Diffusion d'information multimédia
Chapitre 2 - HTTP
Master 2 SIMI

Sébastien Choplin
IUP MIAGe Amiens

2005

Introduction

HTTP (Hyper Text Transfert Protocol), procotole utilisé pour le web.

Défini en 1992. RFC 2616

http ://www.w3.org/Protocols/rfc2616/rfc2616.html

Echange ASCII au dessus de TCP sur le port 80.

Contenu des spécifications :

- les requêtes
- les méthodes (put,get,...)
- les entêtes des requêtes
- les réponses
- les codes d'état (404,...)
- la négociation de formats ...

Liens intéressants

http ://www.lmcp.jussieu.fr/enseignement/ye/informatique/app_reseaux/http/index.htm

http ://www-sop.inria.fr/mascotte/Sebastien.Choplin/cours/minfo-2002/

Localisation

URL (Uniform Resource Locator)
L'URL est une méthode qui permet de repérer de manière unique un objet sur un serveur du réseau.
L'URL contient :
- le type de connexion appelé aussi *méthode*
- le nom du serveur
- le chemin du l'objet
- le nom de l'objet

Le format standard est :

```
<methode>:<localisation de l'objet>
```

Localisation

Le format standard est :

`<methode>:<localisation de l'objet>`

http ://www.u-picardie.fr/miage/index.html

- méthode : *http*,
- serveur : *www.u-picardie.fr*,
- chemin : */miage*,
- objet : *index.html* .

Méthodes de localisation

Méthode les plus courantes :

- http - Hypertext Transfer Protocol
- ftp - File Transfert Protocol
- mailto - Adresse électronique
- file - Fichier local
- telnet
- wais
- news
- gopher
- ...

HTTP (*Hypertext Transfer Protocol*)

Le client Web et le serveur Web échangent les informations en utilisant le protocole HTTP.

Grossièrement :

- Le client demande une connexion avec le serveur à l'aide d'une requête en transférant l'URL du document demandé.
- Le serveur accepte la connexion et fournit au client le document correspondant à l'URL demandé puis coupe la connexion.

HTTP (*Hypertext Transfer Protocol*)

versions >=1.0 : types MIME

extension sécurisée du protocole HTTP : S-HTTP (*Secure HyperText Transfer Protocol*).

Version 1.1 : la connexion peut rester ouverte.

MIME (*Multipurpose Internet Mail Extensions*)

MIME is a freely available specification that offers a way to **interchange text** in languages with **different character sets**, and **multi-media email** among many different computer systems that use **Internet mail standards**.

If you were bored with plain text email messages, thanks to MIME you now can create and read email messages containing these things :

- character sets other than ASCII
- enriched text
- images
- sounds

- other messages (reliably encapsulated)
- tar files
- PostScript
- FTPable file pointers
- other stuff

Pour en savoir plus sur les MIME, se référer à :

ftp ://ftp.enst.fr/faq/mime.faq

Types MIME - exemples

Dans l'entête HTTP, chaque partie pourra mettre quels types MIME ils acceptent :

```
Accept: image/gif, image/x-xbitmap, image/jpeg, */*
```

Type	Exemples de sous-types
application	pdf, msword, ps, zip...
audio	mpeg...
image	gif, jpeg, tiff, g3-fax...
message	http, news...
model	vrml (représentation 3D interactive)...
multipart	encrypted, digest....
text	rtf, enriched, html...
video	jpeg, mpeg...

Requêtes HTTP

Les requêtes d'un client et d'un serveur contiennent trois parties principales :

- Une ligne indiquant s'il s'agit d'une requête ou d'une réponse,
- une section d'entête,
- le contenu de la requête ou de la réponse.

HTTP - Requête du client

- Commande HTTP

  ```
  GET /index.html HTTP/1.0
  ```

  effectue une requête avec la méthode *GET* sur le document *index.html* avec la version *1.0* de HTTP.
- Entêtes d'information

  ```
  <nom>: <valeur>
  User-Agent: Mozilla/2.02Gold (WinNT; I)
  Accept: image/gif, image/jpeg, */*
  ```
- Après les entêtes, le client peut envoyer des données (POST)

HTTP - Réponse du serveur

Le serveur répond en commançant par une ligne contenant 3 champs : la version de HTTP utilisée, un code de retour et une description.

```
HTTP/1.0 200 OK
```

Le code de retour *200* indique que la requête a été acceptée et que la réponse se trouve après les entêtes.

HTTP - Réponse du serveur

Informations sur sa configuration et celle du document requis.

```
Date: Fri, 20 Sep 1996 08:17:58 GMT
Server: NCSA/1.5.2
Last-modified: Mon, 17 Jun 1996 21:53:08 GMT
Content-type: text/html
Content-length: 2482
```

Si la requête est acceptée, les données requises sont ensuite envoyée.

HTTP - Méthodes

Les trois principales sont *GET*, *POST* et *HEAD*.

HTTP - Méthode GET

La méthode GET est une requête d'information. Le serveur traite la demande et renvoie le contenu de l'objet.

Exemple le plus courant : fichier HTML.

```
GET /index.html HTTP/1.1
User-Agent: Mozilla/2.02 (WinNT; I)
Host: www.u-picardie.fr
Accept: image/gif, image/jpeg, */*
```

ATTENTION : *Host : www.u-picardie.fr* obligatoire pour HTTP/1.1

Méthodes : GET

Réponse du serveur :

```
HTTP/1.1 200 Document follows
Date: Fri, 05 Nov 2004 09:28:23 GMT
Server: Apache/2.0.50 (Fedora)
Last-Modified: Mon, 18 Oct 2004 07:40:09 GMT
Content-type: text/html
Content-length: 2674

==Donnees==
```

Remarques : date de modification, type

Méthode GET

La méthode *GET* permet aussi de passer des paramètres à l'URL.

```
GET /cgi-bin/anniversaire.pl?mois=aout&date=24 HTTP/1.0
```

Le client réclame au serveur de déclencher l'application *anniversaire.pl* au travers de l'interface CGI (Computer Gateway Interface) du serveur en lui transmettant en données *mois* et *date*.

Le client emploie la notation hexadécimale pour envoyer des caractères spéciaux :
blanc $\Leftrightarrow$ %20

Méthode HEAD

Comme *GET* sauf qu'elle demande au serveur de ne renvoyer **que les entêtes de la réponse** et pas les données requises.

But : récupérer uniquement des informations sur le document et non le document (date de dernière modification, taille, type, . . .).

Méthode POST

Permet d'envoyer des données au serveur dans la requête.

```
POST /cgi-bin/anniversaire.pl HTTP/1.0
User-Agent: Mozilla/2.02 (WinNT; I)
Host: www.u-picardie.fr
Accept: image/gif, image/jpeg, */*
Content-type: application/x-www-form-urlencoded
Content-length: 17

mois=aout;date=24
```

Méthode POST - multipart

envoyer plusieurs données (ex : transmettre un fichier en upload)

syntaxe du formulaire HTML :

```
<form enctype=''mutipart/form-data'' ...>
```

Méthode POST - multipart - exemple

```
POST /script.php HTTP/1.1
Accept: */*
Content-Type: multipart/form-data; boundary=----------0xKhTmLbOuNdArY
Content-Length: 280
Host: serveur

------------0xKhTmLbOuNdArY
Content-Disposition: form-data; name="NOM"

choplin
------------0xKhTmLbOuNdArY
Content-Disposition: form-data; name="PRENOM"

seb
------------0xKhTmLbOuNdArY
Content-Disposition: form-data; name="GROUPE"

32
------------0xKhTmLbOuNdArY--
```

Méthode OPTIONS

Permet de demander au serveur les méthodes autorisées pour le document référencé. Par exemple, à la requête :

```
OPTIONS / HTTP/1.0
```

Le serveur peut répondre :

```
Allow: GET, HEAD
```

Méthode PUT

La méthode PUT a été définie avec la version 1.1.

Permet de télécharger un document sur le serveur.

Méthode DELETE

Cette méthode complète la précédente et permet d'effacer un document, toujours si le serveur l'autorise.

Méthode TRACE

TRACE est une méthode employée pour le débogage.

Le serveur renvoie, dans le corps de la réponse, le contenu exact qu'il a reçu du client. Ceci permet de comprendre, en particulier, ce qui se passe lorsque la requête transite par plusieurs serveurs intermédiaires.

```
TRACE / HTTP/1.1
Host: www.u-picardie.fr
```

Le serveur renvoie sa réponse associée à un code parmi ceux que nous avons décrits dans la section précédente.

Autres méthodes

- LINK
- UNLINK
- CONNECT
- PATCH
- PROPFIND
- PROPPATCH
- MKCOL
- COPY
- MOVE
- LOCK
- UNLOCK
- ...

Entêtes

En-têtes généraux

- **Cache-Control** : instruction.
  Requête relative à la façon d'employer les caches presents sur toute la chaîne de communication entre le client et le serveur.
- **Connection** : option
  L'option close, en particulier, permet de terminer une connexion persistante.
- **Date** : date
  Fournit la date courante
- **Mime-Version** : version
  Version MIME employée

- **Transfer-Encoding** : option
  Indique le codage qui a été appliqué au message (pas à l'objet entier) pour son transfert. La valeur chunked permet de découper un message en fragments que le client devra reconstituer à l'arrivée.
- **Upgrade** : protocole/version
  Permet au client d'indiquer au serveur les protocoles préférés. Le serveur, s'il le peut, les emploiera et le signifiera avec le code "201 Switching Protocols"

Entêtes - Requêtes client

Fournies par le client, traitées par le serveur.

- **Accept** : type/sous-type
- **Accept-Charset** : jeu de caractères
  Indique les jeux de caractères acceptés (ISO 8859-1) Il est possible d'indiquer un ordre par une valeur de qualité.
- **Accept-Encoding** : types d'encodage
  Spécifie les types d'encodage acceptés par le client comme gzip
- **Accept-Language** : langue
  Indique les langues reconnues par le client et ses préférences : fr, en. Si employé les autres langages

ne seront pas acceptés !

- **Authorization** : schéma
  Fournit l'identification du client nécessaire pour s'authentifier auprès du serveur afin d'accéder à une ressource. (htaccess)
- **Cookie** : nom=valeur
  Contient un doublet nom, valeur spécifique à l'URL qui est envoyé au serveur lorsque celle-ci est demandée. Nous reviendrons sur leur usage dans la section suivante.
- **From** : adresse e-mail
  Adresse de l'utilisateur employant le client. Celle-ci n'a de sens que si le client a été correctement configuré. Il est donc facile de se présenter comme étant une autre personne et cette information doit

être traitée avec prudence par le serveur car elle n'est pas authentifiée. Elle peut servir notamment à rediriger un client vers un serveur national. Les normes indiquent que cet en-tête est optionnel et doit pouvoir être supprimé à la demande de l'usager.

- **Host** : nom_hote :port
  Spécifie l'adresse de l'URI demandée ainsi que, optionnellement, le port d'accès. Sinon le port par défaut sera employé. Cet en-tête est évidemment obligatoire ! (serveurs virtuels)
- **If-Modified-Since** : date
  Employé pour limiter l'envoi d'un URI, par le serveur s'il n'existe pas déjà dans le cache du client une version plus récente. Si le document n'a pas été

modifié le serveur renvoie le code 304 et le client emploie sa copie locale.

- **Referer** : url
  Le client emploie cet en-tête pour indiquer au serveur l'adresse du serveur qui lui a fourni l'URI demandé.
- **User-Agent** : texte
  Donne l'identifiant du programme client. Utile pour les statistiques d'un serveur comme pour un traitement particulier des données à renvoyer en fonction des capacités du client.

Entêtes - Réponses

- **Retry-After** : date ou secondes
  Au cas où le service serait indisponible le serveur peut indiquer, en conjonction avec le code 503 un délai d'attente pour une prochaine tentative.
- **Server** : nom
  Contient la référence logicielle du serveur, par exemple
  ```
  Server: Apache/2.0.50 ( Fedora)
  ```
- **Set-Cookie** : nom=valeur ; options
  Envoie au client un couple nom, valeur qu'il pourra retenir pour usage ultérieur avec cet URI. Les options précisent entre autres une date de fin de validité et

le domaine pour lequel le cookie est valable. Nous y reviendrons dans la section suivante.

- **Allow** : méthodes
  Donne la liste des méthodes d'accès autorisées pour l'URI.
- **Content-Encoding** : schéma MIME
  Fournit les schémas d'encodage MIME contenus dans le corps, dans l'ordre dans lequel ils sont employés.
- **Content-Language** : codes
  Indique le langage auquel le corps correspond. Utile pour employer les alphabets spécifiques.
- **Content-Length** : n
  Donne la longueur du contenu en octets. Cet en-tête pose un problème lorsque la réponse

fournie par le serveur est obtenue de façon dynamique : le résultat de la requête est souvent renvoyé avant que le contenu ne soit entièrement calculé. La longueur est donc inconnue au moment où débute l'émission. Dans ce cas cet en-tête devra être omis mais cela interdit au client tout contrôle.

- **Content-Transfer-Encoding** : schéma
  Indique les codages qui ont été employés pour transmettre l'entité sur le réseau : 7 bit, 8bit, binary, base64 ou quoted-printable. Ces codages sont les mêmes que ceux employés pour la messagerie.
- **Content-Type** : type/ soustype
  Type de média MIME (par dÃaut text/plain sauf pour POST application/x-www-form-urlencoded)
- **Expires** : date

Date de fin de validité du document. Cela n'indique pas qu'il sera forcément modifié.

- **Last-Modified** : date
  Indique la date de mise à jour du document.
- **Location** : uri
  Utilisé par le serveur pour indiquer la nouvelle adresse d'un document. Très utile lorsqu'un document a été déplacé. Nous en verrons un exemple lorsque nous décrirons la configuration du serveur Apache.

TOC