Protocole HTTP

Hyper Text Transfert Protocol

dernière modification : 21/02/2023 21:14

Philippe Genoud

Le protocole HTTP

- Protocole de communication
  - un ensemble de règles,
  - dans un contexte réseau ensemble de règles qui régissent les échanges de données entre deux programmes distants

- HTTP (Hyper Text Transfert Protocol)
  - Protocole de communication permettant de récupérer et d'envoyer des ressources hypermédia.
  - créé au départ pour permettre le transfert de documents HTML uniquement
  - ses usages se sont rapidement étendus pour permettre le transfert d'autres types de ressources (images, vidéos, feuilles de styles, scripts, données…)

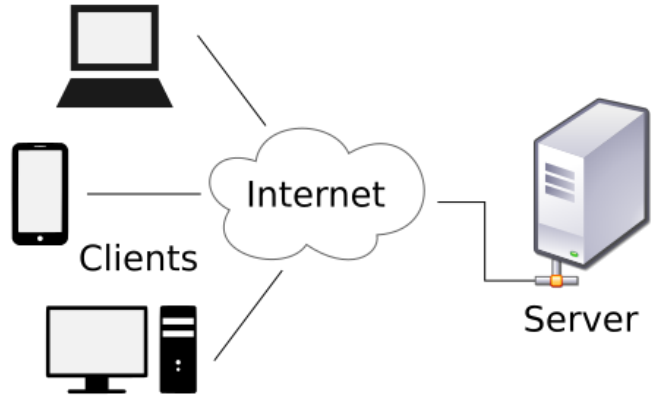
- autres exemples de protocoles
  - SMTP : Simple Mail Transfer Protocol
  - FTP : File Transfert Protocol

Pourquoi étudier le fonctionnement d'HTTP ?

- Comprendre les interactions entre clients web (navigateurs, robots, moteurs de recherche, applications web…) et les serveurs web → meilleure compréhension du web en général
- Interroger manuellement des serveurs web
  - Recevoir informations de bas niveau cachées par navigateurs
  - Mieux comprendre la configuration et les capacités d'un navigateur et d'un serveur
  - Déboguer erreurs de configuration du serveur ou de programmation dans les programmes invoqués par le serveur web.
- Faire un meilleur usage de ce protocole
  - écriture/débogage d'application web dynamiques
    - côté serveur (dans tous les langages de programmation, Java, C#, JavaScript, Python, Ruby, PHP, C++, C ….)
    - côté client (javascript, typescript)

Caractéristiques de HTTP

- HTTP, **un protocole de transfert client – serveur**
  - le client
    - celui qui demande à accéder (demande de récupérer) une ressource
    - toujours à l'origine d'une communication
  - le serveur
    - celui qui va répondre à la demande (qui va « servir » la ressource)
  - exemples
    - clients :
      - logiciels clients : navigateurs web, robots d'indexation, applications web
      - machines clientes : ordinateur portable, ordinateur de bureau, smart phone…
    - serveur webs :
      - logiciel serveur (serveur HTTP) capable de fournir des ressources web : Apache HTTPD, IIS (Internet Information Service – Microsoft), Nginx, Node.js (express)…
      - machine serveur : ordinateur sur lequel tourne le serveur (en général un ordinateur très puissant donc l'unique rôle est de stocker des ressources et de les servir aux clients lorsque ceux-ci les demandent)
        - en production, un serveur doit être toujours allumé et connecté de manière à pouvoir être accessible



Caractéristiques de HTTP

- Message (ou transaction) HTTP :
  - une requête (client) **et** un réponse (serveur) HTTP
  - aux débuts du web la seule opération possible en tant qu'utilisateur était la récupération et la visualisation d'une ressource (document HTML)
  - au fur et à mesure de l'évolution du web diversification des types de ressources (HTML, images, CSS, JavaScript, vidéos, données XML/JSON…) et des usages (le client peut également envoyer ou modifier des informations (formulaires HTML, appels AJAX….)).

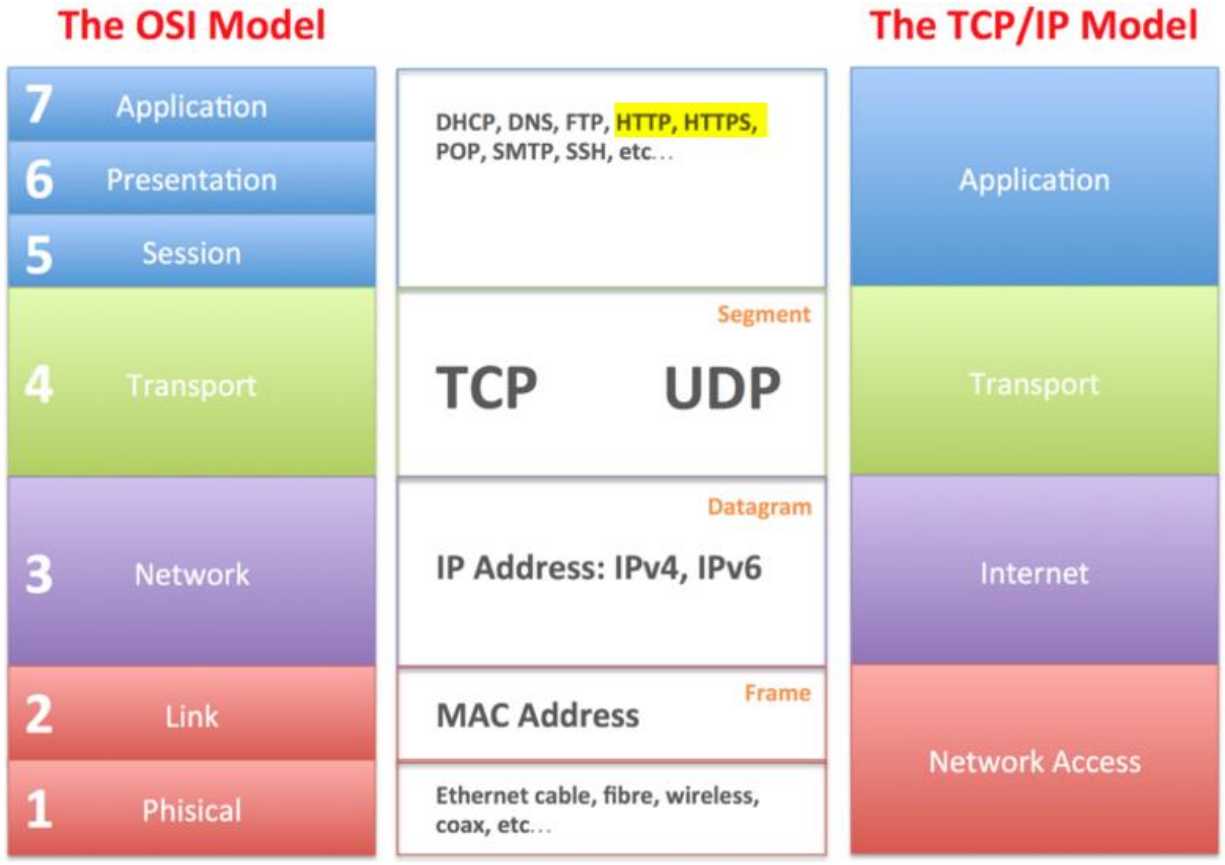
➔nécessité pour HTTP d'évoluer pour fournir d'avantages de possibilités d'échanges et de performances tout en conservant les caractéristiques de départ

1. une requête ne peut récupérer qu'une seule ressource à la fois / n'effectuer qu'une action à la fois
2. HTTP est **un protocole sans état** (**stateless**)
   - chaque nouvelle requête est **complètement indépendante** des précédentes, chaque message peut être vu comme une simple transaction et n'a pas de lien a priori avec les messages précédents
   - le serveur HTTP ne mémorise rien au sujet des messages précédents

Caractéristiques de HTTP

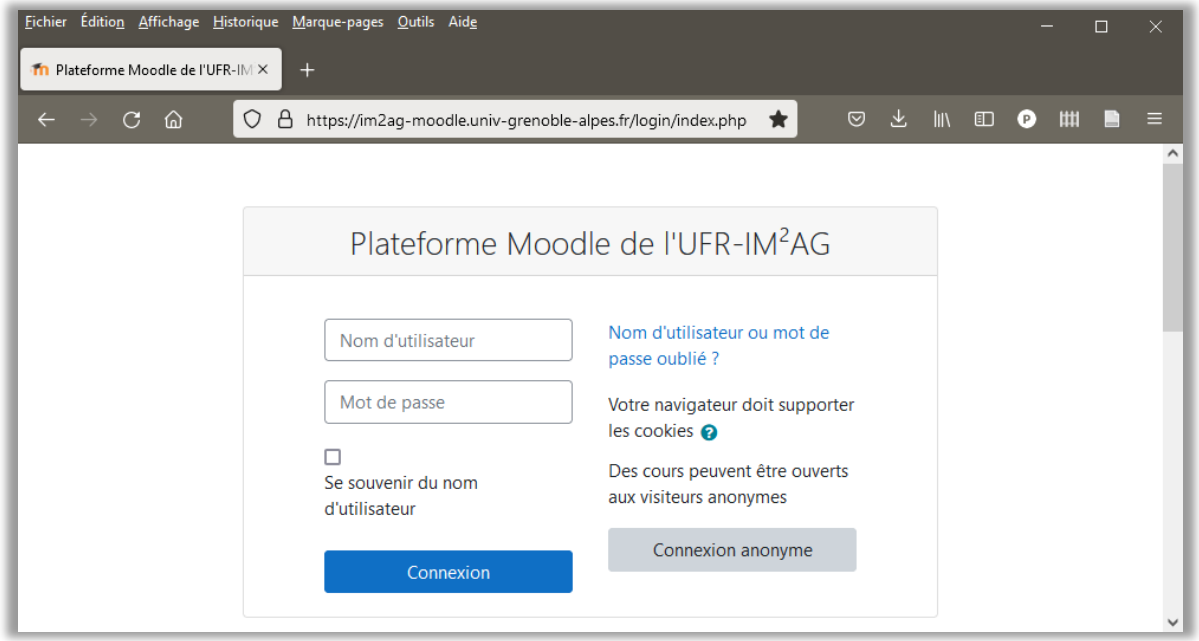
- HTTP est un **protocole applicatif**:
  - HTTP définit la forme de la communication entre un client web et un serveur web, c'est-à-dire la syntaxe des messages envoyés afin que les deux applications puissent effectivement communiquer.
  - HTTP n'indique pas comment les messages doivent transiter pour partir d'une machine A et arriver à une machine B.

OSI : Open Systems Interconnection



HTTPS ?

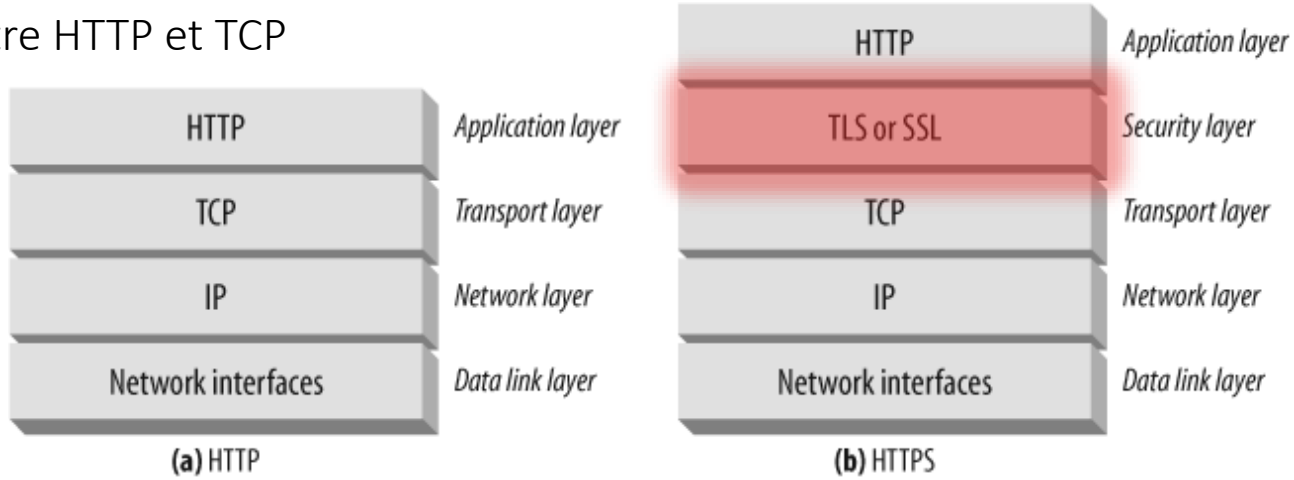
```
https://im2ag-moodle.univ-grenoble-alpes.fr/login/index.php
```



- Hyper Text Transfert Protocol Secure
- permet des échanges sécurisés (cryptés) entre le client et le serveur HTTP

HTTPS : Hyper Text Transfert Protocol Secure

- insère un couche de chiffrement des données entre HTTP et TCP
  - couche SSL (Secure Socket Layers) ou TLS (Transport Layer Security).

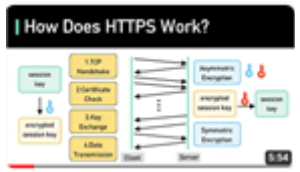


HTTP: The Definitive Guide by David Gourley, Brian Totty, Marjorie Sayer, Anshu Aggarwal, Sailu Reddy Editions O'Reilly 2002

  - utilisation de certificats d'authentification (émis par une autorité tierce réputée fiable)
    - permet vérification l'identité du site auquel le visiteur accède
    - peut permettre de valider l'identité du visiteur (si celui-ci utilise également un certificat d'authentification client).
  - garantit (théoriquement) la confidentialité et l'intégrité des données envoyées par l'utilisateur et reçues du serveur.
    - en particulier les informations entrées dans les formulaires

Pour en savoir plus :

C'est quoi SSL, SSH, HTTPS ?
http://sebsauvage.net/comprendre/ssl/



SSL, TLS, HTTPS Explained (6min)
https://www.youtube.com/watch?v=j9QmMEWmcfo

A complete overview of SSL/TLS and its cryptographic system (37 min)
https://www.youtube.com/watch?v=-f4Gbk-U758

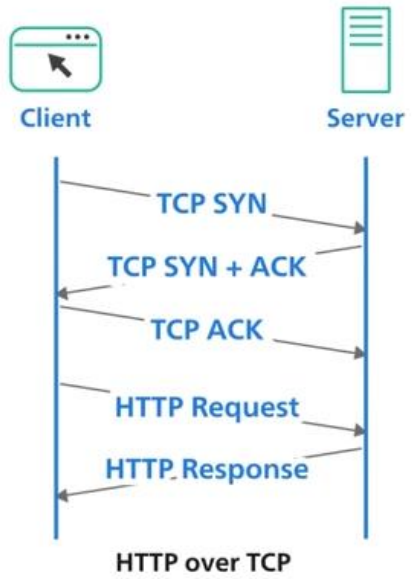
HTTP un protocole évolutif

- HTTP/0.9 (version de HTTP utilisée entre 1991 et 1996).
  - les requêtes se font sur une ligne ;
  - la seule méthode disponible est la méthode GET ;
  - le serveur ferme immédiatement la connexion après l'envoi de la réponse.

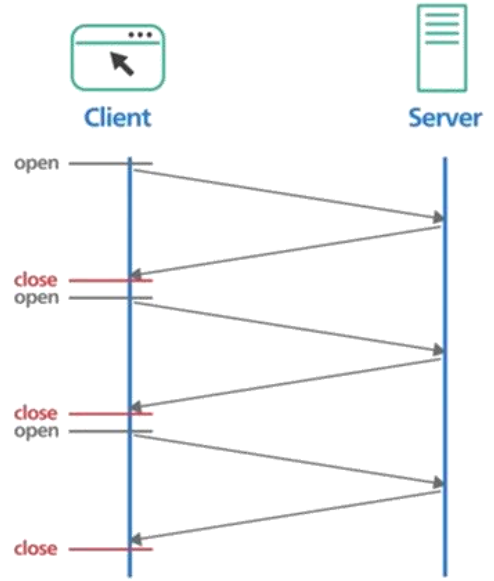
requête `GET /monfichier.html` →

réponse ←
```
<HTML>
Une page HTML très simple
</HTML>
```

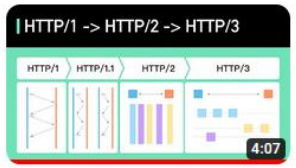
- HTTP/1.0 première version officielle de HTTP (1996).
  - introduit de nombreuses nouvelles fonctionnalités (GET et POST) et les requêtes HTTP deviennent plus complexes (en tête de requêtes, en tête et code de statut dans les réponses…).
  - Le système de connexion reste le même: dès l'envoi d'une réponse, le serveur clôt la connexion



HTTP over TCP



figures d'après
HTTP/1 to HTTP/2 to HTTP/3
ByteByteGo



https://www.youtube.com/watch?v=a-sBfyiXysI

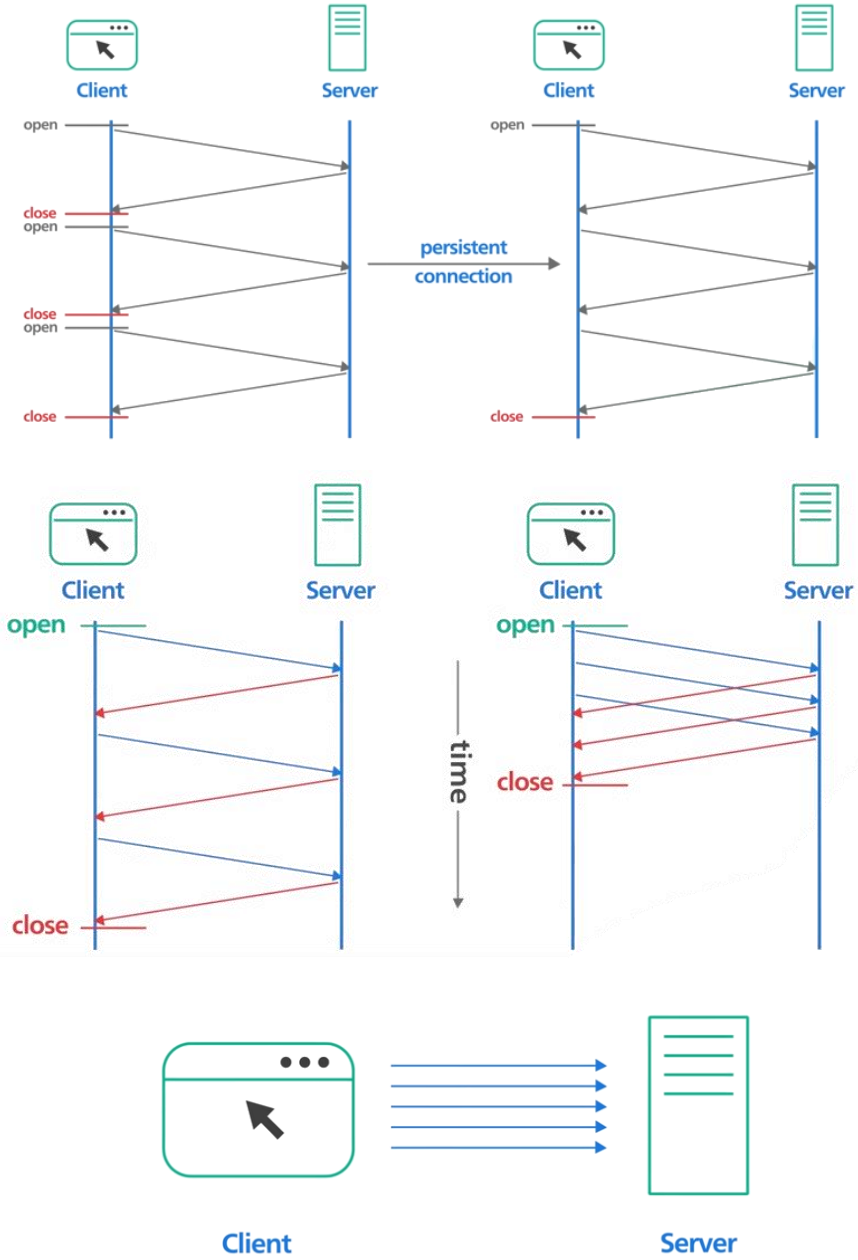
HTTP un protocole évolutif

- HTTP/1.1 standardisation et recherche de performances (1997)
  - introduction de nouvelles fonctinnalités (méthodes), nouveaux en-têtes, négociation de contenu
  - mécanismes de connexion plus performants

    - connexions persistantes ou *"keep-alive"*
      - après la réponse du serveur la connexion est maintenue dans l'attente d'une nouvelle requête.
      - permet d'économiser le temps de la connexion et de la déconnexion

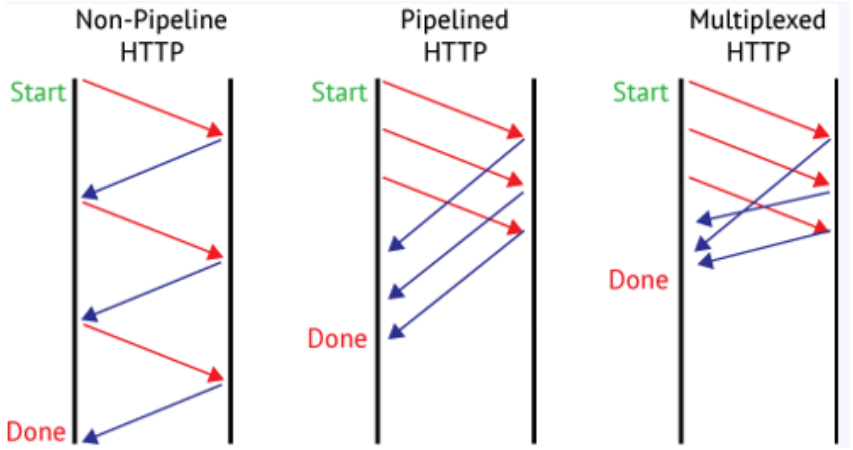
    - connexions en pipeline
      - permet au client d'envoyer une nouvelle requête avant d'avoir reçu la réponse du serveur.
      - Les réponses du serveur doivent cependant arriver dans le même ordre que celui de l'envoi des requêtes.

    - connexions parallèles
      - permet d'établir plusieurs connexions simultanées de façon à pouvoir envoyer plus de requêtes en même temps.



HTTP un protocole évolutif

- HTTP/2 (2015)
  - modifie la façon dont les données sont formatées et transportées entre le client et le serveur sans toucher à la sémantique d'application de HTTP(méthodes HTTP, les codes d'état, les URI et les champs d'en-tête sont conservés tels quels)
  - trouve son origine dans un autre protocole appelé SPDY (prononcer *speedy*) et développé par Google dont l'objectif était de réduire la durée de téléchargement des pages Web.
  - les requêtes sont classées par ordre de priorité (de sorte à ce que les requêtes les plus importantes comme le chargement d'un fichier de style ne soient pas bloquées par d'autres moins importantes comme le chargement d'une image en bas de page).
  - mise en place d'un mécanisme de multiplexing.



permet d'envoyer plusieurs requêtes d'affilée lors d'une même connexion sans attendre à chaque fois la réponse du serveur

les réponses du serveur peuvent revenir dans n'importe quel ordre.

  - compresse les en-têtes, étant donné que des en-têtes similaires sont échangés lors d'une suite de requêtes, on supprime ainsi la duplication et l'échange inutiles des données similaires.
  - mécanisme de push, permet au serveur de remplir le cache du client avant qu'il ne soit demandé par ce dernier, on parle alors d'évènements générés par le serveur.

HTTP un protocole évolutif

- HTTP/3
  - dernière évolution de HTTP
    - juin 2022, Proposed standard publié par IETF (Internet Engineering Task Force)

Internet Engineering Task Force (IETF) M. Bishop, Ed.
Request for Comments: 9114 Akamai
Category: Standards Track
Published: June 2022
ISSN: 2070-1721

HTTP/3

Abstract

The QUIC transport protocol has several features that are desirable in a transport for HTTP, such as stream multiplexing, per-stream flow control, and low-latency connection establishment. This document describes a mapping of HTTP semantics over QUIC. This document also identifies HTTP/2 features that are subsumed by QUIC and describes how HTTP/2 extensions can be ported to HTTP/3.

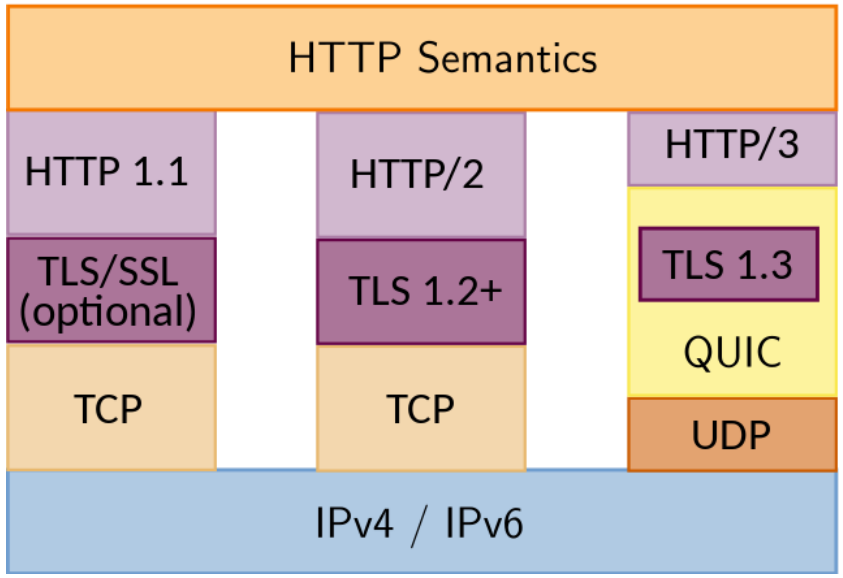
Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has

[Request for Comments (RFC) 9114]

  - basé sur des travaux de Google sur le protocole QUIC (Quick UDP Internet Connections)



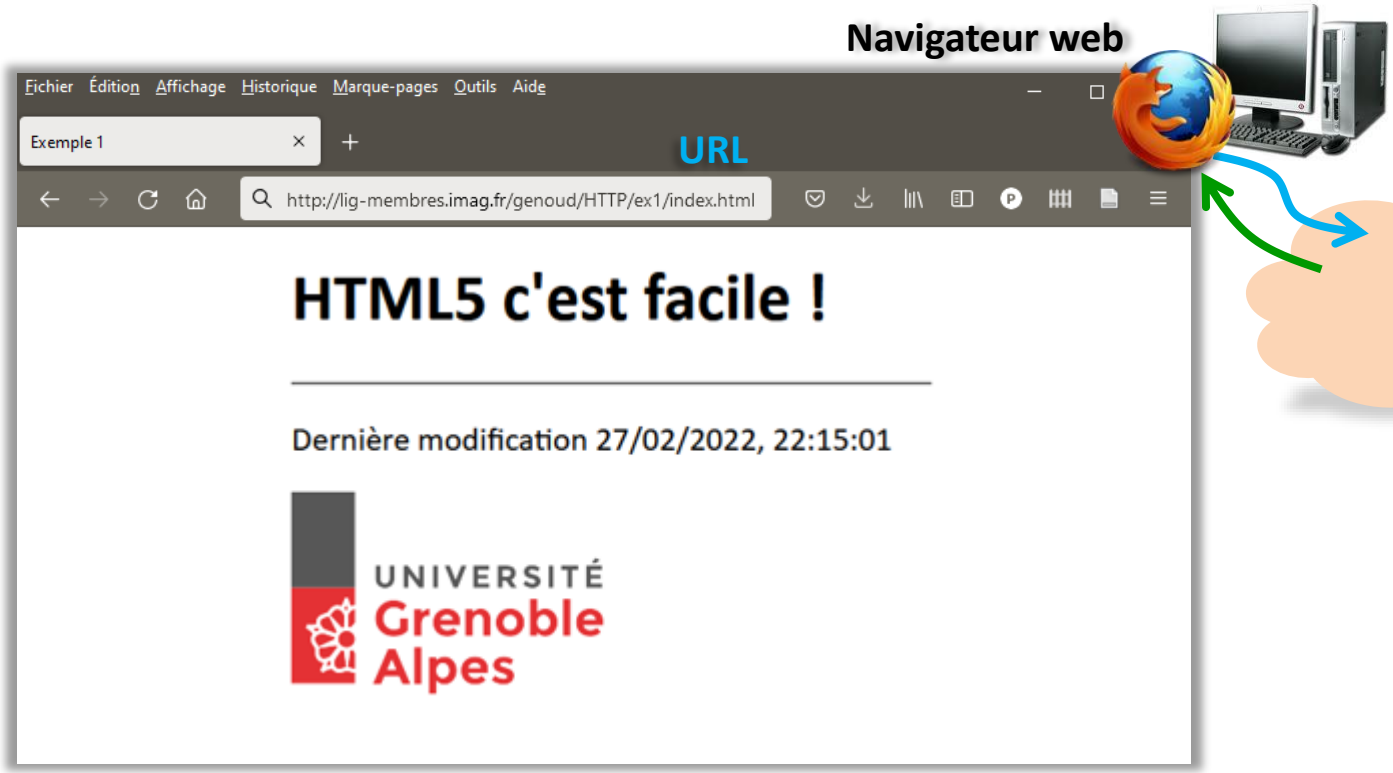
https://en.wikipedia.org/wiki/File:HTTP-1.1_vs._HTTP-2_vs._HTTP-3_Protocol_Stack.svg

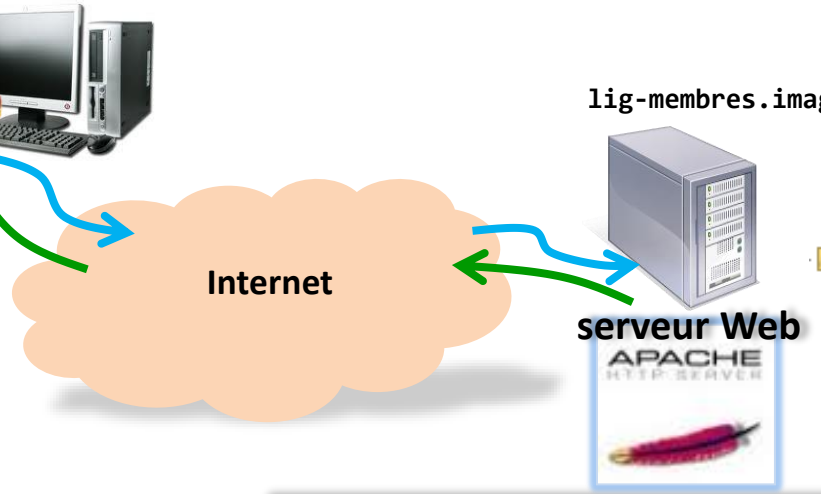
utilise une sémantique similaire à celle des versions précédentes du protocole (mêmes méthodes de requête, codes d'état et champs de message.

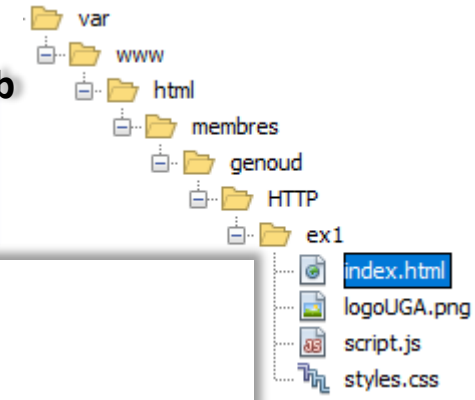
mais utilise le protocole de transport UDP plutôt que le protocole TCP pour proposer des connexions HTTP et des transferts de données plus rapides

déjà implémenté dans la plupart des navigateurs modernes, et proposé par un nombre croissant de serveurs HTTP

HTTP : Que se passe-t-il quand on charge une page HTML ?








```
<!DOCTYPE html>
<html>

<head>
    <title>Exemple 1</title>
    <meta charset="UTF-8">
    <meta name="viewport"
        content="width=device-width, initial-scale=1.0">
    <link href="styles.css" rel="stylesheet">
</head>

<body>
    <div class="wrapper">
        <h1>HTML5 c'est facile !</h1>
        <hr />
        <p>
            Dernière modification <span id="date-modif"></span>
        </p>
        <img src="./logoUGA.png" alt="le logo uga">
    </div>
    <script src="script.js"></script>
</body>

</html>
```

https://lig-membres.imag.fr/genoud/HTTP/ex1/index.html

protocole — serveur — chemin d'accès (path) — la ressource

Quels sont les échanges entre un client web (navigateur) et le serveur web ?

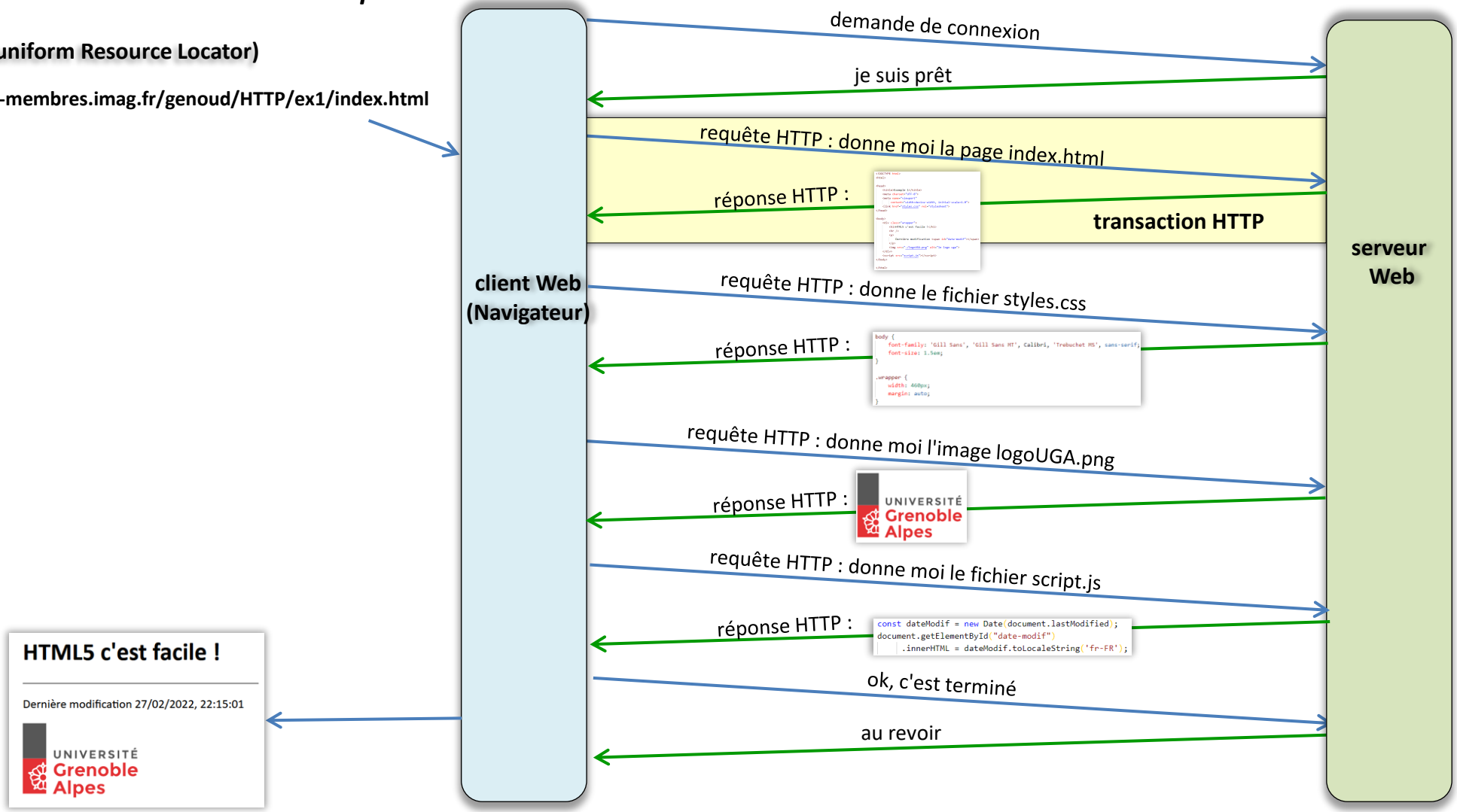
HTTP 1.1

Déroulement du chargement de la page

- connexion *keep-alive*

URL (uniform Resource Locator)

https://lig-membres.imag.fr/genoud/HTTP/ex1/index.html

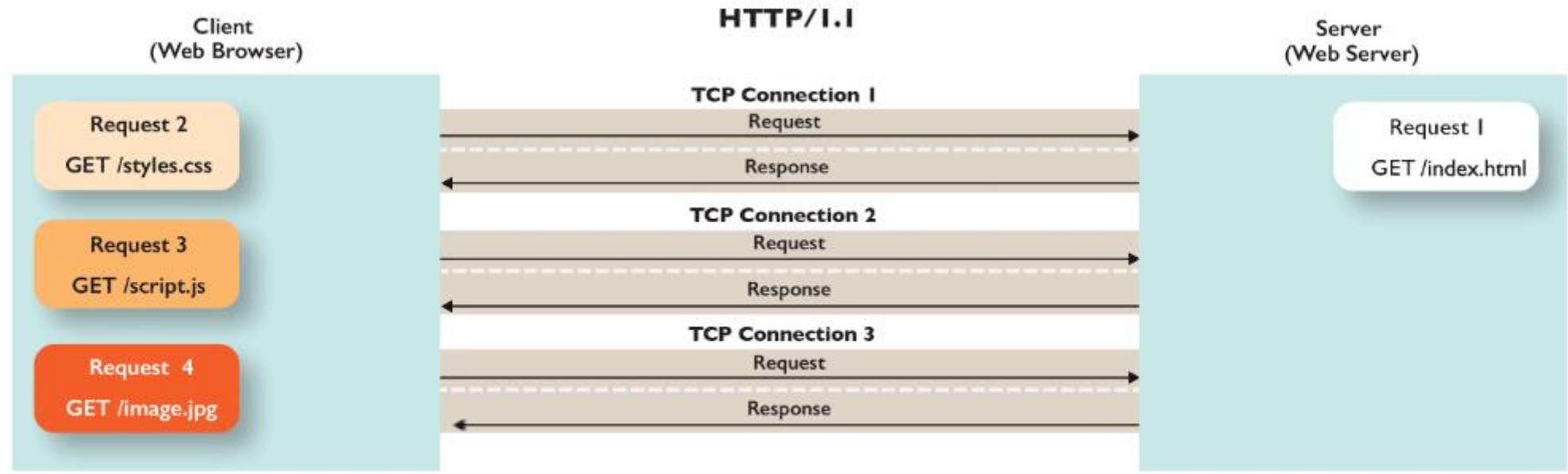


transaction HTTP :
requête et réponse HTTP vont toujours de paire

HTTP 1.1

Déroulement du chargement de la page

- connexions parallèles



From HTTP/2 IN ACTION by BARRY POLLARD, Copyright 2018.

https://freecontent.manning.com/animation-http-1-1-vs-http-2-vs-http-2-with-push/

HTTP 1.1

Déroulement du chargement de la page

- connexions parallèles

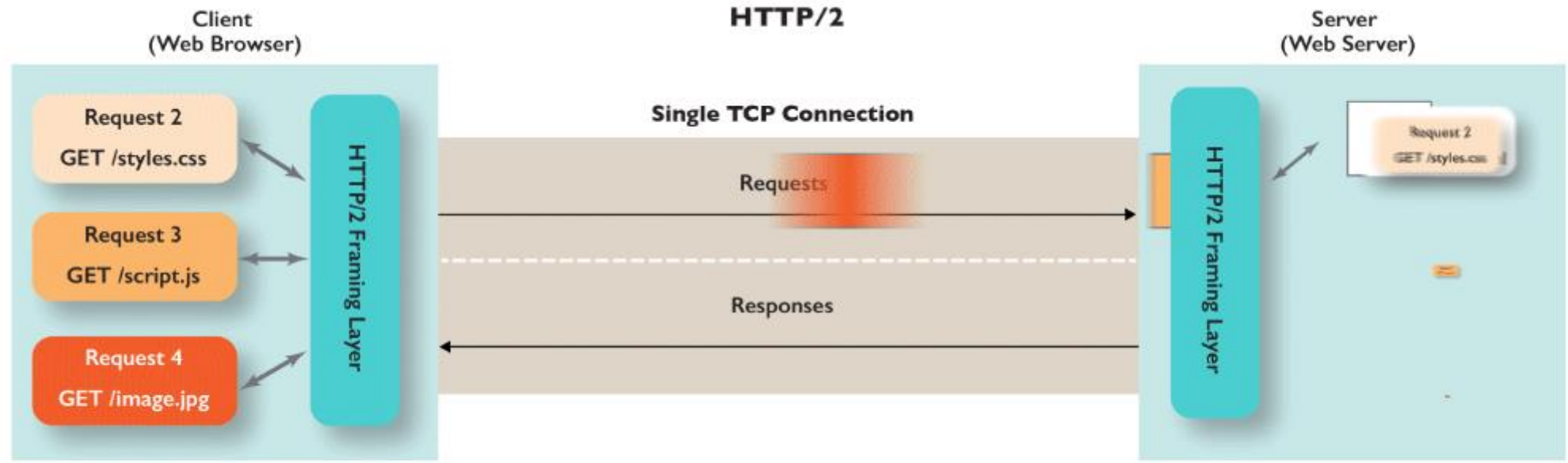
From HTTP/2 IN ACTION by BARRY POLLARD, Copyright 2018.

https://freecontent.manning.com/animation-http-1-1-vs-http-2-vs-http-2-with-push/

HTTP/2

Déroulement du chargement de la page

- connexions multiplexées



From HTTP/2 IN ACTION by BARRY POLLARD, Copyright 2018.

https://freecontent.manning.com/animation-http-1-1-vs-http-2-vs-http-2-with-push/

HTTP/2

Déroulement du chargement de la page

- connexions multiplexées

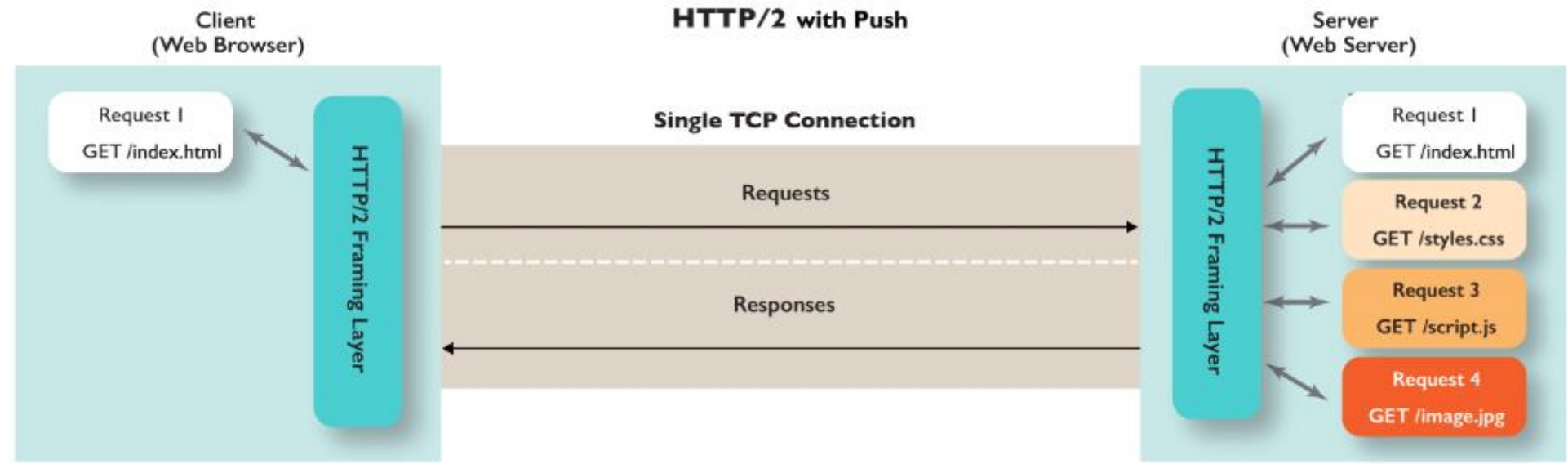
From HTTP/2 IN ACTION by BARRY POLLARD, Copyright 2018.

https://freecontent.manning.com/animation-http-1-1-vs-http-2-vs-http-2-with-push/

HTTP/2

Déroulement du chargement de la page

- connexion avec push



https://freecontent.manning.com/animation-http-1-1-vs-http-2-vs-http-2-with-push/

HTTP/2

Déroulement du chargement de la page

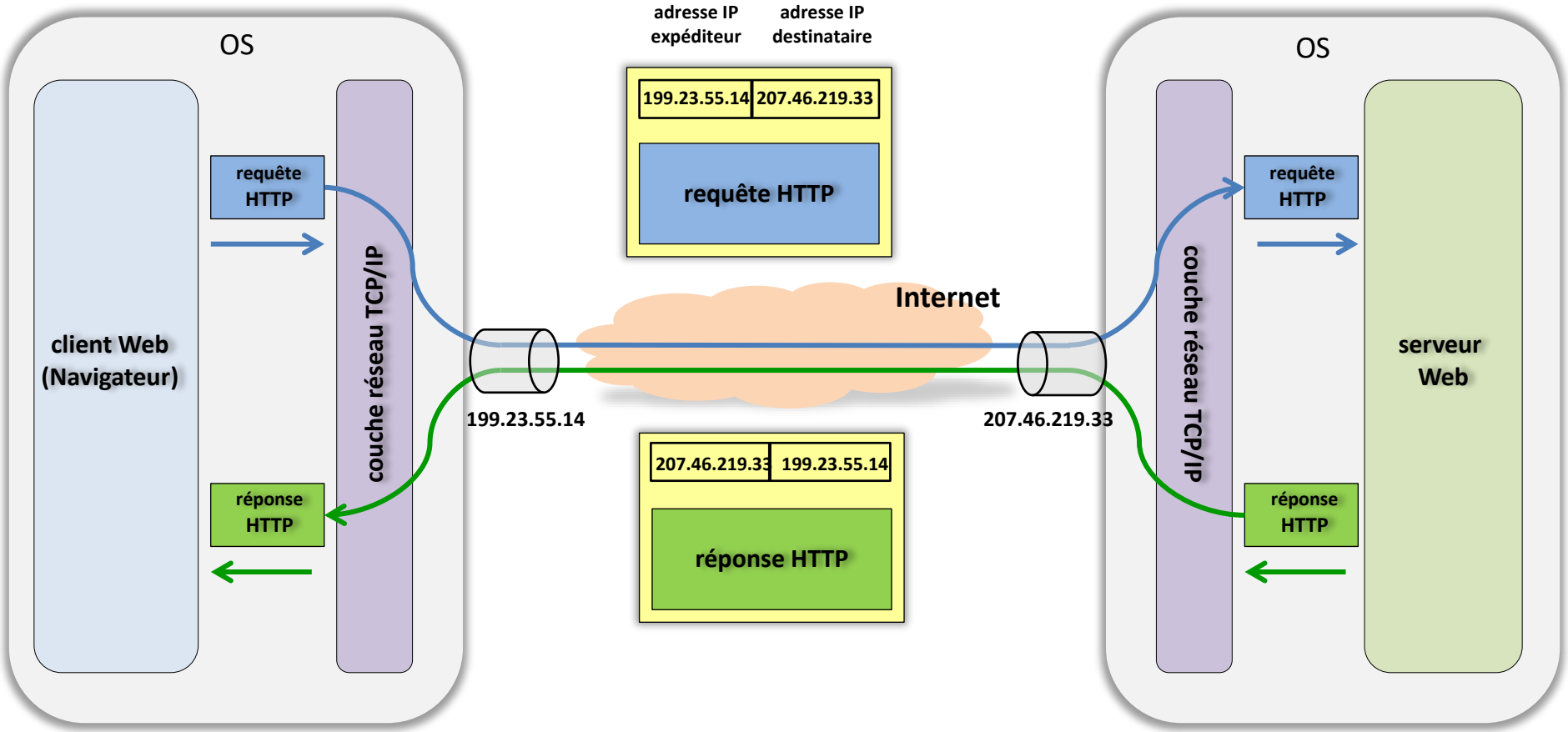
- connexion avec push

From HTTP/2 IN ACTION by BARRY POLLARD, Copyright 2018.

https://freecontent.manning.com/animation-http-1-1-vs-http-2-vs-http-2-with-push/

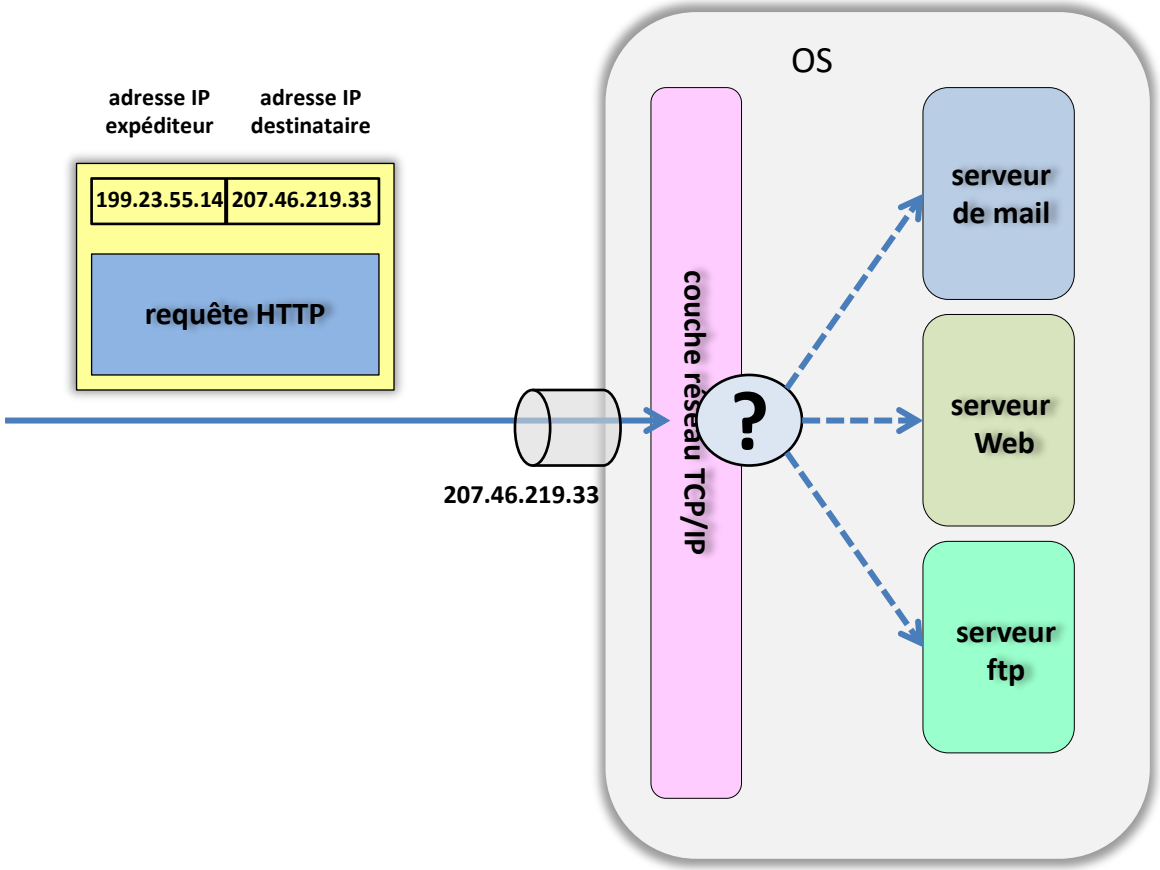
HTTP 1.1/HTTP2

- requête et réponse HTTP passent par la couche de transport TCP/IP
  - rajoute infos nécessaires au routage
  - découpage en paquets
  - reconstitution des paquets



HTTP 1.1/HTTP 2

- n° IP permet de transférer les paquets d'un ordinateur à un autre
- problème : lorsqu'un paquet arrive comment savoir à quel logiciel le paquet est destiné ?



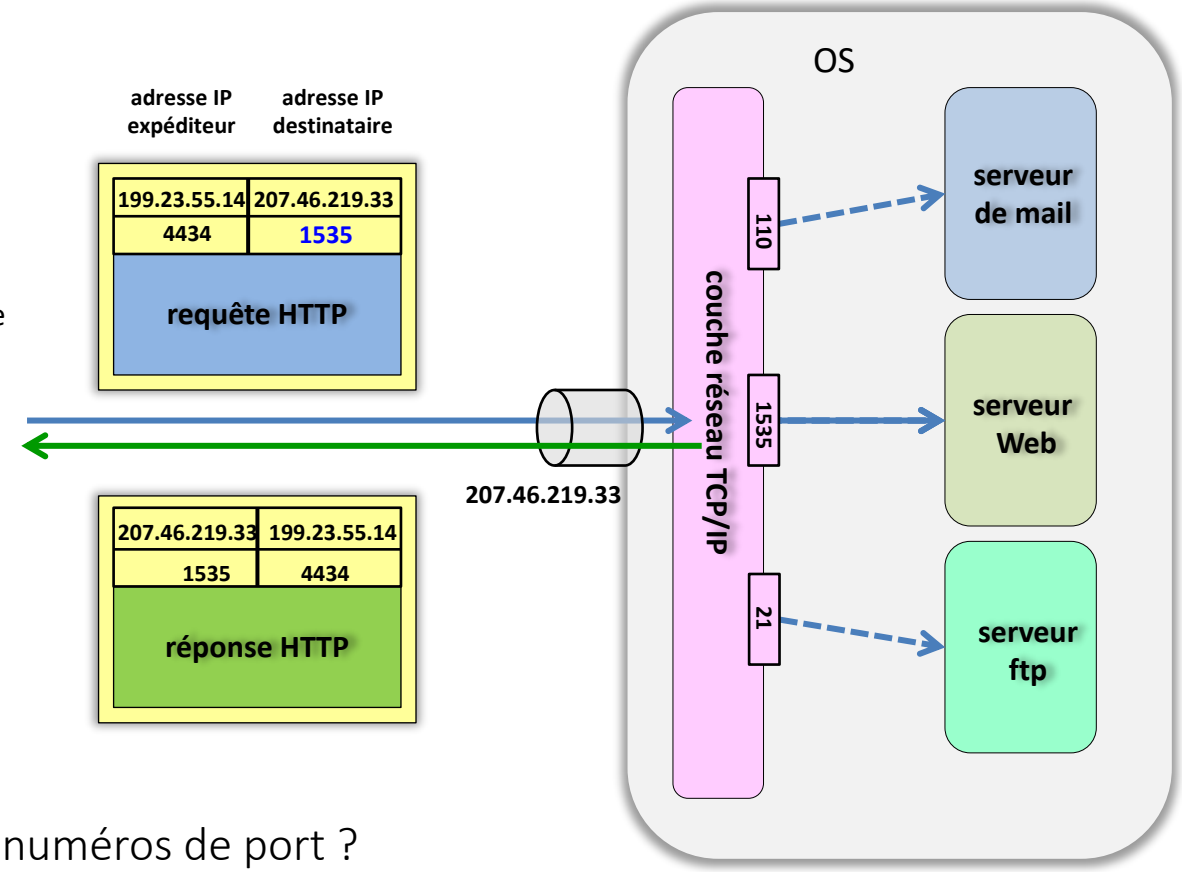
HTTP 1.1/HTTP 2

- n° IP : adresse de l'ordinateur destinataire
- n° de port : identifie le logiciel destinataire
  - entier sur 2 octets (0..65535)

n°IP + n° Port : socket

plusieurs clients peuvent tourner sur la machine expéditeur

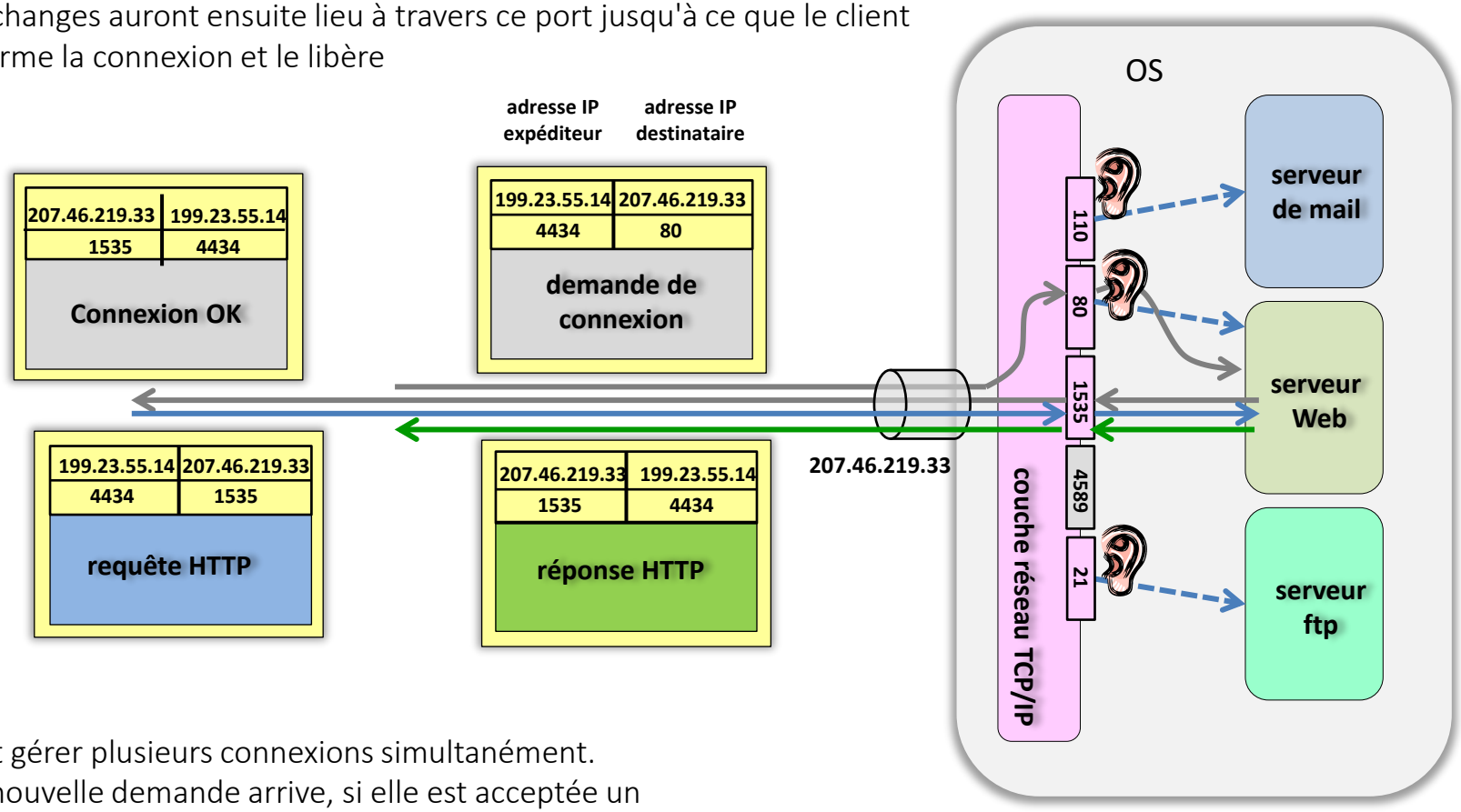
de manière symétrique nécessaire d'avoir un numéro de port sur l'expéditeur pour envoyer la réponse



Comment sont fixés les numéros de port ?

HTTP 1.1 / HTTP 2

- un port prédéterminé (port d'écoute) est attribué au programme serveur pour les demandes de connexion
- Lors d'une demande de connexion
  - le client envoie une demande sur le port d'écoute
  - si le serveur accepte la connexion un port libre est sélectionné et attribué à celle-ci ; le serveur envoie un acquittement
  - échanges auront ensuite lieu à travers ce port jusqu'à ce que le client ferme la connexion et le libère



Le numéro du port d'écoute est fixé à l'avance.
Valeur par défaut :

- **80 pour serveur web HTTP**
- **443 pour HTTPS**
- 110 serveur POP
- 21 server ftp
- ….

pour connaitre les numéros de port attribués : https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

Le numéro du port d'écoute peut être modifié par l'administrateur du serveur

serveur peut gérer plusieurs connexions simultanément. Lorsqu'une nouvelle demande arrive, si elle est acceptée un nouveau port libre lui est attribué.

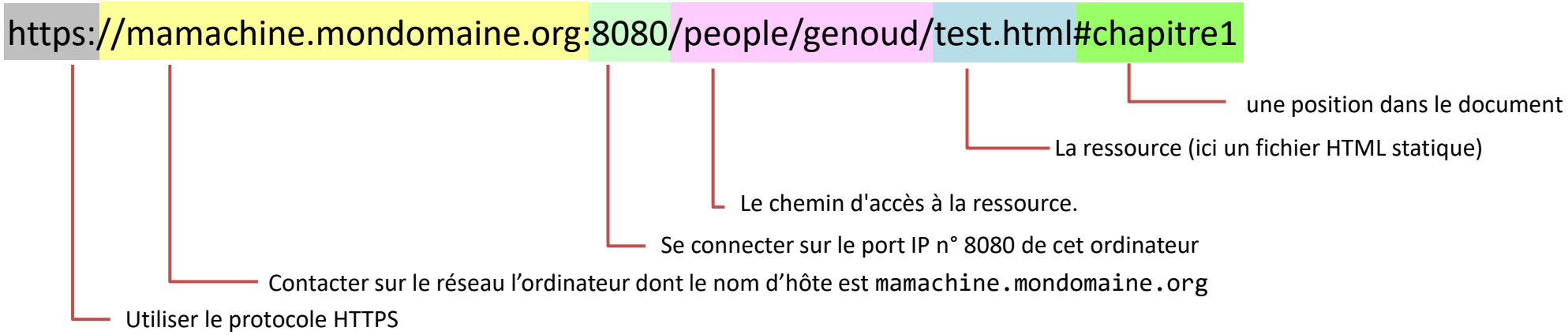
HTTP

URL
forme générale

- URL (Uniform Resource Locator) → localisation d'une ressource sur le web
- Syntaxe:

```
méthode://nomserveur[:port][/path/resource[?params][#ancre]]
```

  - méthode : nom du protocole permettant d'y accéder (http, https, mailto, ftp, file, news …)
  - nomserveur : le nom ou numéro IP de la machine hébergeant le serveur HTTP,
  - port: le numéro du port d'écoute (port par défaut si non spécifié)
  - path : le chemin pour accéder au document
  - resource : nom de la ressource.



URL
exemples

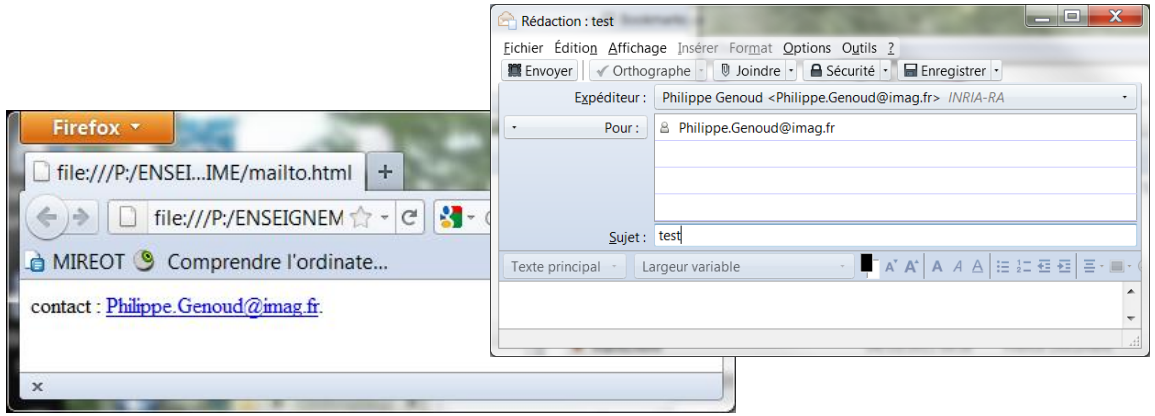
https://mamachine.mondomaine.org:8080/people/genoud/test.html#chapitre1

https://www.google.com google.com

http://123.89.34.80/glpi/front/helpdesk.public.php?show=resa&mois_courant=10

file:///C:/wamp/www/genoud/exempleXHTML.xhtml

mailto:Philippe.Genoud@imag.fr



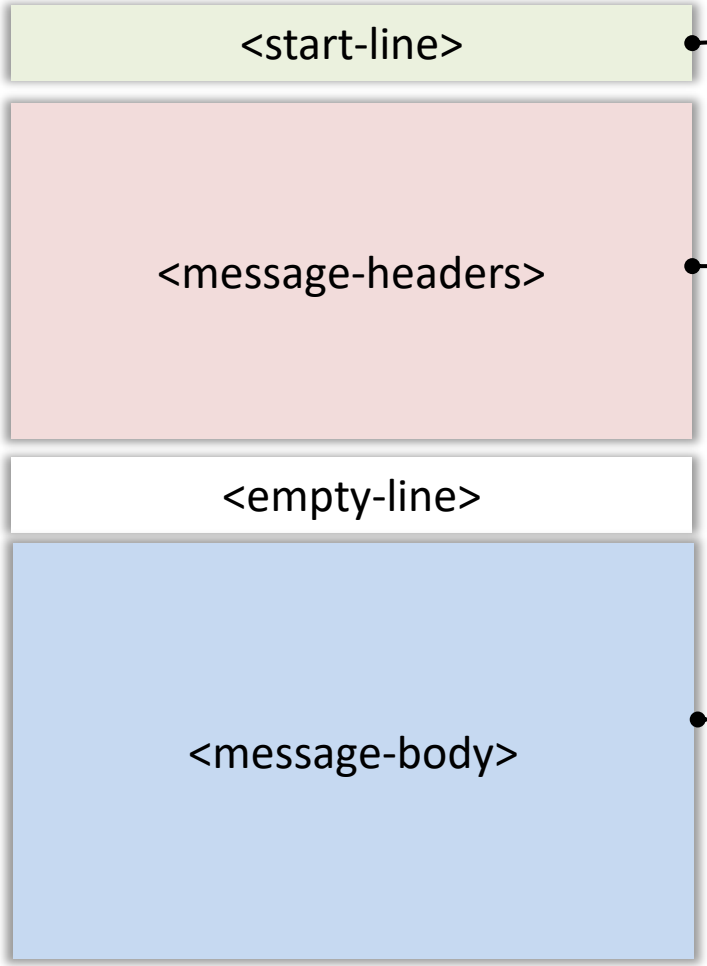

```
contact : <a href="mailto:Philippe.Genoud@imag.fr">Philippe.Genoud@imag.fr</a>.
```

Transaction HTTP

- transaction HTTP
  - échange entre un client web et un serveur web
  - requête du client + réponse du serveur (vont toujours de pair)

Format générique des messages HTTP

- Structure commune partagée par tous les messages HTTP (requêtes et réponses)



Détermine si le message est une requête HTTP ou une réponse HTTP.
Contient de l'information spécifique selon la nature (requête ou réponse) du message.

De simples paires noms-valeur sous la forme *header-name*: *header-value*
Exemples :

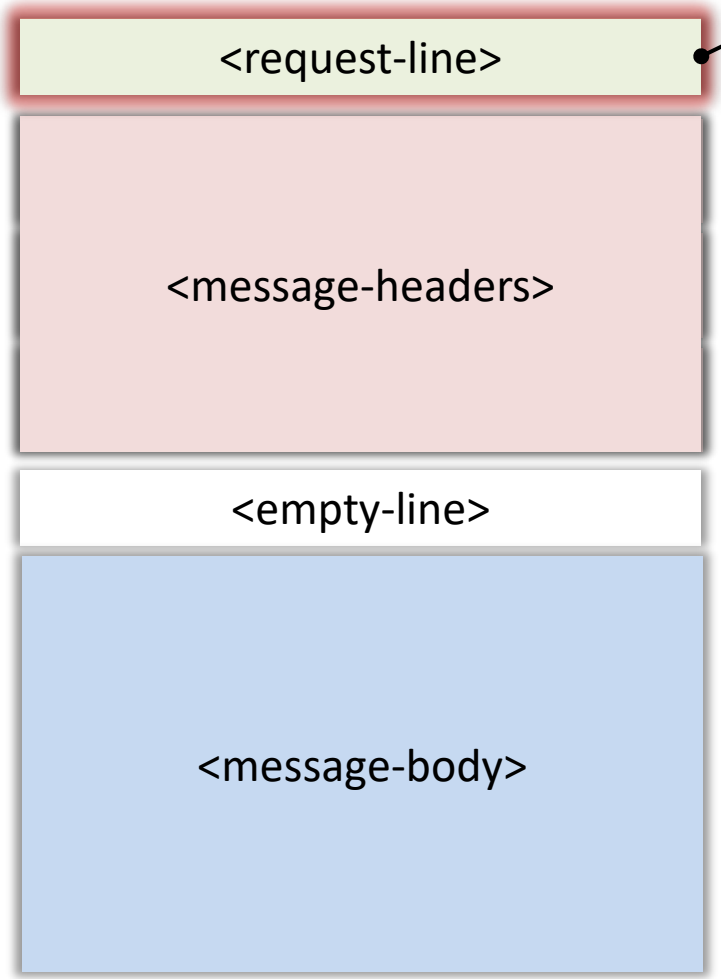
```
User-Agent: Mozilla/5.0 (Windows NT 10.0; …) Gecko/20100101 Firefox/64.0
Server: Apache/2.2.15 (CentOS)
```

Contient l'information effective,
par exemple pour une réponse le code HTML de la page demandé.

Cette partie est optionnelle, tous les messages HTTP n'en ont pas nécessairement besoin

Format des requêtes HTTP

- Messages envoyés par le client



Composé de trois parties

`méthode URI version`

version du protocole HTTP utilisée par le client :
HTTP/3.0, HTTP/2.0, HTTP/1.1 ou HTTP/1.0

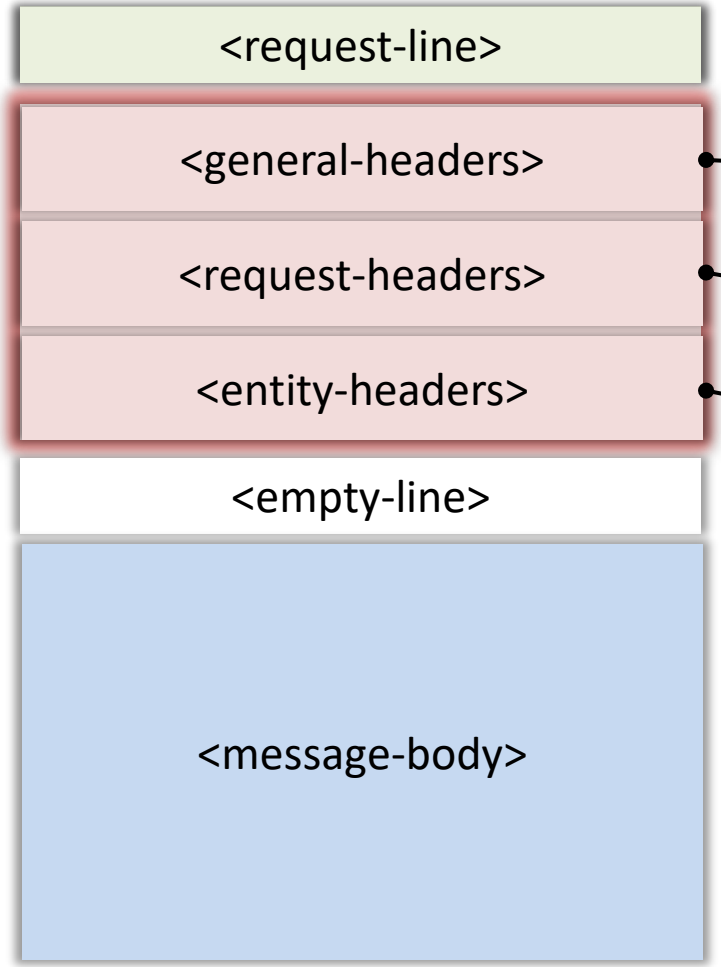
identifie la ressource concernée par l'action

commande qui spécifie au serveur l'action à effectuer

- GET : méthode la plus courante pour demander une ressource. Une requête GET est sans effet sur la ressource, il doit être possible de répéter la requête sans effet.
- POST : utilisée pour soumettre des données en vue d'un traitement à une ressource (typiquement 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.
- PUT : permet de remplacer ou d'ajouter une ressource sur le serveur. L'URI fourni est celui de la ressource en question.
- DELETE : permet de supprimer une ressource du serveur.
- HEAD : ne demande que des informations sur la ressource, sans demander la ressource elle-même.
- PATCH : permet de modifier partiellement une ressource
- OPTIONS : permet d'obtenir les options de communication d'une ressource ou du serveur en général.
- CONNECT : permet d'utiliser un proxy comme un tunnel de communication.
- TRACE: demande au serveur de retourner ce qu'il a reçu, dans le but de tester et effectuer un diagnostic sur la connexion.
- PUT : permet de remplacer ou d'ajouter une ressource sur le serveur. L'URI fourni est celui de la ressource en question.

Format des requêtes HTTP

- Messages envoyés par le client



La section `<message-headers>` contient trois types d'en-têtes

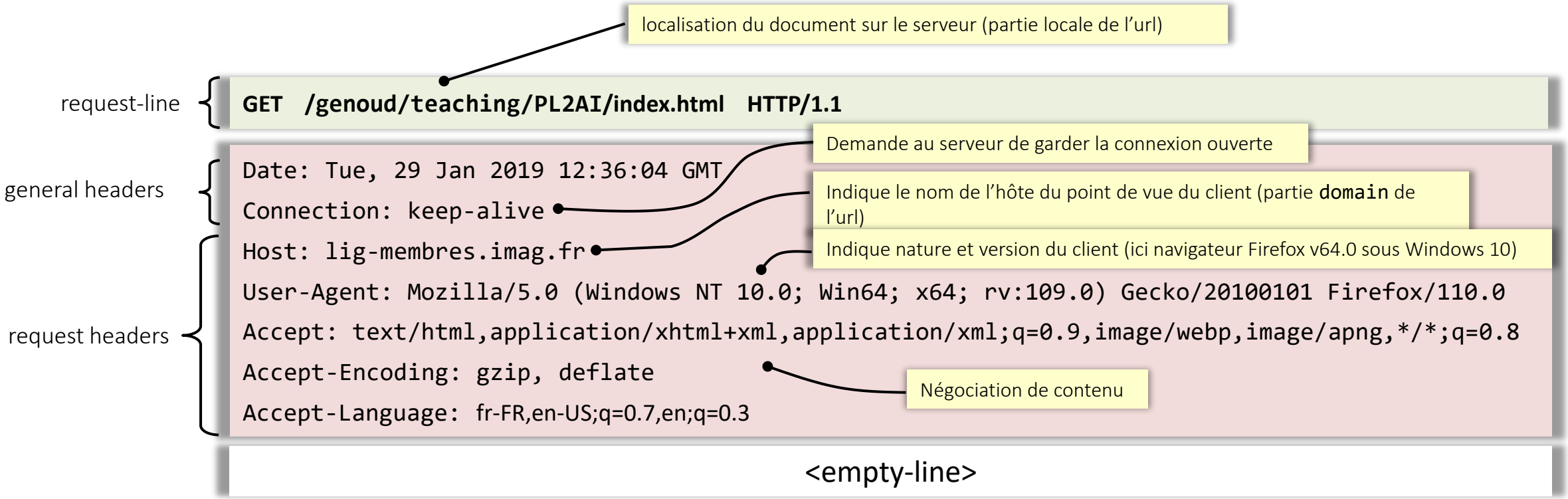
En-têtes concernant le message lui-même et qui ne sont pas spécifiques à une requête particulière.
Ex: date de la requête

En-têtes contenant des détails sur le message lui-même
Ex: le nom de du serveur hôte à qui la requête est destinée

En-têtes décrivant le corps du message si il est présent
Ex: la taille du message

Format des requêtes

- Exemple d'une requête GET
  - Requête envoyée par un navigateur lorsque l'on demande la page d'accueil du cours PL2AI https://lig-membres.imag.fr/genoud/teaching/PL2AI/index.html

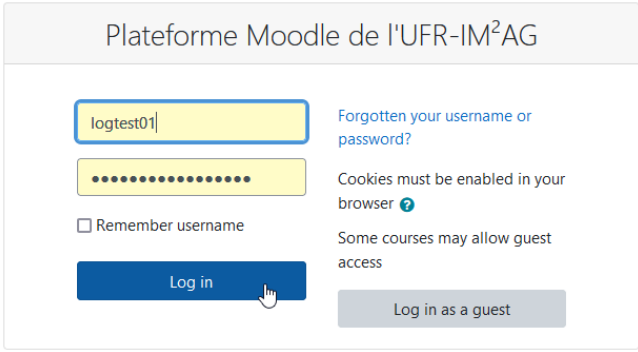


entity-headers et message-body sont vides

HTTP

Format des requêtes

- Exemple d'une requête POST
  - Requête envoyée par un navigateur lorsque l'on soumet le formulaire de connexion https://im2ag-moodle.univ-grenoble-alpes.fr /login/index.php



request-line
POST /login/index.php HTTP/1.1

general headers
request headers
entity headers

```
Connection: keep-alive
Host: im2ag-moodle.univ-grenoble-alpes.fr
...
Content-Type: application/x-www-form-urlencoded
Content-Length: 94
...
```

Type MIME du contenu dans le corps du message

Taille du contenu (en octets)

empty line
<empty-line>

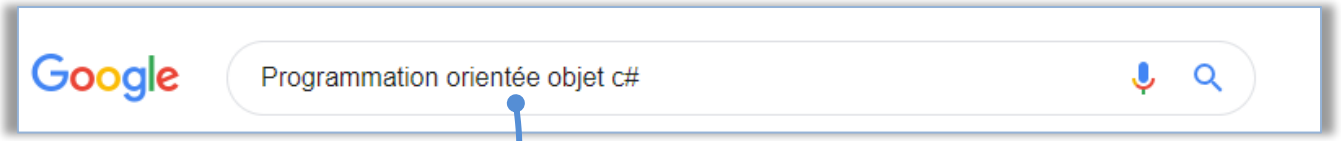
message body
`username=logtest01&password=logtest%26%25*&anchor=&logintoken=3mQiOB4NLECFeOIbCh3kmOFpBhsjZtwB`

Champs du formulaire (*form fields*) sous la forme de couples clé valeur url-encodé

```
username=logtest01
password=logtest&%*
anchor=
logintoken=3mQiOB…Ztw
```

URL encoding (Percent encoding)

- Encodage pour les données ajoutées à la fin d'une l'URL
  → ne peuvent contenir espaces, saut de ligne ….
  → format spécial : URL-Encoded



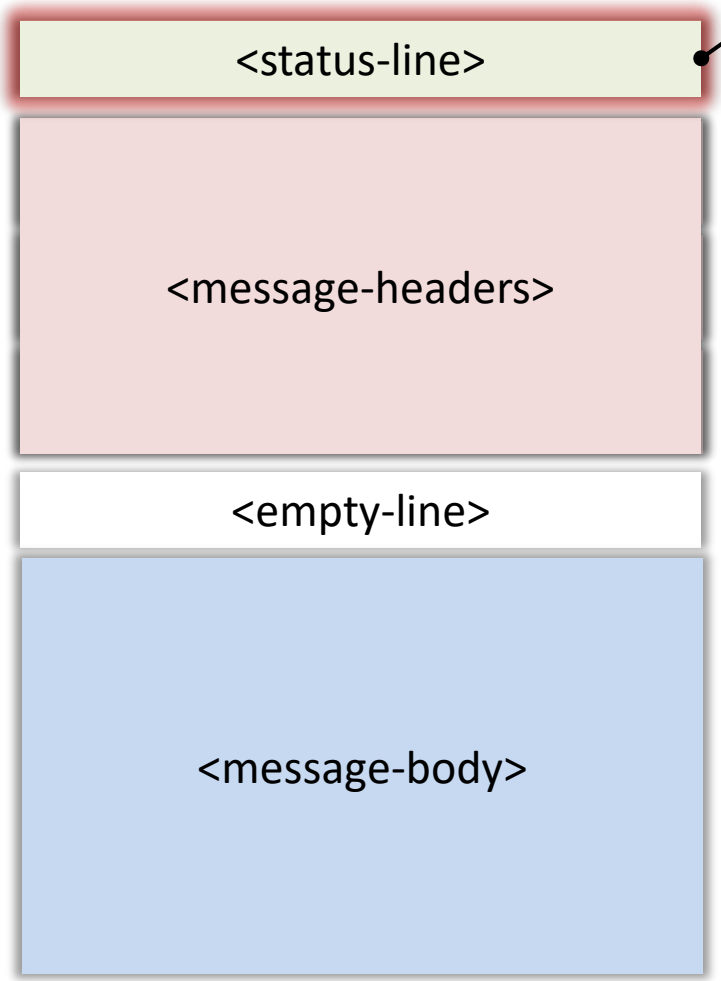
`https://www.google.com/search?q=Programmation+orient%C3%A9e+objet+c%23&...`

- le format URL-Encoded
  - une seule ligne
  - suite de paires ***`nomVariable=valeur`*** séparées par &
  - espaces remplacés par '+' ou %20
  - les caractères ayant un sens spécial ('=', '&', '<' …) ou les caractères accentués sont remplacés par % suivi de leur code en héxadecimal
    - ex : '=' → %3D (voir http://www.w3schools.com/tags/ref_urlencode.asp)
  - fonctions dans les langages de programmation pour encoder/décoder les URL : classes **`URLDecoder`** et **`URLEncoder`** du package **`java.net`**, fonctions **`encodeURI()`** et **`decodeURI()`** JavaScript, **`rawurlencode()`** PHP, **`Server.URLEncode()`** ASP…

HTTP

Format des réponses

- Réponses envoyées par le serveur



Composé de trois parties

`version – status code – texte réponse`

texte explicatif

code d'état : informe le client du traitement de la requête par le serveur

version du protocole HTTP utilisée par le serveur : ne peut être plus grand que celui de la requête

- code d'état (sur 3 chiffres) répartis en 5 groupes
  - **1xx** : message d'information
    - Exemple : **101 Switching Protocols**, le client à demandé de changer de protocole le serveur envoie un message de confirmation
  - **2xx** : requête du client accomplie avec succès
    - Exemple: **200 OK**
  - **3xx** : requête du client redirigée, d'autres actions sont nécessaires
    - Exemple: **303 Moved Permanently**
  - **4xx** : erreur du client
    - Exemple: **404 Resource Not Found**
  - **5xx** : erreurs du serveur. Elle peut provenir du serveur lui-même, mais plus généralement d'un programme serveur (Perl, PHP, ASP, Java….) chargé de générer la réponse.
    - Exemple: **503 Service Unavailable**, le serveur ne peut répondre à la requête, par exemple suite à une surcharge
- la majorité de ces codes d'état sont traités de manière transparente pour l'utilisateur sauf certains codes des classes 4 et 5 (ex: 404 Not Found)

https://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml

- code d'état de classe 2 (succès)
  Cette classe de codes d'état indique que l'action demandée par le client a été reçue, comprise et acceptée.

  - 200 OK : Réponse standard pour les requêtes HTTP traitées avec succès.
    - La réponse réelle dépendra de la méthode de requête utilisée. Dans une requête GET, la réponse contiendra une entité correspondant à la ressource demandée. Dans une requête POST, la réponse contiendra une entité décrivant ou contenant le résultat de l'action.
  - 201 Created : requête a été traitée et la ressource a été créée
  - 202 Accepted : requête a été reçue et est en cours de traitement mais sans garantie de résultat. La connexion peut être interrompue
  - 204 No Content : requête a été traitée mais la réponse ne contient pas de corps. Utile pour programmes serveur (CGI,PHP,…) qui veulent accepter données d'un formulaire sans que la vue du navigateur ne change
  - 205 Reset Content : Le serveur a traité la demande avec succès, demande au client de réinitialiser sa vue de document (par exemple le navigateur devrait effacer le formulaire utilisé pour cette transaction) et ne renvoie aucun contenu.
  - …

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

HTTP

- code d'état de classe 3 (redirection – traitement incomplet)
  Cette classe de code d'état indique que le client doit prendre des mesures supplémentaires pour terminer la demande. Beaucoup de ces codes d'état sont utilisés dans la redirection d'URL.

  – 301 Moved Permanently : La ressource a été assignée à une nouvelle adresse. L'URL est donnée par le champ `Location`
  – 302 Moved Temporarily : La ressource a été assignée temporairement à une nouvelle adresse. L'URL est donnée par le champ `Location`
  – 304 Not Modified : La ressource n'a pas été modifiée depuis la date précisée par champ `If-Modified-Since` de la requête. L'entité de corps n'est pas envoyée, le client doit utiliser sa propre copie du document (qui a été mise en cache).
  – …

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

HTTP

- code d'état de classe 4 (erreur client)
  Cette classe de code d'état est destinée aux situations dans lesquelles l'erreur semble avoir été causée par le client.

  - 400 Bad Request : Erreur de syntaxe dans la requête
  - 401 Unauthorized : La requête nécessite une identification préalable de l'utilisateur
  - 403 Forbidden : Le serveur a compris la requête, mais refuse de la traiter. Contrairement à l'erreur 401, s'authentifier ne fera aucune différence. Sur les serveurs où l'authentification est requise, cela signifie généralement que l'authentification a été acceptée mais que les droits d'accès ne permettent pas au client d'accéder à la ressource.
  - 404 Not Found : Le serveur n'a pas trouvé la ressource demandée
  - …

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

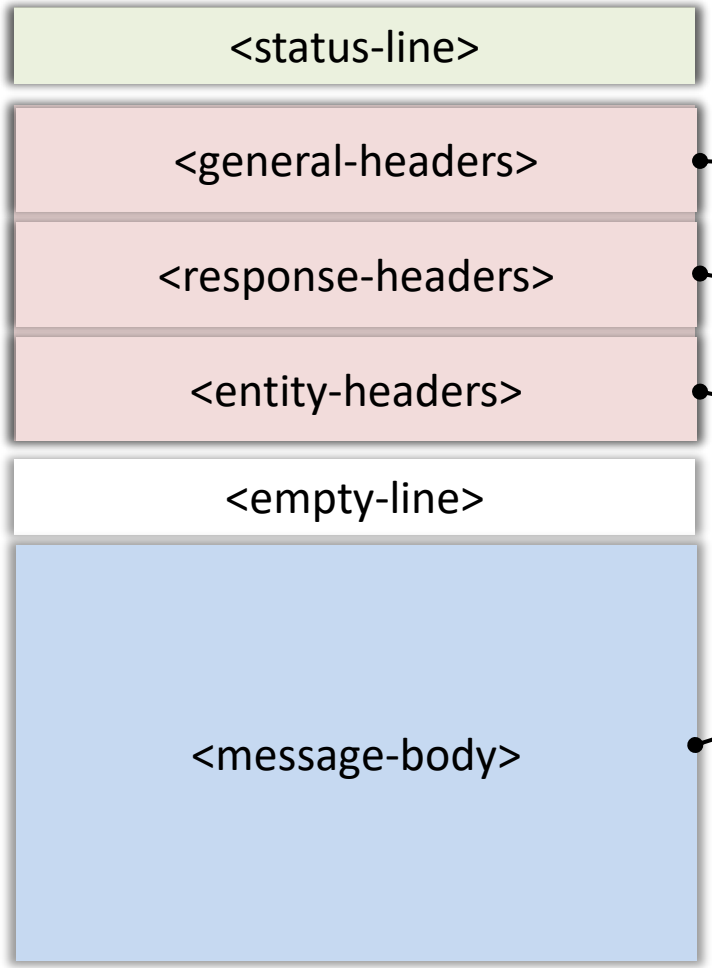
- code d'état de classe 5 (Erreur serveur)
  Les codes d'état de cette classe indiquent les cas dans lesquels le serveur est conscient qu'il a rencontré une erreur et est dans l'incapacité d'exécuter la demande.

  - 500 Internal Server Error : Erreur propre au serveur
  - 501 Not Implemented : Le serveur ne possède pas la fonctionnalité pour traiter la requête
  - 502 Bad Gateway : le serveur (ou proxy) a rencontré une réponse invalide en provenance d'un autre serveur ou proxy.
  - 503 Service Unavailable : Le serveur n'est pas en mesure de traiter la requête pour des raisons de surcharge ou de maintenance. L'en-tête `Retry-After:` indique au client si il peut réessayer à nouveau la requête.
  - …

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

Format des réponses

- Messages envoyés par le serveur



La section <message-headers> contient trois types d'en-têtes

En-têtes concernant le message lui-même et qui ne sont pas spécifiques à une requête particulière, identiques aux *general headers* de la requête.
Ex: date de la réponse

En-têtes contenant des informations additionnelles pour le client, ces informations dépendent du status code du message

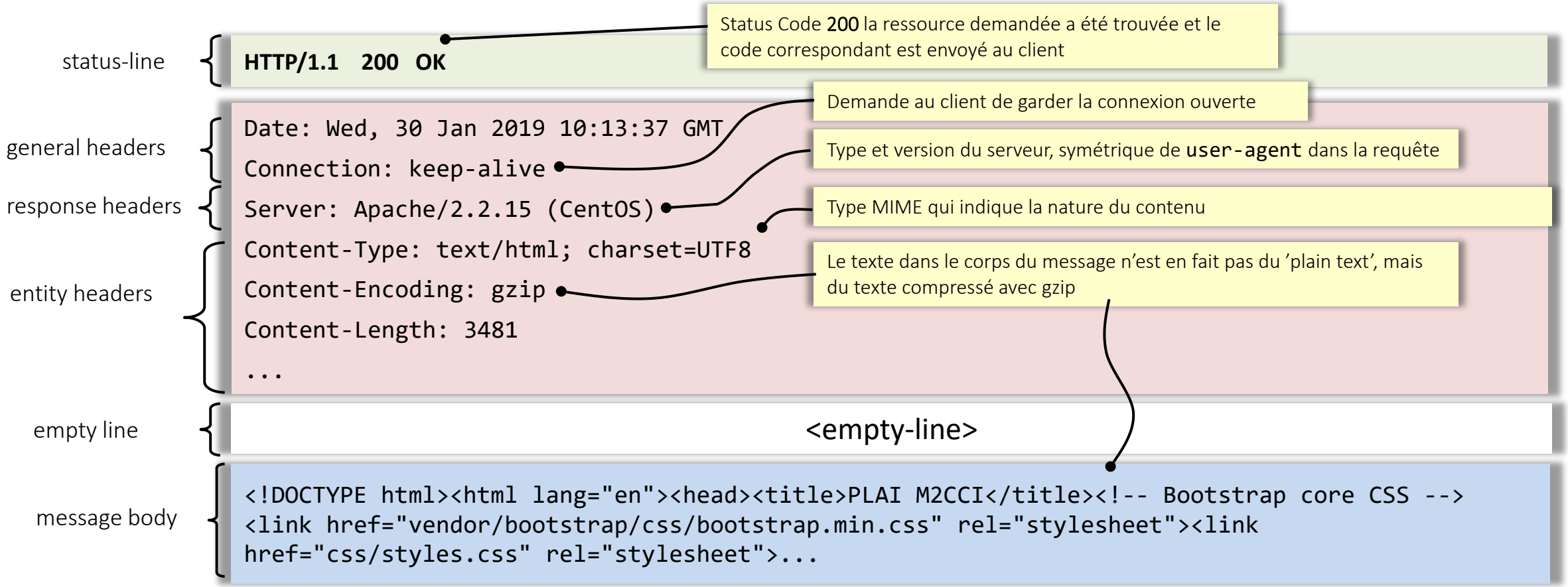
En-têtes décrivant le corps du message si il est présent
Ex: la taille du message

Contient soit la ressource demandée, soit un message d'information si une erreur s'est produite,

HTTP

Format des réponse

- Exemple d'une réponse
  - Réponse à la requête envoyée par un navigateur lorsque l'on demande la page d'accueil du cours PL2AI http://lig-membres.imag.fr/genoud/teaching/PL2AI/index.html



HTTP

Format des réponse

- Exemple d'une réponse lorsque la ressource demandée n'est pas trouvée
  - Reponse à la requête envoyée par un navigateur lorsque l'on demande la page d'accueil du cours PL2AI http://lig-membres.imag.fr/genoud/teaching/PL2AI/test.html

Status Code **404** la ressource demandée n'a pas été trouvée

status-line

```
HTTP/1.1   404   Not Found
```

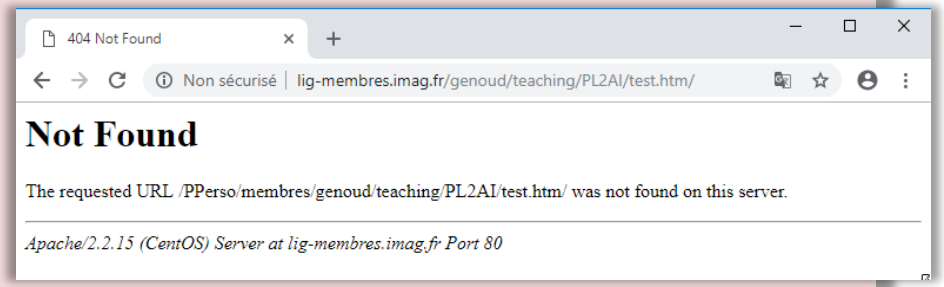
Demande au client de fermer la connexion

general headers

response headers

entity headers

```
Date: Wed, 30 Jan 2019 10:13:37 GMT
Connection: close
Server: Apache/2.2.15 (CentOS)
Content-Type: text/html; charset=UTF8
Content-Encoding: gzip
Content-Length: 265
...
```



Le corps du message contient un message d'erreur destiné à l'utilisateur

empty line

<empty-line>

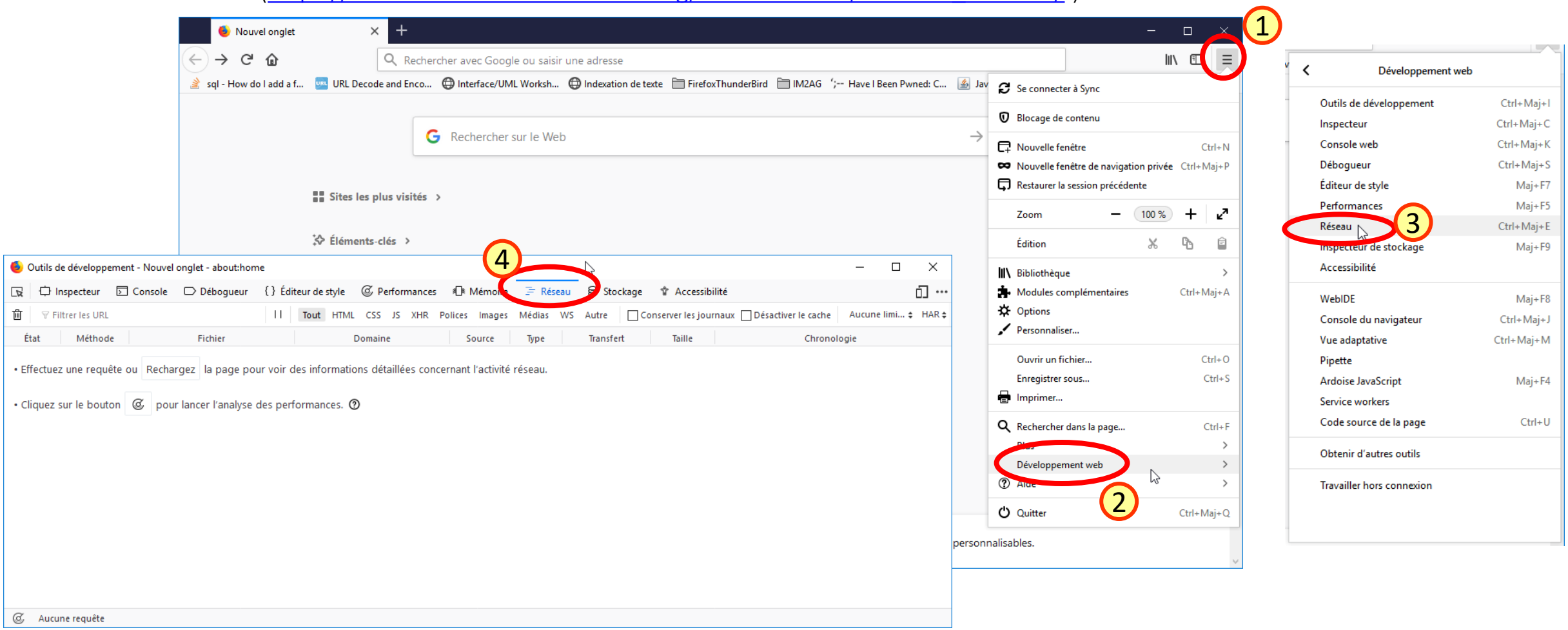
message body

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN"><html><head><title>404 Not Found</title>
</head><body><h1>Not Found</h1><p>The requested URL /PPerso/membres/genoud/teaching/PL2AI/
test.htm/ was not found on this server.</p><hr><address>Apache/2.2.15 (CentOS) Server at lig-
membres.imag.fr Port 80</address></body></html>
```

HTTP

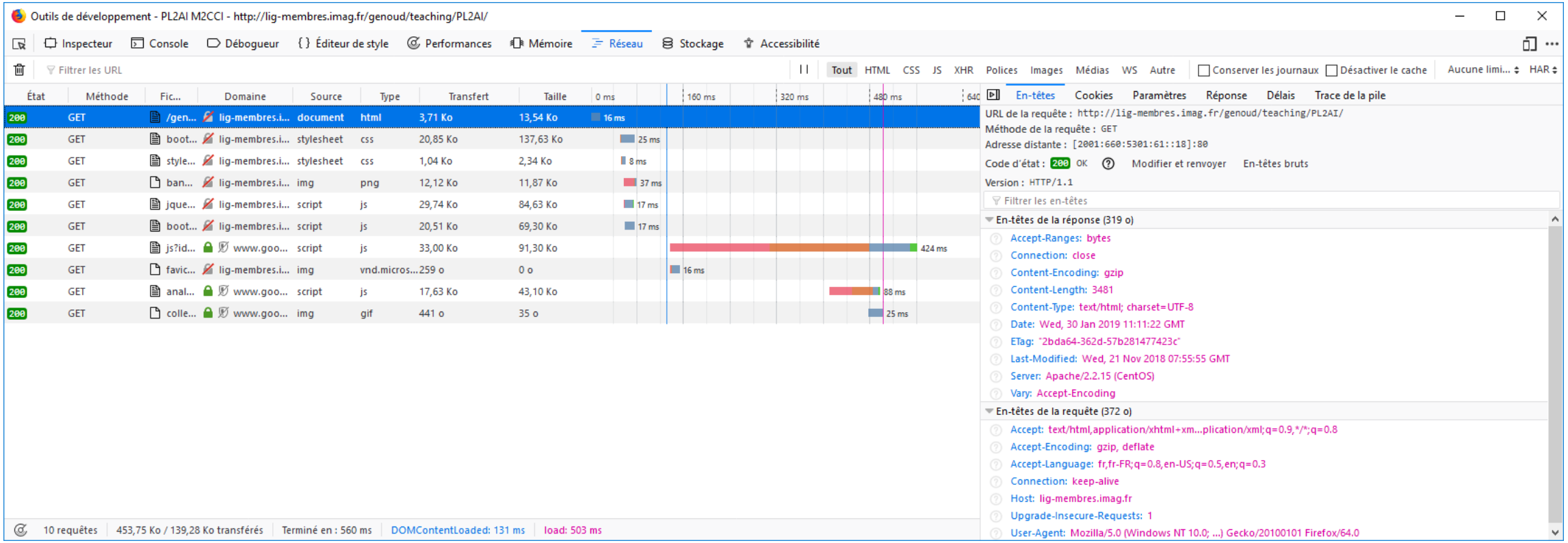
Outils de développement des navigateurs

- Les outils de développement des navigateurs permettent de suivre l'activité réseau et de tracer les requêtes HTTP
  - Ex: Firefox (https://firefox-source-docs.mozilla.org/devtools-user/network_monitor/)



Outils de développement des navigateurs

- Les outils de développement des navigateurs permettent de suivre l'activité réseau et de tracer les requêtes HTTP
  - Quand un page est chargée possibilité de voir le détail de toutes les requêtes effectuées



HTTP

- en-têtes contiennent un ensemble de valeurs présentées sous la forme

  `Nom: valeur` (cf. balises <meta> en HTML)

- 3 types d'en-têtes dans les requêtes
  - en-têtes généraux
    - utilisés à la fois par les clients et serveurs
    - informations générales : date, fait de maintenir ou non la connexion….
  - en-têtes de requête
    - communiquent au serveur des informations sur la configuration du client et sur le format de document désiré
  - en-têtes d'entité
    - utilisés pour les requêtes de type PUT ou POST
    - décrivent format des données envoyées au serveur

quelques exemples d'en-têtes

- `Connection: Close|Keep-Alive` (en tête général)
  - spécifie options désirées pour cette connexion
  - `Close` : la connexion est fermée après la réponse (par défaut avec HTTP/1.0)
  - `Keep-Alive` : crée une connexion persistante (par défaut avec HTTP/1.1).
    Sur serveurs Apache 2.2 et +, timeout de 5 secondes

- `User-Agent: chaîne` (en tête de requête)
  - informations sur le programme client (pour maintenir des statistiques ou permettre au serveur d'adapter la réponse selon le client)

- `Referer: url` (en tête de requête)
  - URL du document qui a donné un lien sur la ressource demandée (permet au serveur de tracer l'origine des demandes)

- `Accept: type/sous_type [q=valeur_de_qualité]` (en tête de requête)
  - liste les types MIME (Multipurpose Internet Mail Extension) de contenu acceptés par le client
  - * peut servir à spécifier tous les types / sous types
  - valeur de qualité : nombre de 0 (inacceptable) à 1 (acceptable)
  - ex : **`Accept: text/*, image/gif, image/tiff q=0`**

- `Accept-Languages: langue [q=valeur_de_qualité]` (en tête de requête)
  - indique quelles langues le client préfère
  - ex : **`Accept-Languages: en, fr`**

- `Accept-Charset:jeu-de-caractères[q=valeur_de_qualité]`
  (en tête de requête)
  - indique quels jeux de caractères le client préfère
  - ex : **`Accept-Charset: UTF-8, ISO-8859-1`**

- `If-Modified-Since: date` (en tête de requête)
  - indique que les données référencées par l'URL ne doivent être envoyées par le serveur que si le document a été modifié depuis la date indiquée.
  - permet de "cacher" des données au niveau client.
    - si le document n'a pas été modifié, le serveur retourne le code 304 et le client doit utiliser sa copie locale

- `Content-Type: type/sous-type` (en-tête d'entité)
  - Le format MIME du corps de la requête.

- `Content-Length: n` (en-tête d'entité)
  - Taille en octets du corps de la requête.

- `Content-Encoding: schema_d_encodage` (en-tête d'entité)
  - indication du schéma d'encodage (gzip, compress…) appliqué au corps de la réponse

HTTP

- en-têtes contiennent un ensemble de valeurs présentées sous la forme

  `Nom: valeur` (cf. balises <meta> en HTML)

- 3 types d'en-têtes dans les requêtes
  - en-têtes généraux
    - utilisés à la fois par les clients et serveurs
    - informations générales : date, fait de maintenir ou non la connexion….
  - en-têtes de réponse
    - communiquent au client des informations sur la configuration du serveur et sur l'URL demandée
  - en-têtes d'entité
    - décrivent format des données envoyées au client

quelques exemples d'en-têtes

- `Date: date` (en tête général)
  - Date et heure de la génération de la réponse.

- `Server: chaîne` (en tête de réponse)
  - Information sur le serveur sollicité (type, version ….)

- `Location : chaîne` (en tête de réponse)
  - Identifie l'URL exacte de la ressource demandée

- `Last-Modified: date` (en-tête de réponse)
  - Date et heure de la dernière modification du document.
- `Expires: date` (en-tête d'entité)
  - indique date et heure à laquelle le document peut changer ou les informations associées à la réponse peuvent devenir invalides
- `Content-Type: type/sous-type` (en-tête d'entité)
  - Le format MIME du corps de la réponse.
- `Content-Length: n` (en-tête d'entité)
  - Taille en octets du corps de la réponse.
- `Content-Encoding: schema_d_encodage` (en-tête d'entité)
  - indication du schéma d'encodage (gzip, compress…) appliqué au corps de la réponse

Mise en cache HTTP

- objectif : améliorer les performances des sites et applications web en réutilisant les ressources déjà collectées précédemment
- technique qui stocke une copie d'une ressource donnée et la renvoie quand elle est demandée.
  - quand un cache HTTP a une ressource demandée dans son espace de stockage, il intercepte la requête et renvoie sa copie au lieu de la re-télécharger depuis le serveur d'origine.
- avantages
  - améliore la performance en étant plus proche du client (moins de temps pour transmettre à nouveau la ressource)
  - réduit la charge du serveur qui n'a pas besoin de servir tous les clients lui-même

Mise en cache HTTP

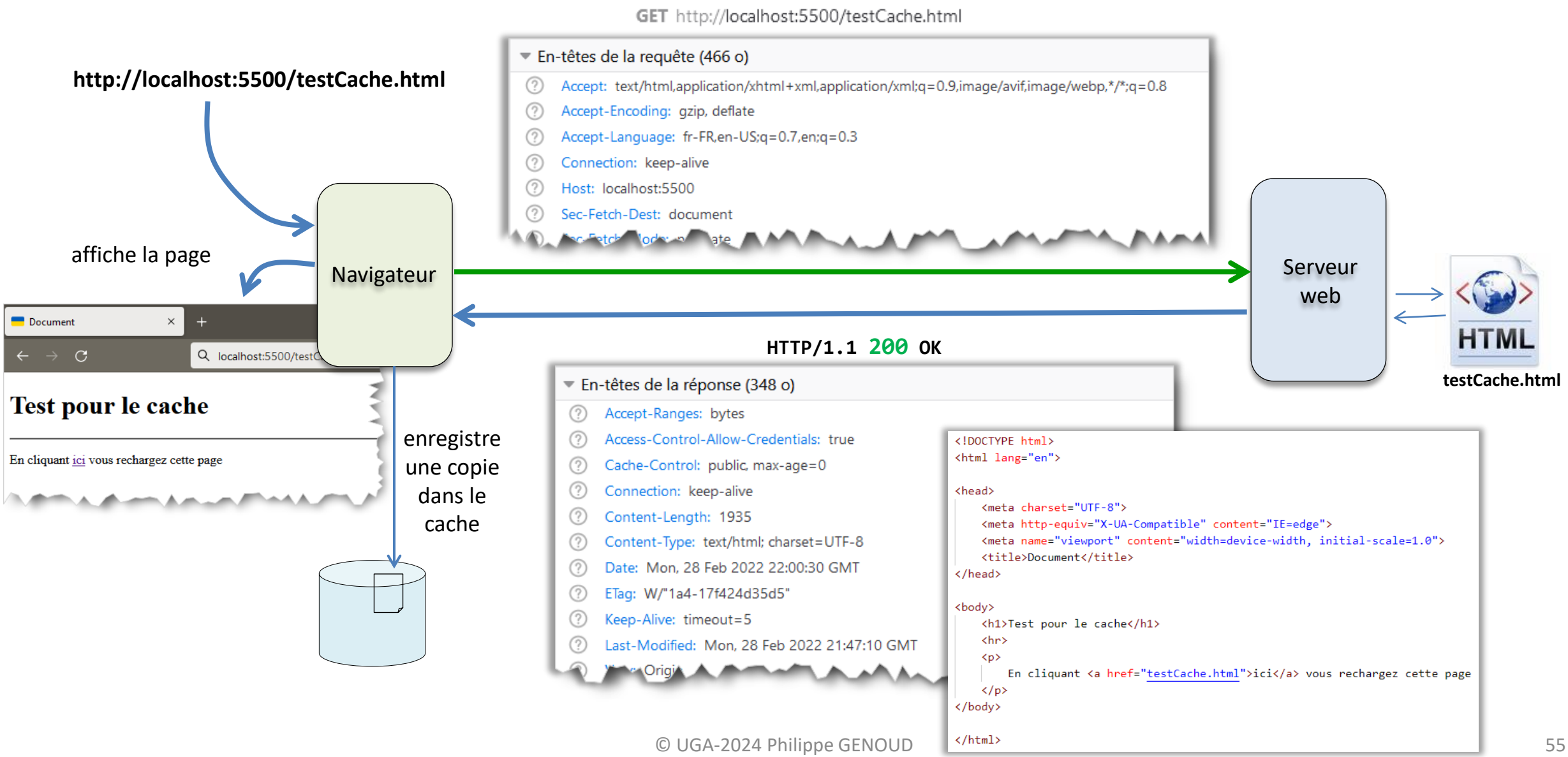
- différents types de caches
  - caches privés :
    - dédié à un seul utilisateur
    - le cache de votre navigateur
  - caches partagés
    - cache qui stocke les réponses pour qu'elles soient réutilisées par plus d'un utilisateur
    - cache de proxy, caches de passerelles, caches de CDN, répartiteurs de charge…

Mise en cache HTTP

- caches privés :
  - contient tous les documents téléchargés via HTTP par l'utilisateur
  - utilisé pour rendre les documents visités disponibles à la navigation via les boutons précédent / suivant, la sauvegarde, l'affichage du code source sans nécessiter un aller-retour supplémentaire vers le serveur
  - peut permettre la navigation hors-ligne de contenu en cache
- caches partagés
  - exemple : un proxy web au sein de son infrastructure de réseau local pour servir des utilisateurs multiples,
  - les ressources populaires sont réutilisées plusieurs fois, réduisant le trafic réseau et la latence.

pour en savoir plus : https://developer.mozilla.org/en-US/docs/Web/HTTP/Caching

Exemple de requêtes
utilisation du cache



Exemple de requêtes
utilisation du cache

on redemande la page
http://localhost:5500/testCache.html

affiche la page

Navigateur

Document

localhost:5500/testC

Test pour le cache

En cliquant ici vous rechargez cette page

recherche dans le cache

lit la copie dans le cache

GET http://localhost:5500/testCache.html

En-têtes de la requête (606 o)

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: fr-FR,en-US;q=0.7,en;q=0.3
Connection: keep-alive
Host: localhost:5500
If-Modified-Since: Mon, 28 Feb 2022 21:47:10 GMT
If-None-Match: W/"1a4-17f424d35d5"
Referer: *http://localhost:5500/testCache.html*

Serveur web

HTML
testCache.html

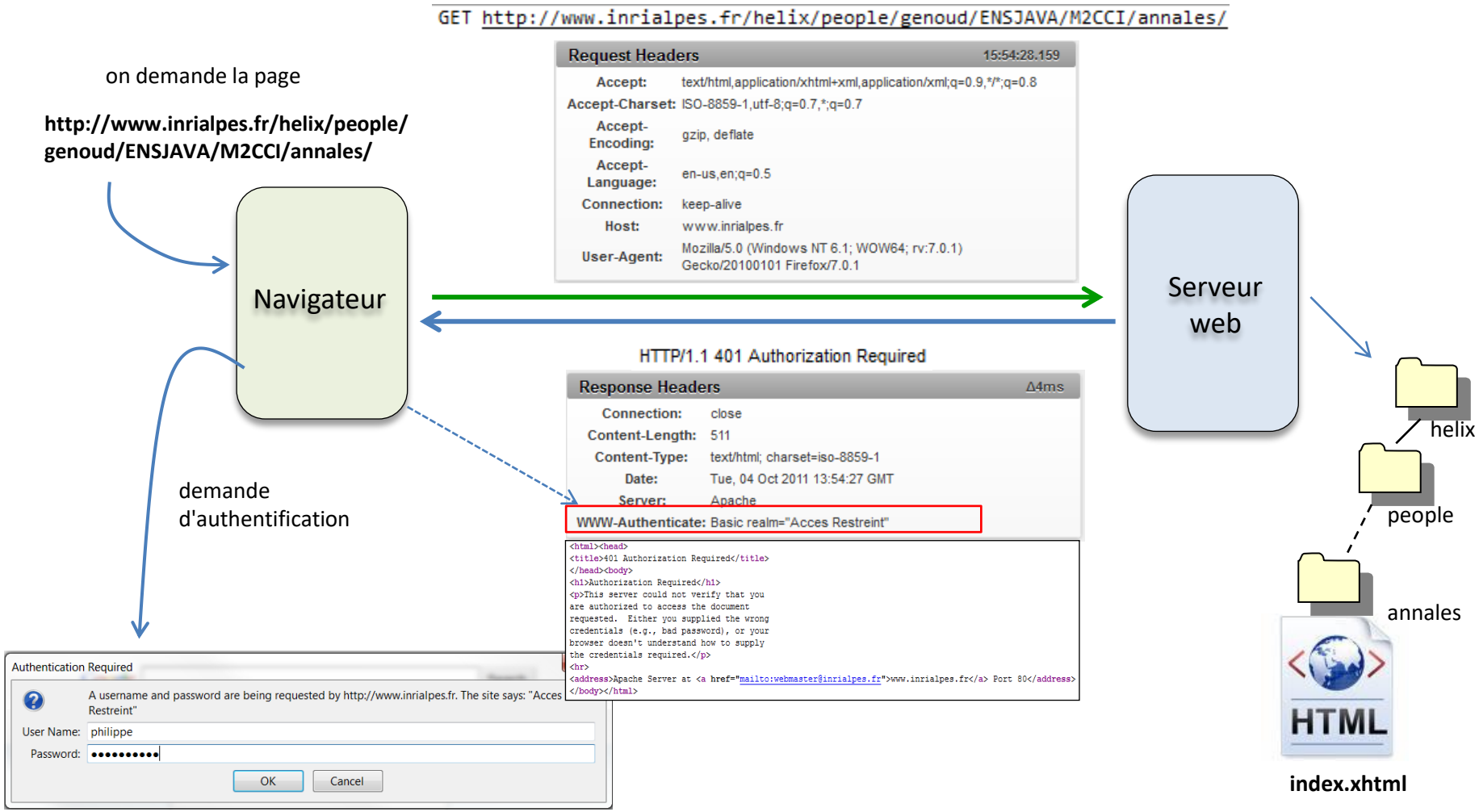
HTTP/1.1 304 Not Modified

En-têtes de la réponse (296 o)

Accept-Ranges: bytes
Access-Control-Allow-Credentials: true
Cache-Control: public, max-age=0
Connection: keep-alive
Date: Mon, 28 Feb 2022 22:13:22 GMT
ETag: W/"1a4-17f424d35d5"
Keep-Alive: timeout=5
Last-Modified: Mon, 28 Feb 2022 21:47:10 GMT

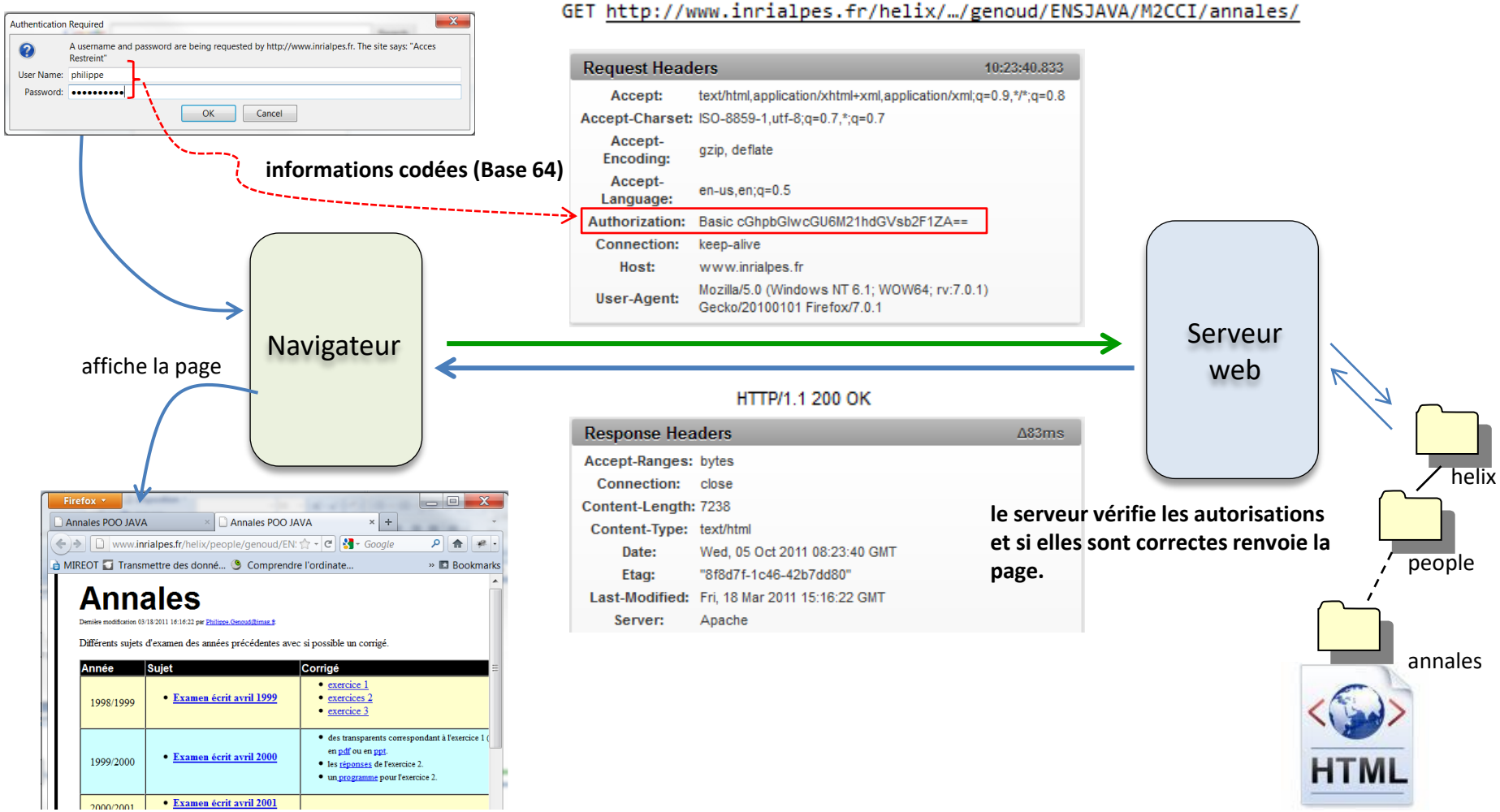
Exemple de requêtes
réponse d'erreur : 404 Not found

Exemple de requêtes
réponse d'erreur : 401 Unauthorized



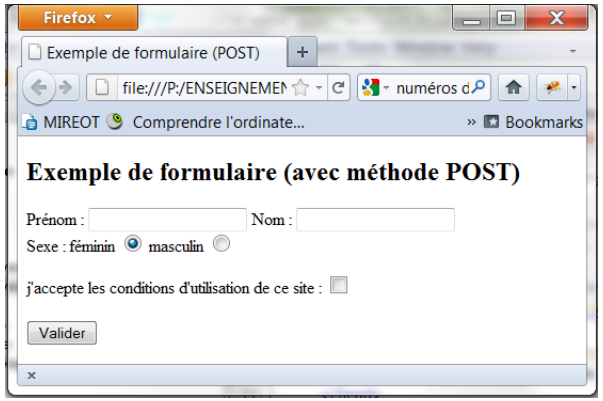
Exemple de requêtes
réponse d'erreur : 401 Unauthorized

le client renvoie à nouveau la requête en y incluant les informations d'autorisation



HTTP

Formulaires HTML : permettent de définir saisir des données et de les transmettre à un serveur Web.



Pour définir un formulaire :

```
<form action="xxx" method="yyy"> … </form>
```

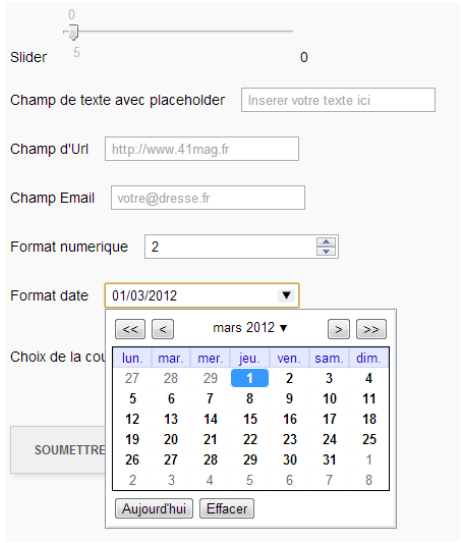
xxx = URL du programme chargé de récupérer et éventuellement de traiter les données

yyy = méthode de transmission des données : **GET** ou **POST**

HTML 4 propose un certain nombre de balises de base pour définir :

- des zones de saisie de texte `<input type="text">`
- des listes de choix `<input type="radio">`
- des cases à cocher `<input type="checkbox">`
- des boutons `<input type="submit">`
- des listes de sélection `<select>`

Attention au support selon les navigateurs



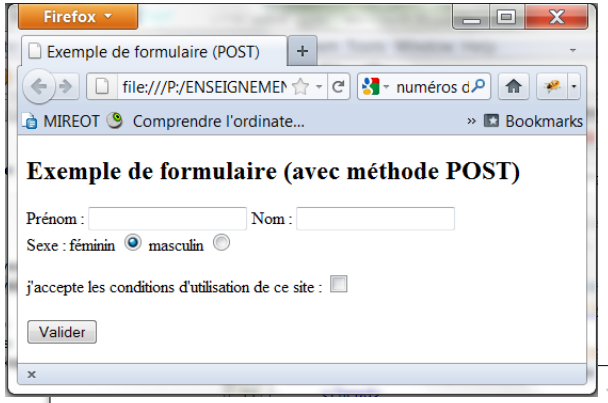
HTML 5 propose de nouveaux types:

- zone de saisie de date
- zone de saisie de couleur
- zone de saisie d'adresse mail
- sliders …

pour en savoir plus

https://developer.mozilla.org/en-US/docs/Learn/Forms/HTML5_input_types
https://www.w3schools.com/html/html_form_input_types.asp

formulaire HTML
exemple

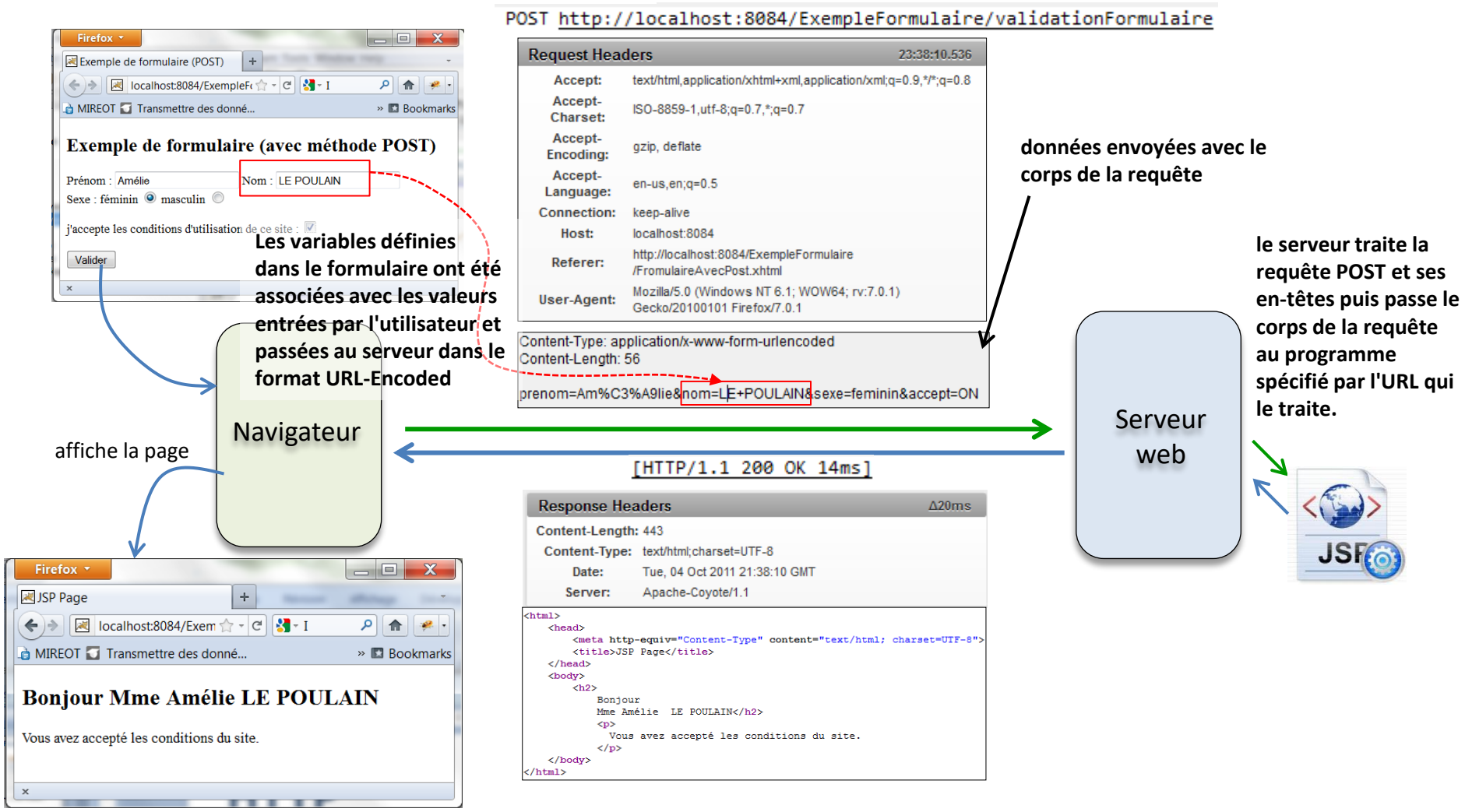


Formulaires HTML : permettent de saisir des données et de les transmettre à un serveur Web.

```
XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>Exemple de formulaire (POST)</title>
    </head>
    <body>
       <h2>
        Exemple de formulaire (avec méthode POST)
       </h2>
       <form action="validationFormulaire" method="POST">
           <p>
            Prénom : <input type="text" name="prenom" value="" size="20" />
            Nom    : <input type="text" name="nom" value="" size="20" /><br/>
            Sexe   : féminin <input type="radio" name="sexe" value="feminin" checked="checked" />
            masculin <input type="radio" name="sexe" value="masculin" /><br/>
           </p>
           j'accepte les conditions d'utilisation de ce site :
            <input type="checkbox" name="accept" value="ON" /><br/><br/>
            <input type="submit" value="Valider" />
       </form>
    </body>
</html>
```

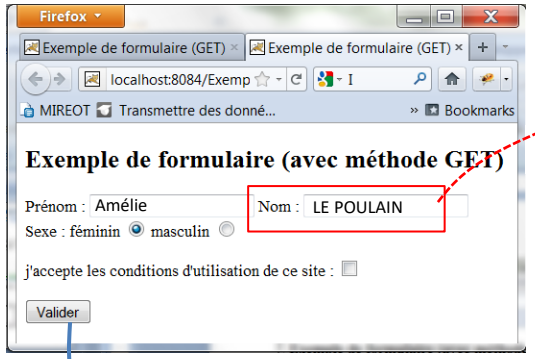
formulaire HTML
envoi de données au serveur (POST)

```
<form action="validationFormulaire" method="POST">
```



formulaire HTML
envoi de données au serveur (GET)

```
<form action="validationFormulaire" method="GET">
```



http://localhost:8084/ExempleFormulaire/validationFormulaire?prenom=Am%C3%A9lie&nom=LE+POULAIN&sexe=feminin&accept=ON

Les données sont envoyées dans l'URL

Request Headers	23:55:04.608
Accept:	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Charset:	ISO-8859-1,utf-8;q=0.7,*;q=0.7
Accept-Encoding:	gzip, deflate
Accept-Language:	en-us,en;q=0.5
Connection:	keep-alive
Host:	localhost:8084
Referer:	http://localhost:8084/ExempleFormulaire/FromulaireAvecGet.xhtml
User-Agent:	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:7.0.1) Gecko/20100101 Firefox/7.0.1

Navigateur

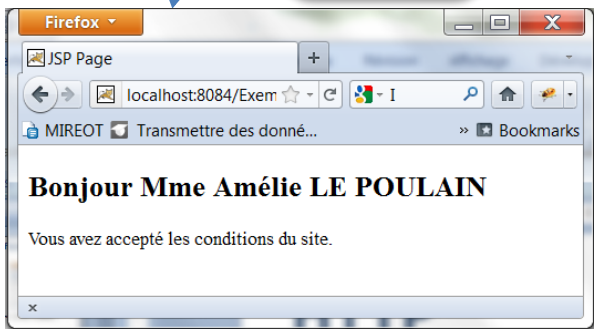
affiche la page

Serveur web

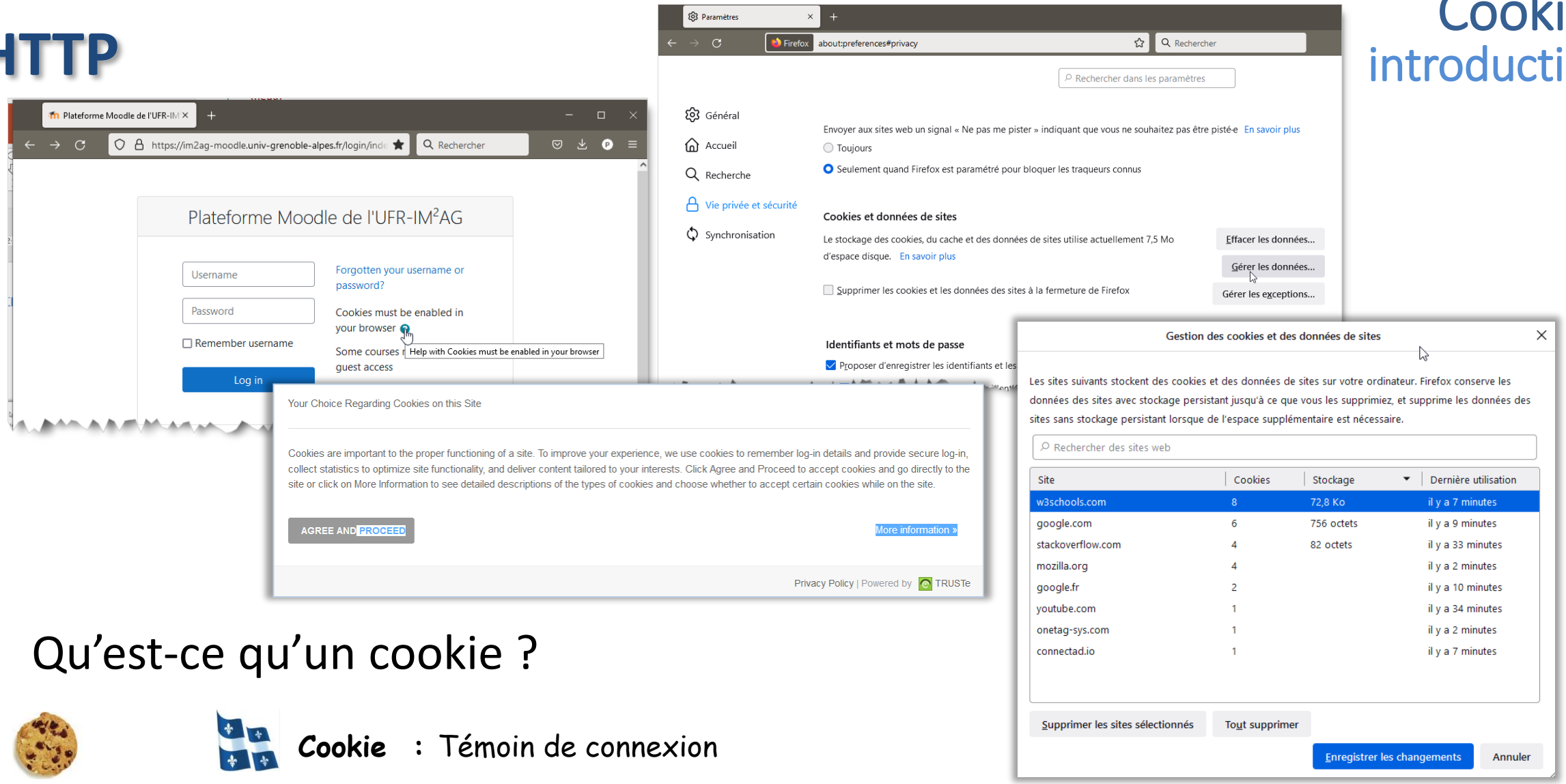
[HTTP/1.1 200 OK 14ms]

Response Headers	Δ21ms
Content-Length:	443
Content-Type:	text/html;charset=UTF-8
Date:	Tue, 04 Oct 2011 21:55:04 GMT
Server:	Apache-Coyote/1.1

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h2>
            Bonjour
            Mme Amélie  LE POULAIN</h2>
            <p>
              Vous avez accepté les conditions du site.
            </p>
    </body>
</html>
```



Cookies introduction



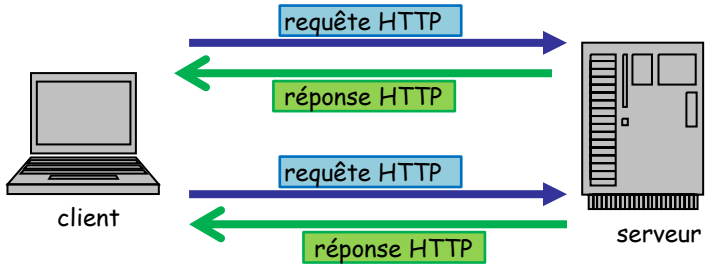
- Qu'est-ce qu'un cookie ?

Cookie : Témoin de connexion

- Un petit fichier texte déposé sur le disque dur du client
- Permet au serveur de reconnaître l'utilisateur lorsqu'il reviendra ensuite sur le site.

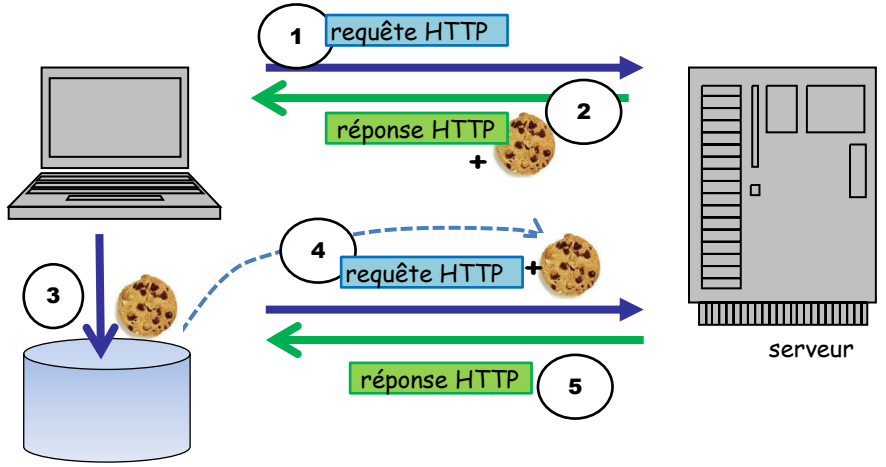
HTTP

- HTTP protocole sans mémoire (sans état *stateless*)
  - indépendance de chaque requête
  - pas de contrôle de l'ordre de navigation



le serveur n'a pas le moyen de relier deux transactions provenant d'un même client

- mécanisme de cookies
  - le serveur dépose de l'information chez le client
  - cette information est ensuite renvoyée au serveur lors des transactions suivantes



1. le client émet une requête HTTP
2. le serveur renvoie sa réponse à laquelle il associe un cookie
3. le client stocke localement le cookie
4. lors de requêtes suivante vers le serveur le client transmet le cookie avec la requête
5. le serveur peut adapter sa réponse en fonction du client et des traitements précédents

HTTP

- utilisations des cookies
  - gestion de session
    - serveur crée et envoie un identifiant de session unique
    - navigateur renvoie cet identifiant à chaque requête suivante

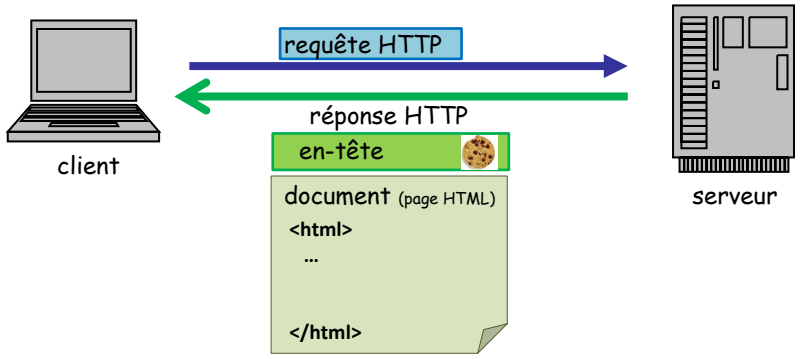
    → le serveur peut enregistrer des données (côté serveur) associées à cet identifiant
      - exemple : panier électronique
  - personnalisation
    - cookie permet de mémoriser l'information sur l'utilisateur d'un site

    → le serveur peut ensuite lui montrer un contenu approprié
  - pistage

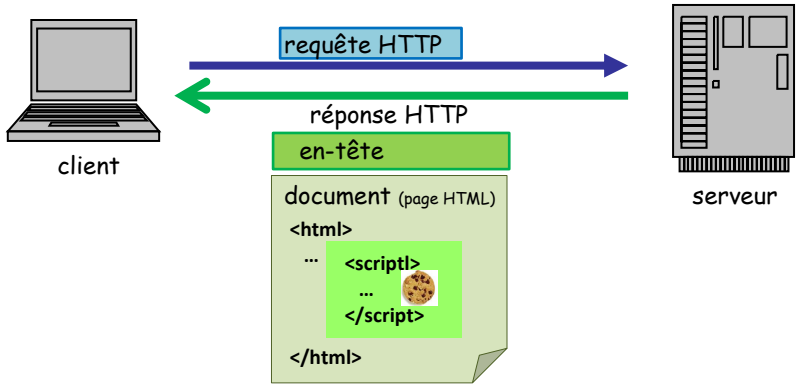
    → permet à un serveur de tracer les clients (usage statistique, choix des publicités à afficher…)
- inconvénients potentiels
  - lectures non désirées
    - d'un serveur pour lire les infos d'autres sites
    - d'une personne qui utiliserait votre ordinateur
  - renvoyés vers le serveur à chaque requête
    - augmentation du temps de chargement de la page

Cookies
dépôt de cookies

- 2 techniques de dépôt des cookies



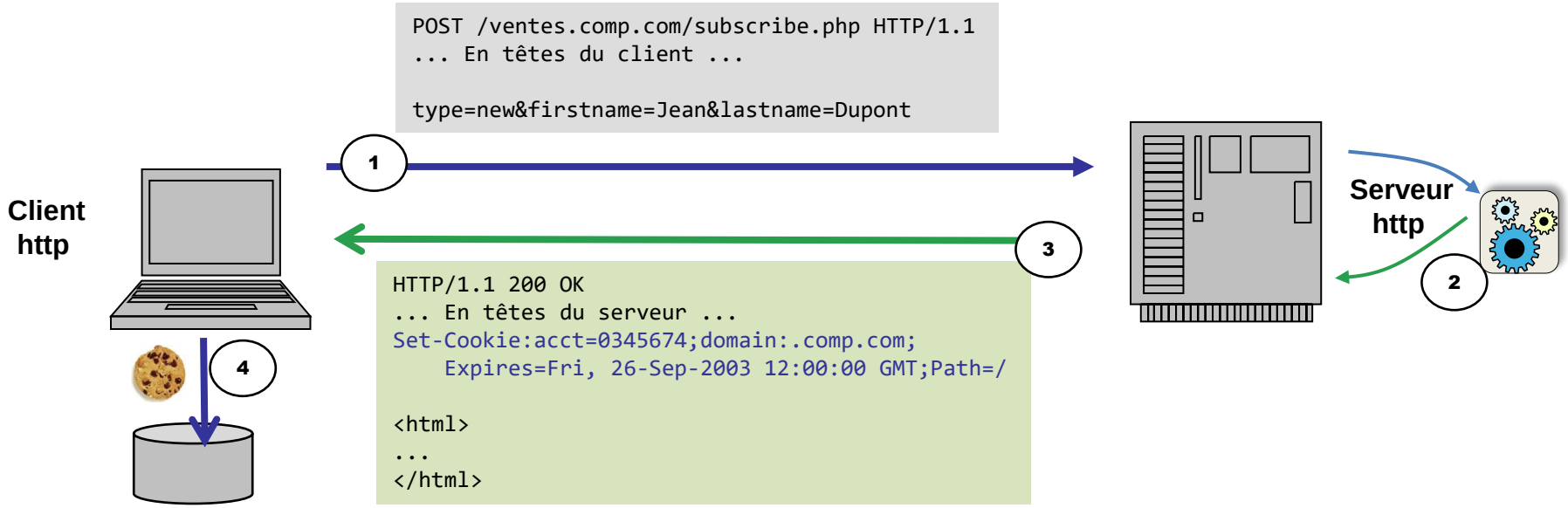
1ère méthode : la demande de création de cookie est insérée dans l'entête de la réponse HTTP



2ème méthode : les instructions de création de cookie (écrites dans un langage de programmation ex: javascript) sont encapsulées dans une page HTML

Cookies
dépôt de cookies

- 1ère **méthode** : la demande de création de cookie est insérée dans l'entête de la réponse HTTP



1. Le client émet une requête

2. Un programme du serveur traite cette requête et souhaite stocker des informations d'états chez le client

3. Le programme du serveur génère un en-tête **Set-Cookie** dans la réponse HTTP envoyée au client

4. Le programme client (navigateur) analyse la réponse et stocke le cookie dans un fichier sur le disque dur du client

Cookies
dépôt de cookies

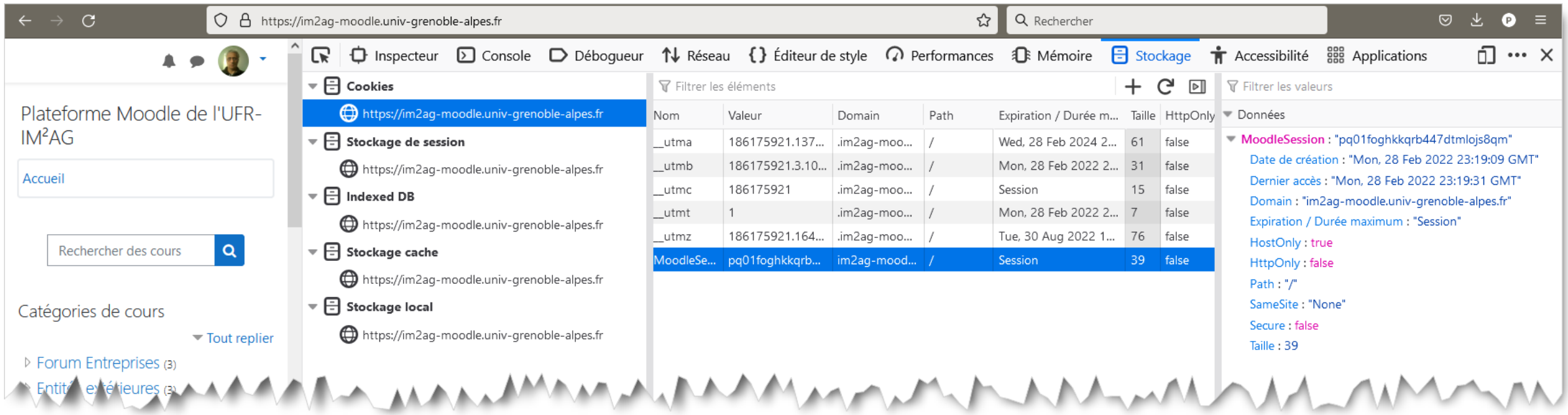
- 2ème **méthode :** les instructions de création de cookie (écrites en javascript) sont encapsulées dans une page HTML

1. Le client émet une requête
2. Le serveur traite cette requête
3. Il renvoie une page HTML, comportant un script réalisant un dépôt de cookie
4. Le script est exécuté au niveau du client et enregistre le cookie

Cookies
attributs des cookies

- attributs d'un cookie

- Nom du cookie
- Valeur
- Nom du serveur (domaine) qui l'a déposé
- Date d'expiration
- Protection
- Activé



Cookies
attributs des cookies

- positionnement des attributs dans l'en-tête de la réponse HTTP

Set-Cookie: *Nom*=*Valeur*; expires=*Date*; path=*Chemin*; domain=*NomDomaine*

- ***Nom*=*Valeur***
  - champ obligatoire : associe une valeur à une variable spécifique.
  - si il existe déjà un cookie sur le client avec le même nom sa valeur est modifiée
- **expires=**Date
  - date d'échéance du cookie

    → le cookie ne sera renvoyé au serveur que si la date courante < date expiration
  - si pas de date d'expiration le cookie n'est pas persistant, il sera supprimé à la fermeture du navigateur
  - le cookie peut être invalidé si sa date d'expiration est changée (par le serveur ou par un script) en une date du passé.

Cookies
attributs des cookies

- positionnement des attributs dans l'en-tête de la réponse HTTP

Set-Cookie: *Nom*=*Valeur*; expires=*Date*; path=*Chemin*; domain=*NomDomaine*

; secure ; httponly

- domain=NomDomaine
  - identification du serveur accédé correspondant au cookie.
- path=Chemin
  - association du cookie à un sous-ensemble de ressources
- secure
  - le cookie ne sera transmis par le client que si la connexion est sécurisée (HTTPS)
- httponly
  - le cookie n'est accessible que par le protocole HTTP (pas par scripts clients comme javascript)

requête **POST** https://im2ag-moodle.univ-grenoble-alpes.fr/login/index.php →

réponse ←

```
HTTP/1.1 303 See Other
Date: Mon, 28 Feb 2022 23:40:07 GMT
Server: Apache/2.4.38 (Debian)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: MoodleSession=2gh7st7tqfgjf177a84olan13p; path=/
Set-Cookie: MOODLEID1_=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; Max-Age=0; path=/
...
```

plusieurs directives **Set-Cookie** peuvent être insérées par le serveur dans une même réponse

Cookies
insertion dans une requête HTTP

- lorsque le client établit une requête pour accéder à une URL
  1. recherche parmi les cookies mémorisés ceux s'appliquant au serveur (attribut **domain**) et à l'URL (attribut **path**) et n'ayant pas expirés
     - le serveur appartient au même domaine que celui spécifié par l'attribut **domain**
     - si la ressource demandée dans l'url est située sous le chemin défini par **path**
  2. insertion dans l'en-tête de la requête d'une ligne avec les paires nom/valeur correspondantes

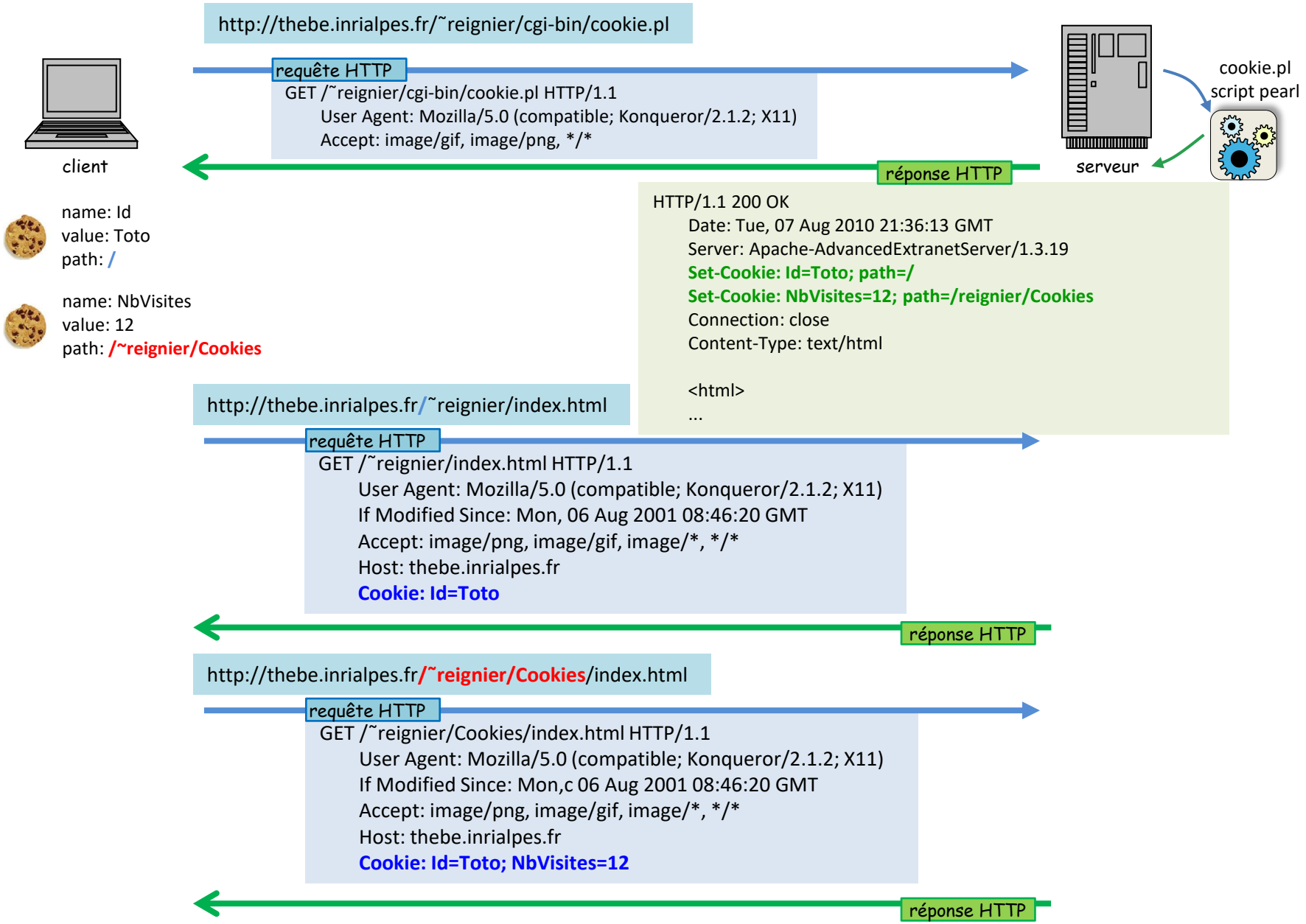
Cookie: *Nom1*=*Valeur1*; *Nom2*=*Valeur2*; ....

requête GET https://im2ag-moodle.univ-grenoble-alpes.fr/course/index.php?categoryid=33 →

réponse ←

```
GET /course/index.php?categoryid=33 HTTP/1.1
Host: im2ag-moodle.univ-grenoble-alpes.fr
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: fr-FR,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: https://im2ag-moodle.univ-grenoble-alpes.fr/
Cookie: MoodleSession=2gh7st7tqfgjf177a84olan13p; __utma=186175921.1620508457.1646091358.1646091358.1646091358.1;
__utmb=186175921.5.10.1646091358; __utmc=186175921;
__utmz=186175921.1646091358.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none)
…
```

Cookies
exemples



Le Monde et des tiers selectionnés, notamment des partenaires publicitaires, utilisent des cookies ou des technologies similaires. Les cookies nous permettent d'accéder à, d'analyser et de stocker des informations telles que les caractéristiques de votre terminal ainsi que certaines données personnelles (par exemple : adresses IP, données de navigation, d'utilisation ou de géolocalisation, identifiants uniques).

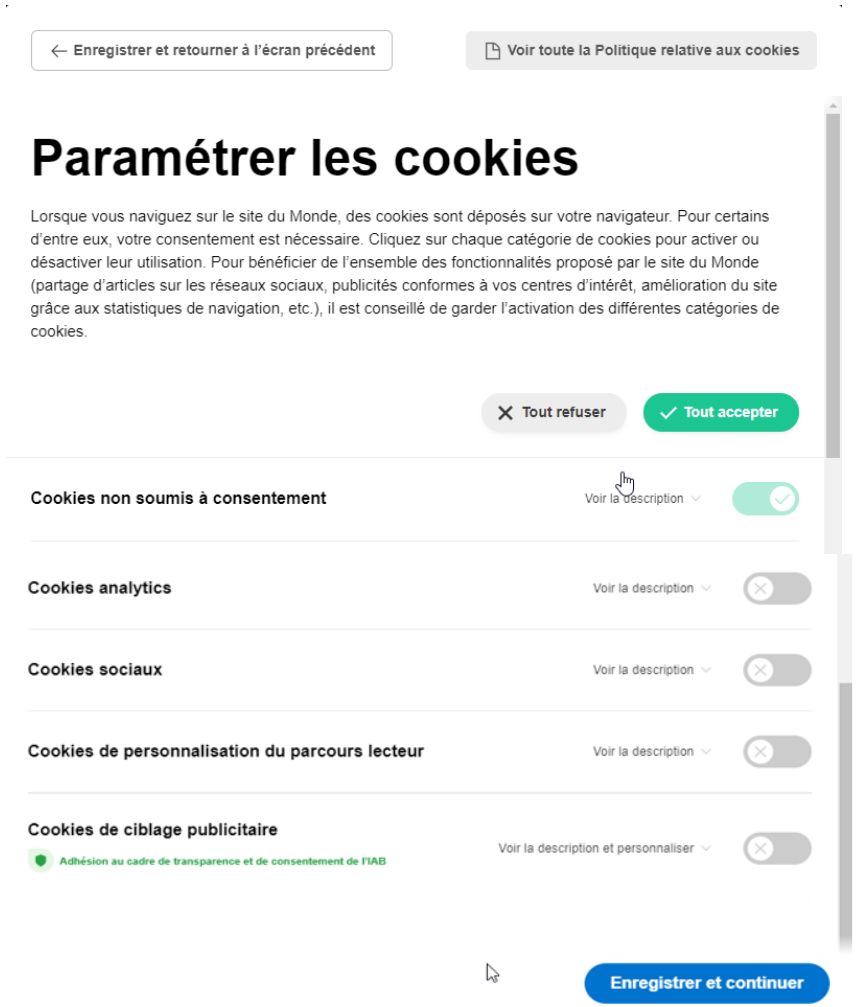
Ces données sont traitées aux fins suivantes : analyse et amélioration de l'expérience utilisateur et/ou de notre offre de contenus, produits et services, mesure et analyse d'audience, interaction avec les réseaux sociaux, affichage de publicités et contenus personnalisés, mesure de performance et d'attractivité des publicités et du contenu.

Pour plus d'information, consulter notre politique de confidentialité. Vous pouvez librement donner, refuser ou retirer votre consentement à tout moment en accédant à notre outil de paramétrage des cookies et/ou, en ce qui concerne la publicité, au panneau des préférences publicitaires. Si vous ne consentez pas à l'utilisation de ces technologies, nous considérerons que vous vous opposez également à tout dépôt de cookie fondé sur un intérêt légitime.

Vous pouvez consentir à l'utilisation de ces technologies en cliquant sur « accepter »

Accepter

Paramétrer les cookies



Alternatives au cookies

- des solutions "propiétaires"
  - Flash (Adobe) : Flash Local Storage Objects
  - Gears (Google), utilise une base de données SQL locale

  … mais basées sur des plugins additionnels

  →**solution standard intégrée à HTLM 5.**

  API (javascript) **Web Storage** pour la persistance de données côté client

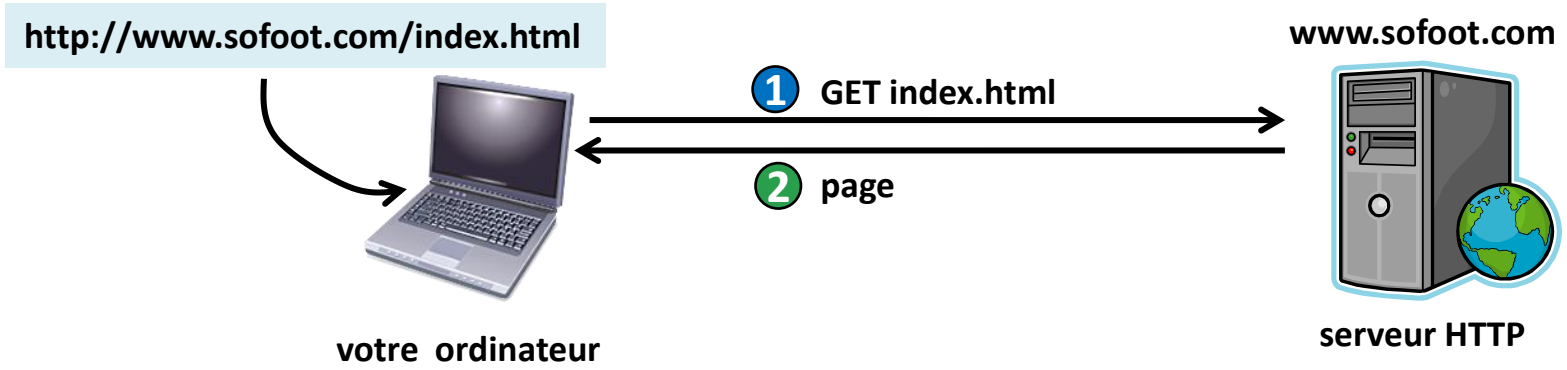
  http://www.w3schools.com/html/html5_webstorage.asp

  http://www.alsacreations.com/article/lire/1402-web-storage-localstorage-sessionstorage.html

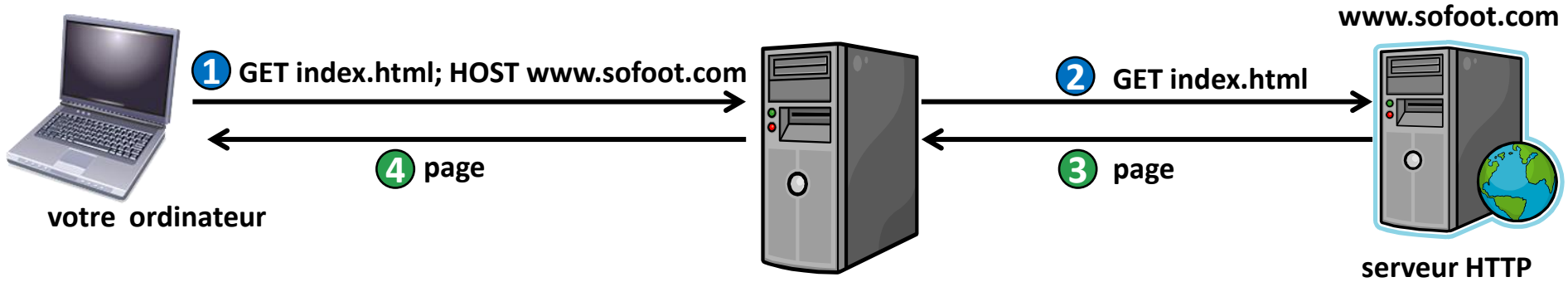
  → applications *offline* : web déconnecté

HTTP

- Requête HTTP sans proxy
  - votre ordinateur se connecte au serveur HTTP et lui demande la page



- Requête HTTP avec proxy
  - votre ordinateur se connecte au proxy et lui demande d'aller chercher la page sur le serveur HTTP



d'après : http://sebsauvage.net/comprendre/proxy/index.html

HTTP

- intérêts d'un serveur proxy
  - accélération de la navigation (proxy-cache)
    - mise en cache des pages les plus demandées
  - sécurité du réseau local
    - autorise votre ordinateur à se connecter à l'extérieur mais interdit aux ordinateurs d'internet de se connecter sur le votre.
  - filtrage
    - interdit l'accès à certains sites
  - anonymat (proxy-anonyme)
    - masque les informations concernant votre ordinateur (adresse IP, navigateur….)

HTTP

- les risques
  - confidentialité
    - le proxy peut connaître toutes les pages visitées
    - le proxy peut intercepter vos éventuels mots de passes (à moins d'utiliser HTTPS/SSL)
  - modifications
    - le proxy peut modifier à la volée les pages qu'il vous fourni
  - censure
    - interdiction de l'accès à certains sites
  - proxy transparents
    - détournement de vos requêtes vers un proxy sans configuration par l'utilisateur

références

- https://developer.mozilla.org/fr/docs/Web/HTTP/Overview
- https://www.pierre-giraud.com/http-reseau-securite-cours/evolution-fonctionnalite-performance/
- https://www.youtube.com/watch?v=iYM2zFP3Zn0&t=286s

- https://www.youtube.com/watch?v=Q-BpqyOT3a8