portant sur le même document devraient retourner des réponses sémantiquement identiques (certaines en-têtes pouvant influer sur le comportement du serveur, les réponses peuvent ne pas être totalement identiques). Aucune donnée à traiter ne peut être envoyée au serveur par cette méthode. Il est par contre possible d'ajouter les paramètres d'URL (aussi nommés paramètres GET). Le corps de la requête DOIT être vide, et le document spécifié dans la requête (la Page) est celui qui doit être retourné.
HEAD	Cette méthode ne demande que des informations sur la ressource, sans demander la ressource elle-même.
POST	Cette méthode est utilisée pour transmettre des données en vue d'un traitement à une ressource (le plus souvent depuis un formulaire HTML). L'URI fournie est l'URI d'une ressource à laquelle s'appliqueront les données envoyées. Le résultat peut être la création de nouvelles ressources ou la modification de ressources existantes. À cause de la mauvaise implémentation des méthodes HTTP (pour Ajax) par certains navigateurs (et la norme HTML qui ne supporte que les méthodes GET et POST pour les formulaires), cette méthode est souvent utilisée en remplacement de la requête PUT, qui devrait être utilisée pour la mise à jour de ressources.
OPTIONS	Cette méthode permet d'obtenir les options de communication d'une ressource ou du serveur en général.
CONNECT	Cette méthode permet d'utiliser un proxy comme un tunnel de communication.
TRACE	Cette méthode demande au serveur de retourner ce qu'il a reçu, dans le but de tester et effectuer un diagnostic sur la connexion.
PUT	Cette méthode permet de remplacer ou d'ajouter une ressource sur le serveur. L'URI fourni est celui de la ressource en question.
PATCH	Cette méthode permet, contrairement à PUT, de faire une modification partielle d'une ressource.
DELETE	Cette méthode permet de supprimer une ressource du serveur.

9. HTTP 1.1

La différence avec HTTP 1.0 est une meilleure gestion du cache. L'en-tête Host devient obligatoire dans les requêtes.

Les soucis majeurs des deux premières versions du protocole HTTP sont d'une part le nombre important de connexions lors du chargement d'une page complexe (contenant beaucoup d'images ou

d'animations) et d'autre part le temps d'ouverture d'une connexion entre client et serveur (l'établissement d'une connexion TCP prend un temps triple de la latence entre client et serveur). Des expérimentations de connexions persistantes ont cependant été effectuées avec HTTP 1.0 (notamment par l'emploi de l'en-tête Connection: Keep-Alive), mais cela n'a été définitivement mis au point qu'avec HTTP 1.1.

Par défaut, HTTP 1.1 utilise des connexions persistantes, autrement dit la connexion n'est pas immédiatement fermée après une requête, mais reste disponible pour une nouvelle requête. On appelle souvent cette fonctionnalité keep-alive. Il est aussi permis à un client HTTP d'envoyer plusieurs requêtes sur la même connexion sans attendre les réponses. On appelle cette fonctionnalité pipelining. La persistance des connexions permet d'accélérer le chargement de pages contenant plusieurs ressources, tout en diminuant la charge du réseau.

La gestion de la persistance d'une connexion est gérée par l'en-tête Connection.

HTTP 1.1 supporte la négociation de contenu. Un client HTTP 1.1 peut accompagner la requête pour une ressource d'en-têtes indiquant quels sont les langues et formats de données préférés. Il s'agit des en-têtes dont le nom commence par Accept-.

Les en-têtes supplémentaires supportés par HTTP 1.1 sont :

Connection	Cet en-tête peut être envoyé par le client ou le serveur et contient une liste de noms spécifiant les options à utiliser avec la connexion actuelle. Si une option possède des paramètres ceux-ci sont spécifiés par l'en-tête portant le même nom que l'option (Keep-Alive par exemple, pour spécifier le nombre maximum de requêtes par connexion). Le nom close est réservé pour spécifier que la connexion doit être fermée après traitement de la requête en cours.
Accept	Cet en-tête liste les types MIME de contenu acceptés par le client. Le caractère étoile * peut servir à spécifier tous les types / sous-types.
Accept-Charset	Spécifie les encodages de caractères acceptés.
Accept-Language	Spécifie les langues acceptées.

10. HTTPS

HTTPS (Hypertext Transfer Protocol Secure ou protocole de transfert hypertexte sécurisé) est un protocole de communication Internet qui protège l'intégrité et la confidentialité des données de vos visiteurs lors du transfert d'informations entre l'ordinateur de l'internaute et le site. Par exemple, lorsqu'un internaute saisit des informations dans un formulaire en ligne afin de recevoir des notifications ou d'acheter un produit, le protocole HTTPS protège les informations personnelles de cet internaute lors du transfert d'informations entre l'internaute et le site.

Les données envoyées à l'aide du protocole HTTPS sont sécurisées via le protocole Transport Layer Security (TLS), qui offre trois niveaux clés de protection :

1. Le chiffrement : consiste à coder les données échangées pour les protéger des interceptions illicites. Cela signifie que lorsqu'un internaute navigue sur un site Web, personne ne peut "écouter" ses conversations, suivre ses activités sur diverses pages ni voler ses informations.
2. L'intégrité des données : les informations ne peuvent être ni modifiées, ni corrompues durant leur transfert, que ce soit délibérément ou autrement, sans être détectées.

3. L'authentification : prouve que les internautes communiquent avec le bon site Web. L'authentification protège contre les attaques des intercepteurs et instaure un climat de confiance pour l'internaute qui se traduit par d'autres retombées pour votre activité.