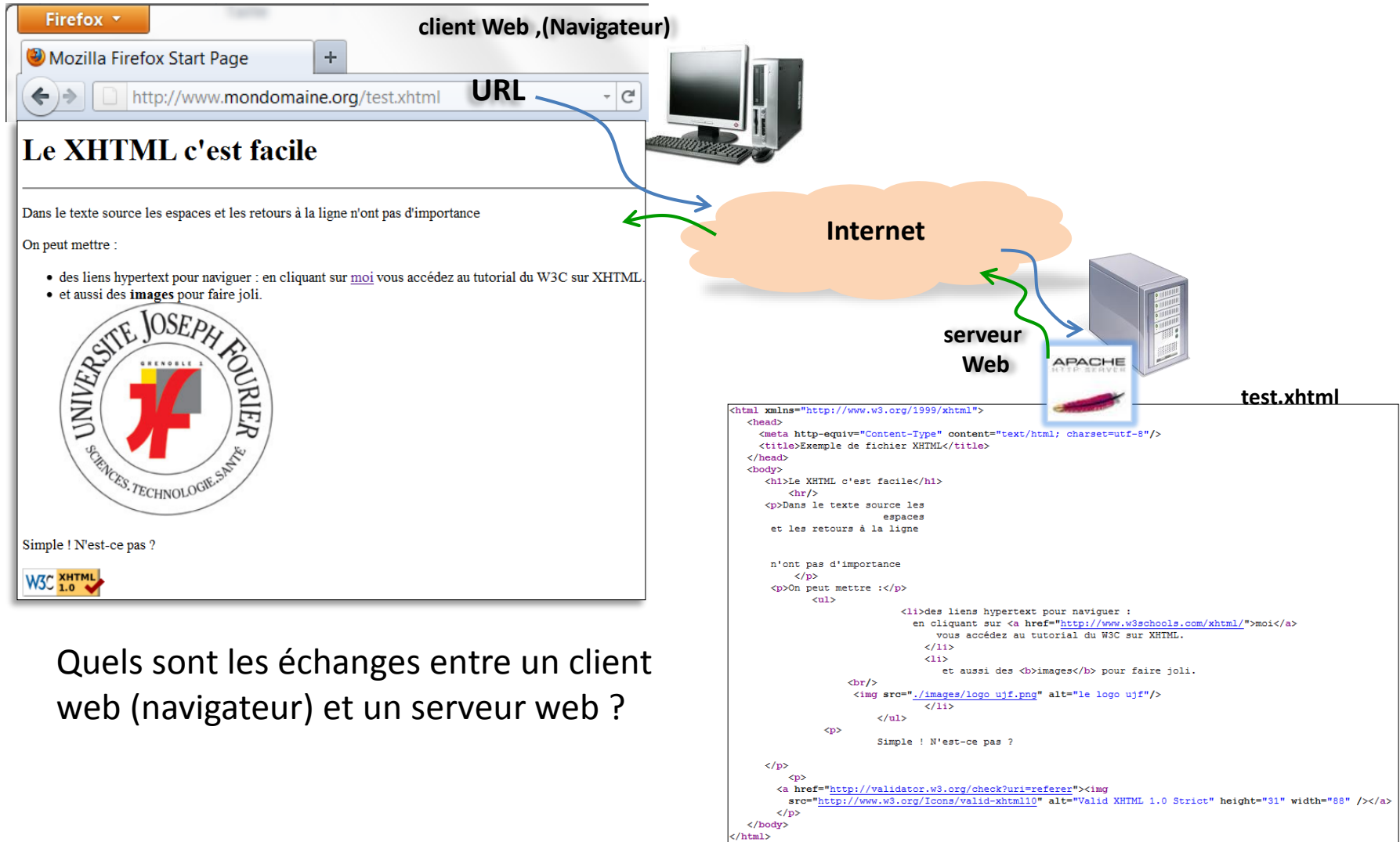


HTTP

Hyper Text Transfert Protocol

HTTP



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

Le protocole HTTP

- Protocole
 - langage spécifiant comment deux programmes distants communiquent entre eux
 - Comment les clients demandent les données
 - Comment les serveurs répondent à ces requêtes
- **HTTP (Hyper Text Transfert Protocol)**
 - Protocole utilisé pour échanges entre clients et serveurs web
- autres exemples de protocoles
 - SMTP : Simple Mail Transfer Protocol
 - FTP : File Transfert Protocol

Comprendre HTTP

- Comprendre les interactions entre clients web (navigateurs, robots, moteurs de recherche...) et les serveurs web
- Interroger manuellement des serveurs web
 - Recevoir informations de bas niveau cachées par navigateurs
 - Mieux comprendre la configuration et capacités 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 d'application web dynamiques

HTTP

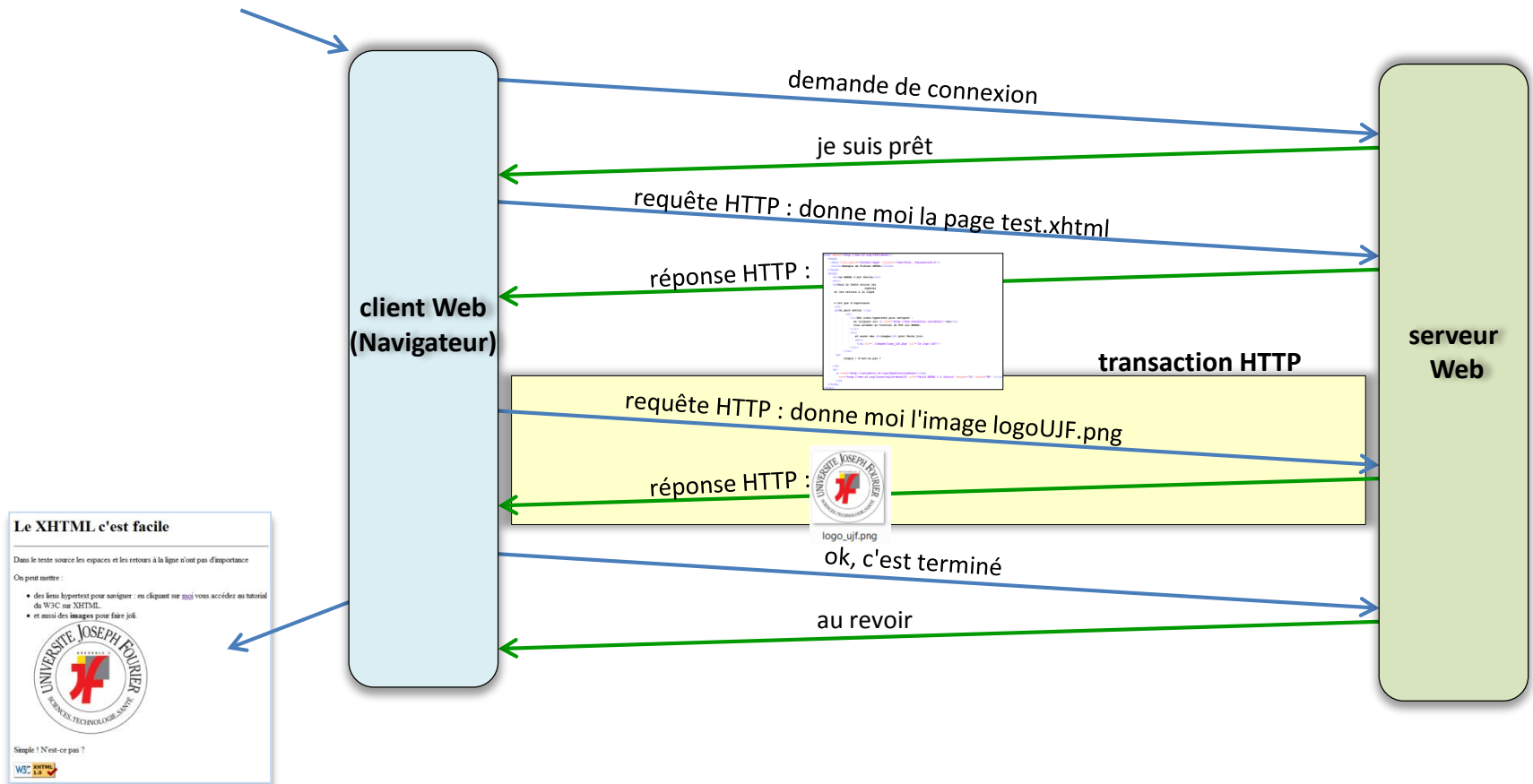
Architecture Client/Serveur

- Déroulement du chargement d'une page

URL (uniform Resource Locator)

`http://www.mondomaine.org/test.xhtml`

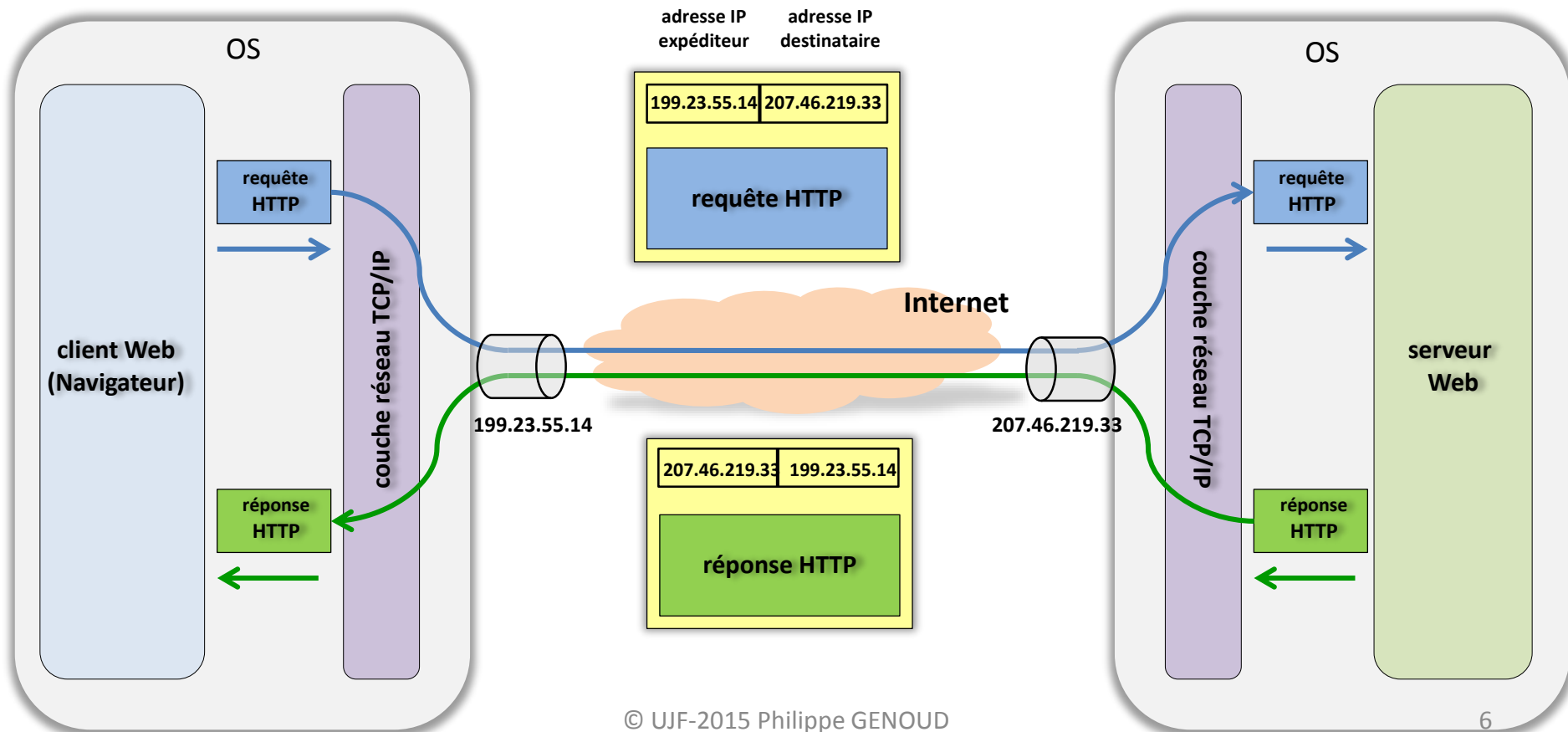
transaction HTTP : requête et réponse
HTTP vont toujours de paire



HTTP

protocole HTTP transport sur TCP/IP

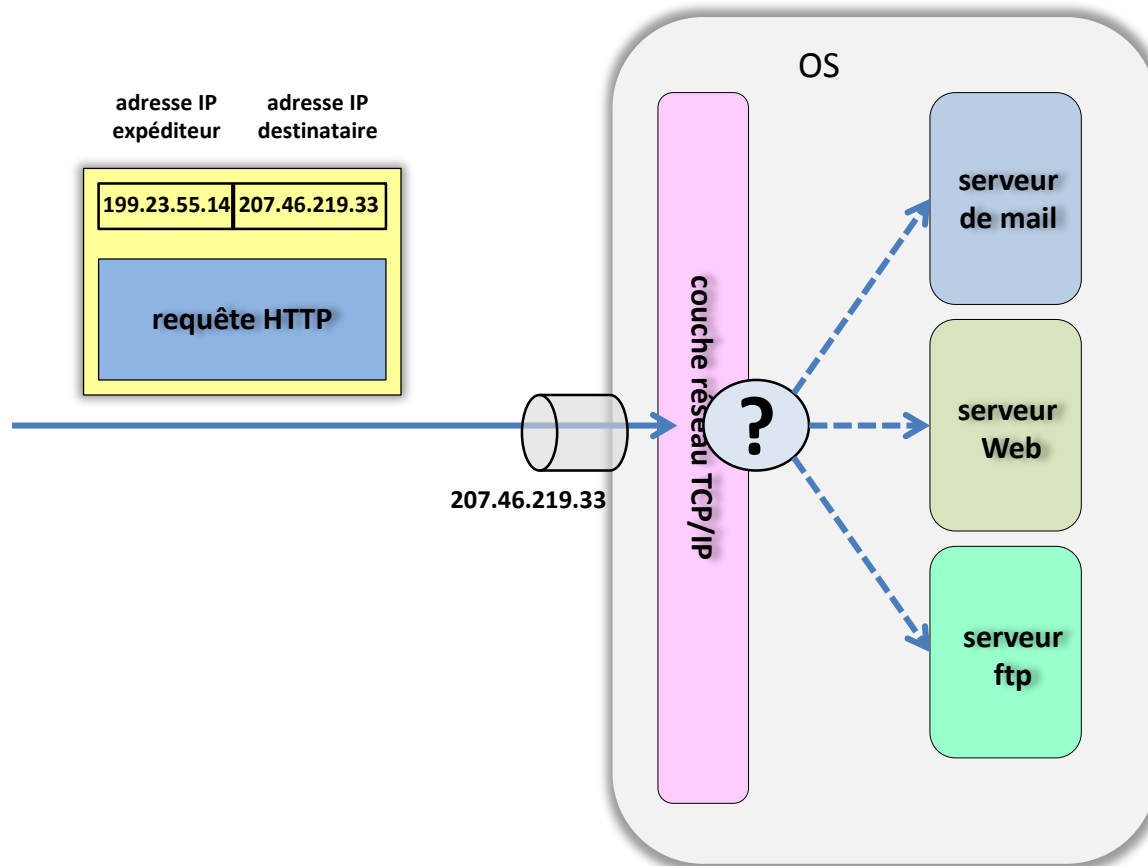
- requête et réponse HTTP passent par la couche de transport TCP/IP
 - rajoute infos nécessaires au routage
 - découpage en paquets
 - reconstitution des paquets



HTTP

transport sur TCP/IP ports

- 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 ?



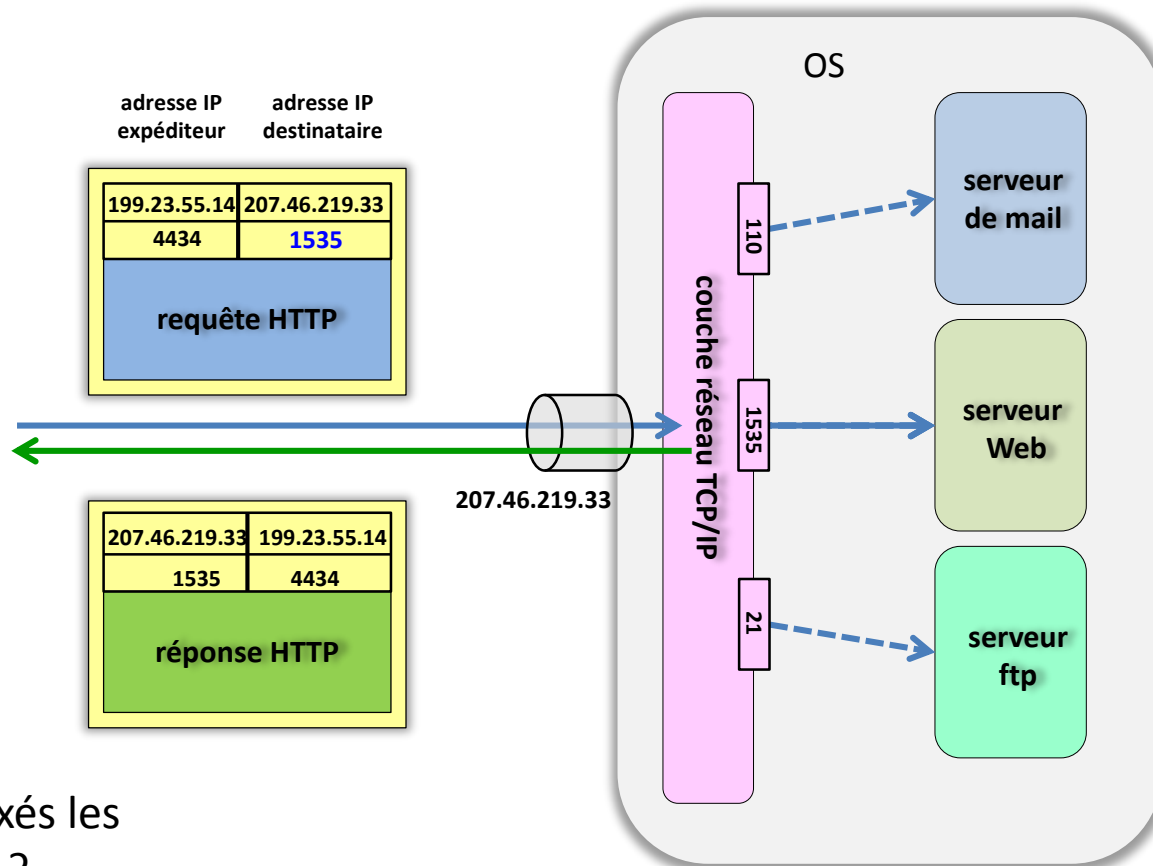
HTTP

transport sur TCP/IP ports

- **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

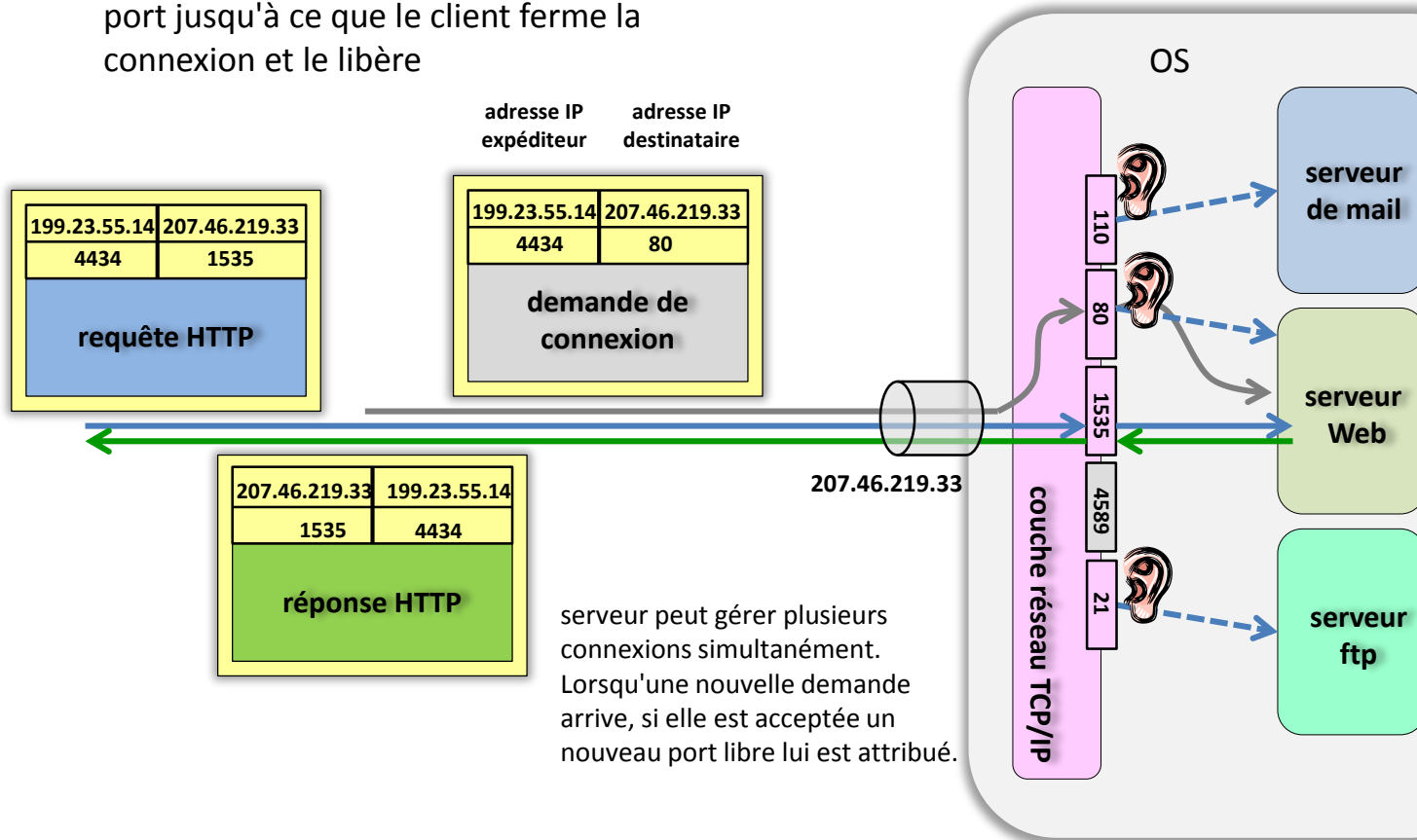


Comment sont fixés les
numéros de port ?

HTTP

transport sur TCP/IP ports

- un port prédéterminé (port d'écoute) est attribué au serveur pour les demandes de connexion
- à la demande de connexion
 - un numéro de port libre est attribué au client
 - si le serveur accepte la connexion un port libre est sélectionné et attribué à celle-ci
- 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**

110 serveur POP
21 server ftp

....

pour connaître les numéros de port attribués :
<http://www.docmemo.com/internet/ports.php>

Le numéro du port d'écoute peut être modifié côté serveur et explicité dans l'url de connexion

HTTP

URL forme générale

- **URL (Uniform Resource Locator)** → localisation d'un document
- Syntaxe:

`méthode://nomserveur[:port][/répertoires/fichier[?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,
- port: le numéro du port d'écoute (port par défaut si non spécifié)
- répertoires le chemin pour accéder au document
- fichier nom du document.

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

Utiliser le protocole HTTP

Contacteur sur le réseau l'ordinateur dont le nom d'hôte est www.mamachine.org

Se connecter sur le port IP n° 8080 de cet ordinateur

Le chemin du document. Ici fichier `test.html` dans répertoire `/people/genoud/`

une position dans le document

HTTP

URL exemples

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

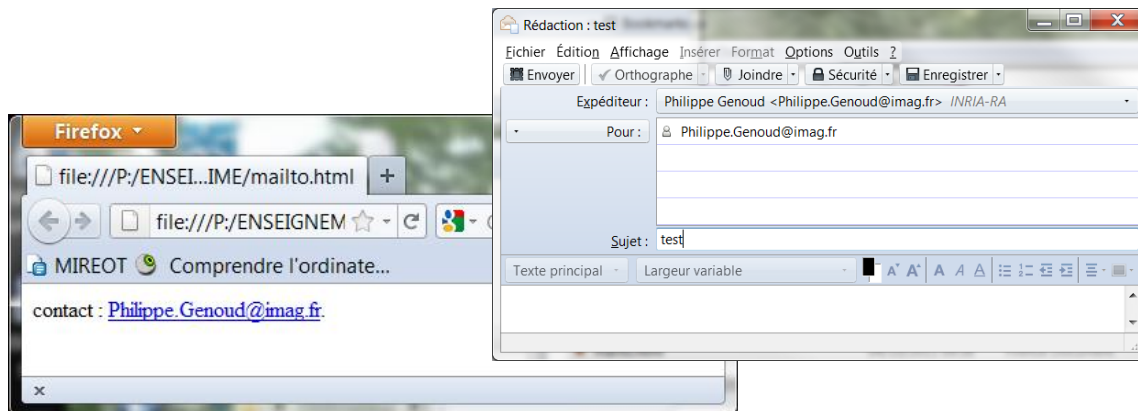
http://www.google.com

google.com

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

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

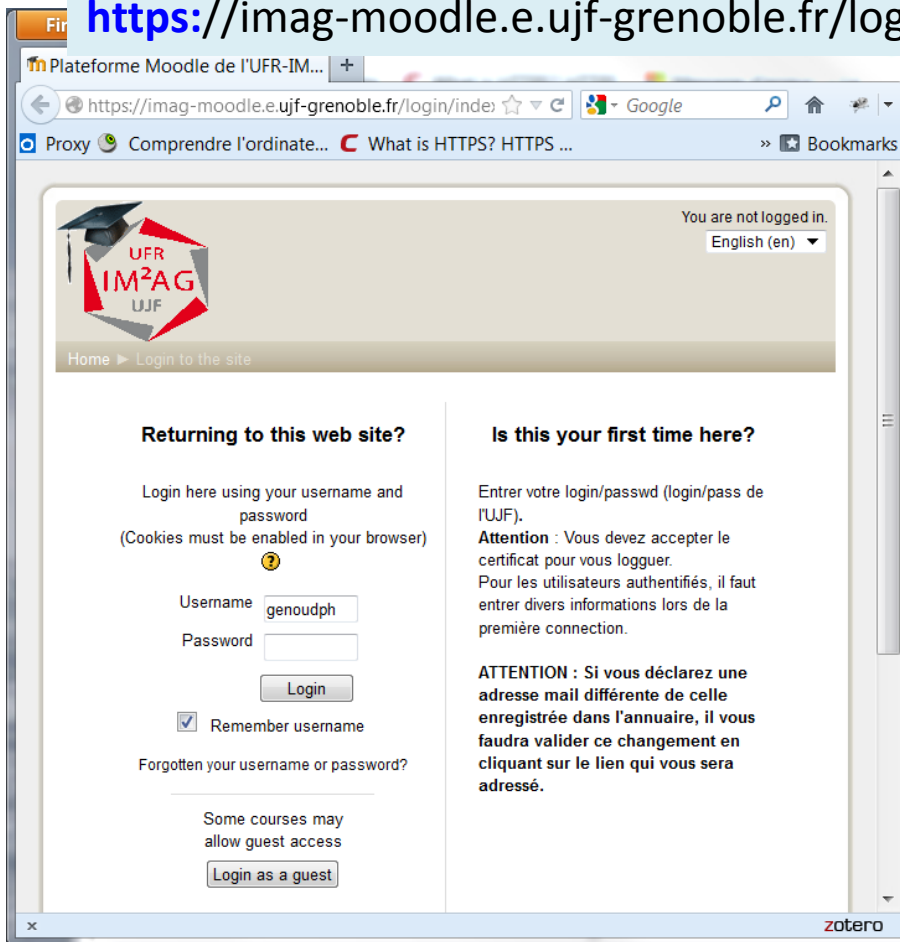
mailto:Philippe.Genoud@imag.fr



contact : `Philippe.Genoud@imag.fr`.

HTTPS

<https://imag-moodle.e.ujf-grenoble.fr/login/index.php>



C'est quoi SSL, SSH, HTTPS ?

<http://sebsauvage.net/comprendre/ssl/>

What is HTTPS

<http://www.youtube.com/watch?v=JCvPnwpWVUQ> © UJF-2015 Philippe GENOUD

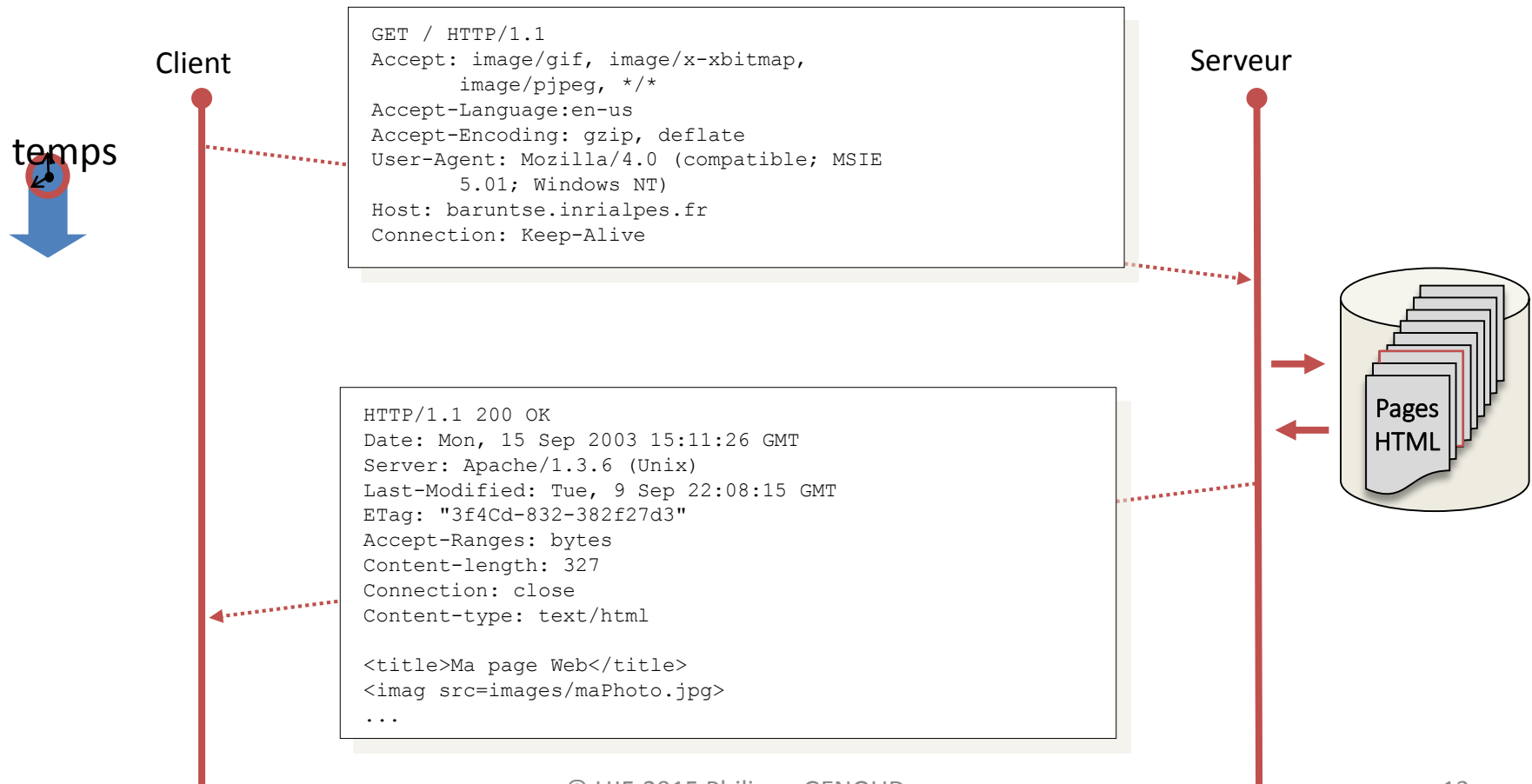
Hyper Text Transfert Protocol Secure

- combinaison de HTTP avec un couche de chiffrement SSL (Secure Socket Layers) ou TLS (Transport Layer Security).
 - 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
- port par défaut serveurs HTTPS : 443

HTTP

transaction HTTP

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



GET / HTTP/1.1

Accept: image/gif, image/x-xbitmap,
image/pjpeg, */*

Accept-Language: en-us

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE
5.01; Windows NT)

Host: baruntse.inrialpes.fr

Connection: Keep-Alive

Demande un document :

- / sa localisation sur le serveur
- version protocole HTTP utilisée par client

Indique au serveur les types de documents acceptés par le client

Langue préférée du client

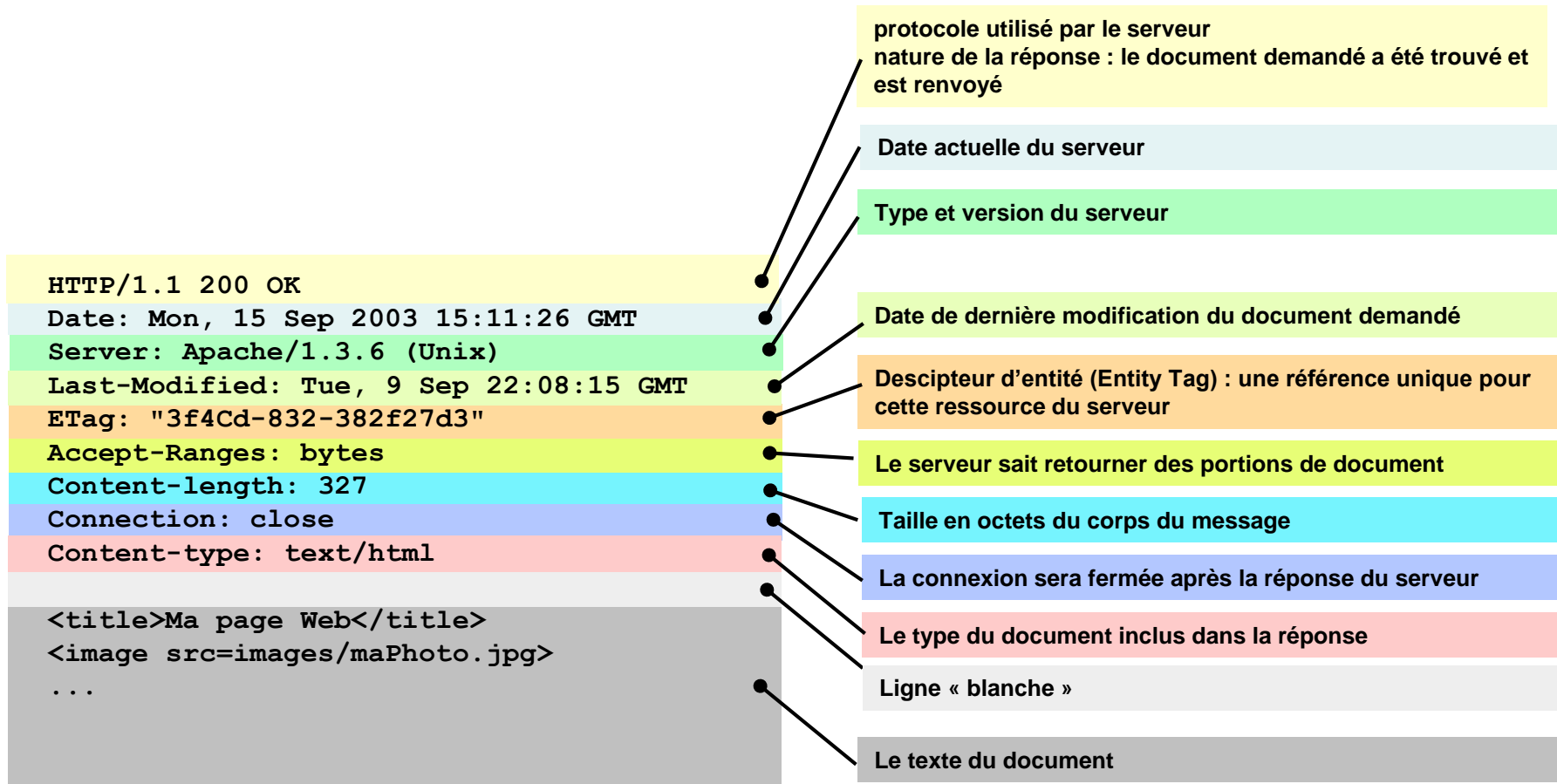
- peut être utilisé par le serveur si il possède le document en plusieurs langues

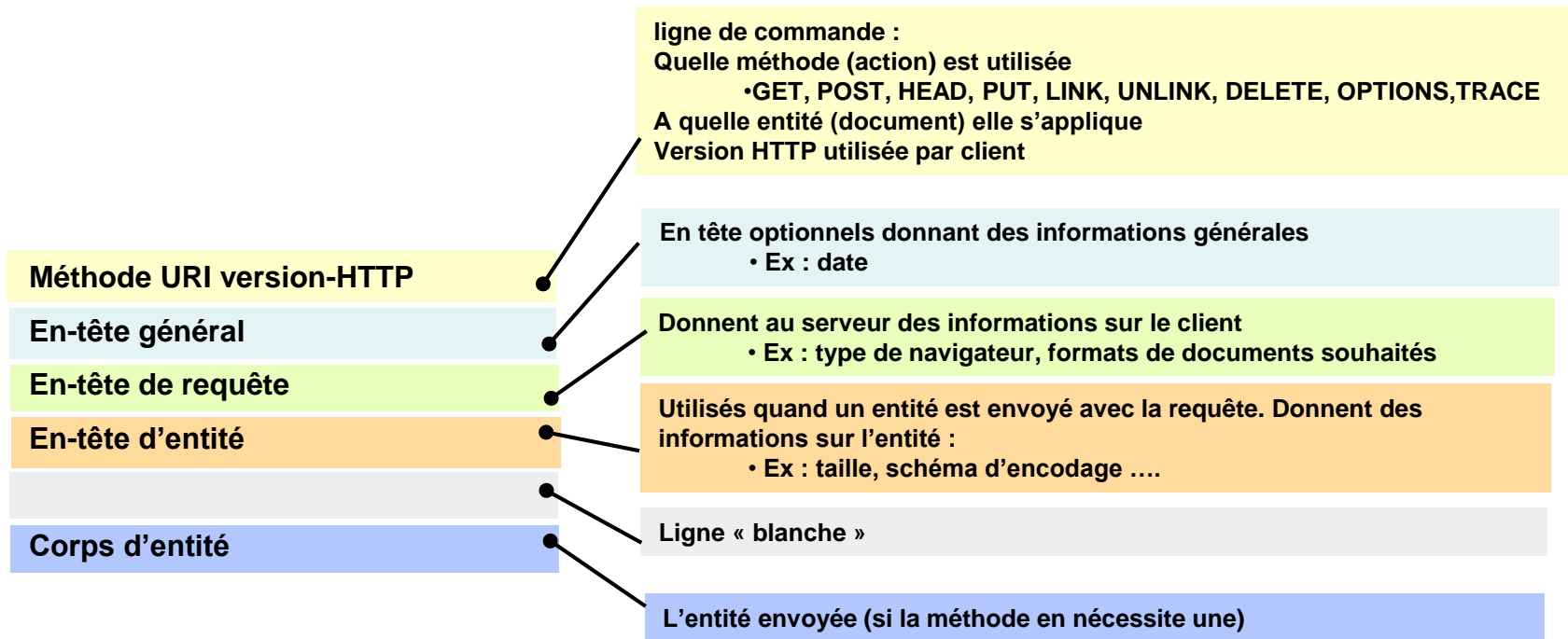
Le client sait traiter fichiers compressés avec algorithme gzip ou deflate

Version 4.0 de Mozilla tournant sur windows NT

Indique le nom de l'hôte du point de vue du client

Demande au serveur de garder la connexion ouverte





méthode URI version

version du protocole HTTP utilisée par le client: 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.
- **HEAD** : ne demande que des informations sur la ressource, sans demander la ressource elle-même.
- **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.
- **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.
- **DELETE** : permet de supprimer une ressource du serveur.

HTTP

requêtes Version HTTP

méthode URI version

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

- **GET** : Spécification HTTP/2 publiée par IETF (Internet Engineering Task Force) en Mai 2015 - Request For Comments (RFC) 7540 in May 2015.

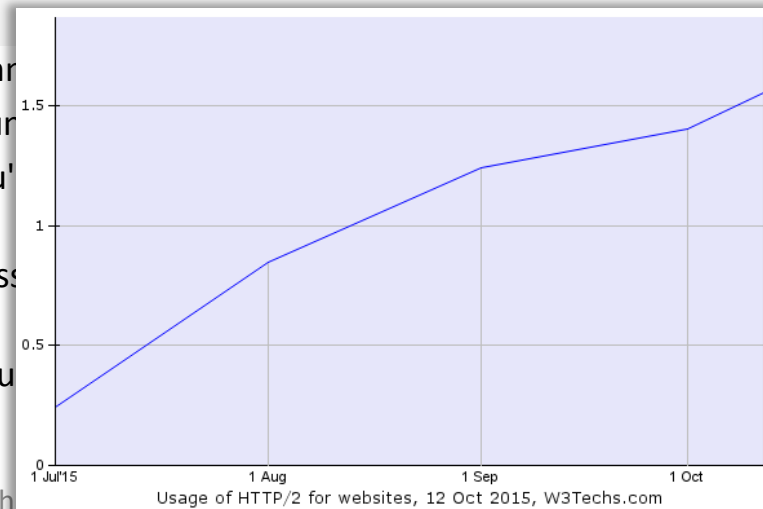
- **OPTIONS** : permet d'obtenir les options de communication disponibles.
- **CONNECT** : permet d'utiliser un proxy comme un tunnel.
- **TRACE** : demande au serveur de retourner ce qu'il reçoit de la connexion.
- **PUT** : permet de remplacer ou d'ajouter une ressource en question.
- **DELETE** : permet de supprimer une ressource du serveur.

Basé sur SPDY (prononcer *speedy*) un protocole réseau développé initialement par Google:

- Améliorer le temps de latence lors du chargement des pages
- Améliorer la sécurité

Une requête GET est sans effet sur la ressource, sans demander la ressource elle-même. Selon W3Techs HTTP/2 utilisé par 1.6 % des sites webs au 12/10/2015

<http://w3techs.com/technologies/details/ce-http2/all/all>



- 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

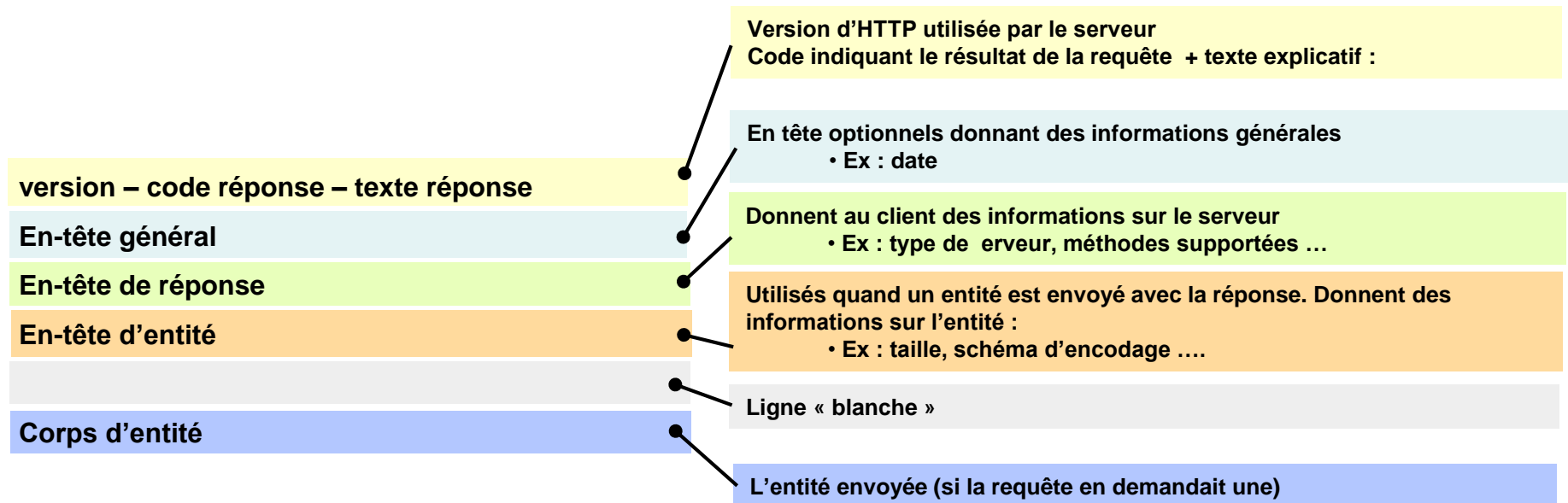
- **Connection: Close|Keep-Alive** (en tête général)
 - spécifie options désirées pour cette connexion
 - **Close** : la connexion est fermée après la réponse (par défaut avec HTTP/1.0)
 - **Keep-Alive** : crée une connexion persistante (par défaut avec HTTP/1.1).
Sur serveurs Apache 2.2 et +, timeout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

réponses structure générale



version – code réponse – 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: HTTP/1.1 ou HTTP/1.0

- **code d'état (sur 3 chiffres) répartis en 4 groupes**
 - **2xx** : requête du client accomplie avec succès
 - **3xx** : requête du client redirigée, d'autres actions sont nécessaires
 - **4xx** : requête du client incomplète
 - **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.
 - la majorité de ces code d'état sont traités de manière transparente pour l'utilisateur sauf certains codes des classes 4 et 5 (ex: 404 Not Found)

www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

- code d'état de classe 2 (succès)
 - **200 OK** : requête traitée avec succès
 - **201 Created** : requête a été traitée et la ressource a été créée
 - **202 Accepted** : requête a été reçue et est en cours de traitement. 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 navigateur devrait effacer le formulaire utilisé pour cette transaction.

- code d'état de classe 3 (redirection – traitement incomplet)
 - **301 Moved Permanently** : La ressource a été assignée à une nouvelle adresse. L'URL est donnée par le champ `Location`
 - **301 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.
- code d'état de classe 4 (erreur client)
 - **400 Bad Request** : Erreur de syntaxe
 - **401 Unauthorized** : La requête nécessite une identification préalable de l'utilisateur
 - **403 Forbidden** : Le serveur refuse de traiter la requête
 - **404 Not Found** : Le serveur n'a pas trouvé la ressource demandée

- code d'état de classe 5 (Erreur serveur)
 - **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.

- 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

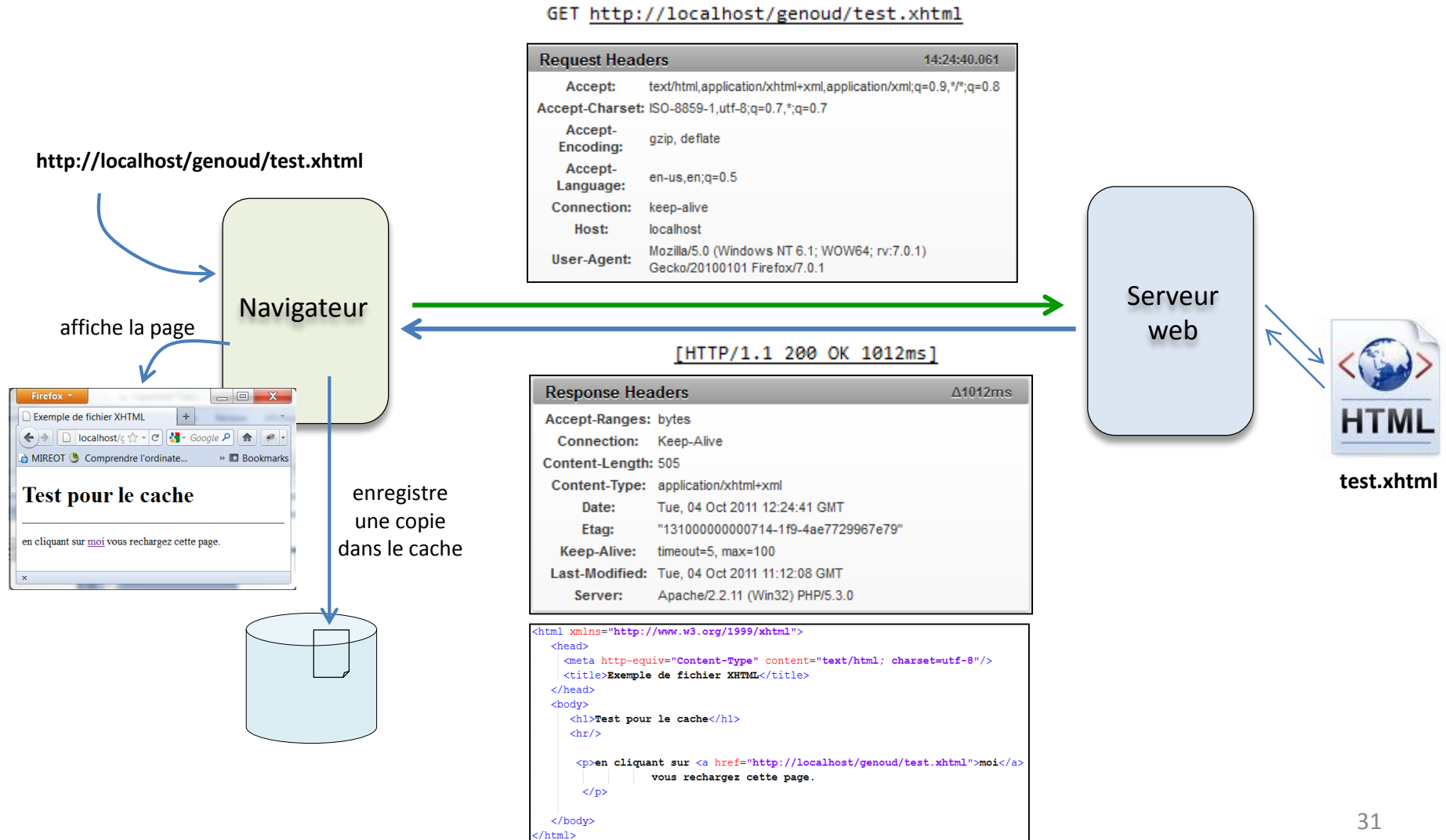
quelques exemples d'en-têtes

- **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

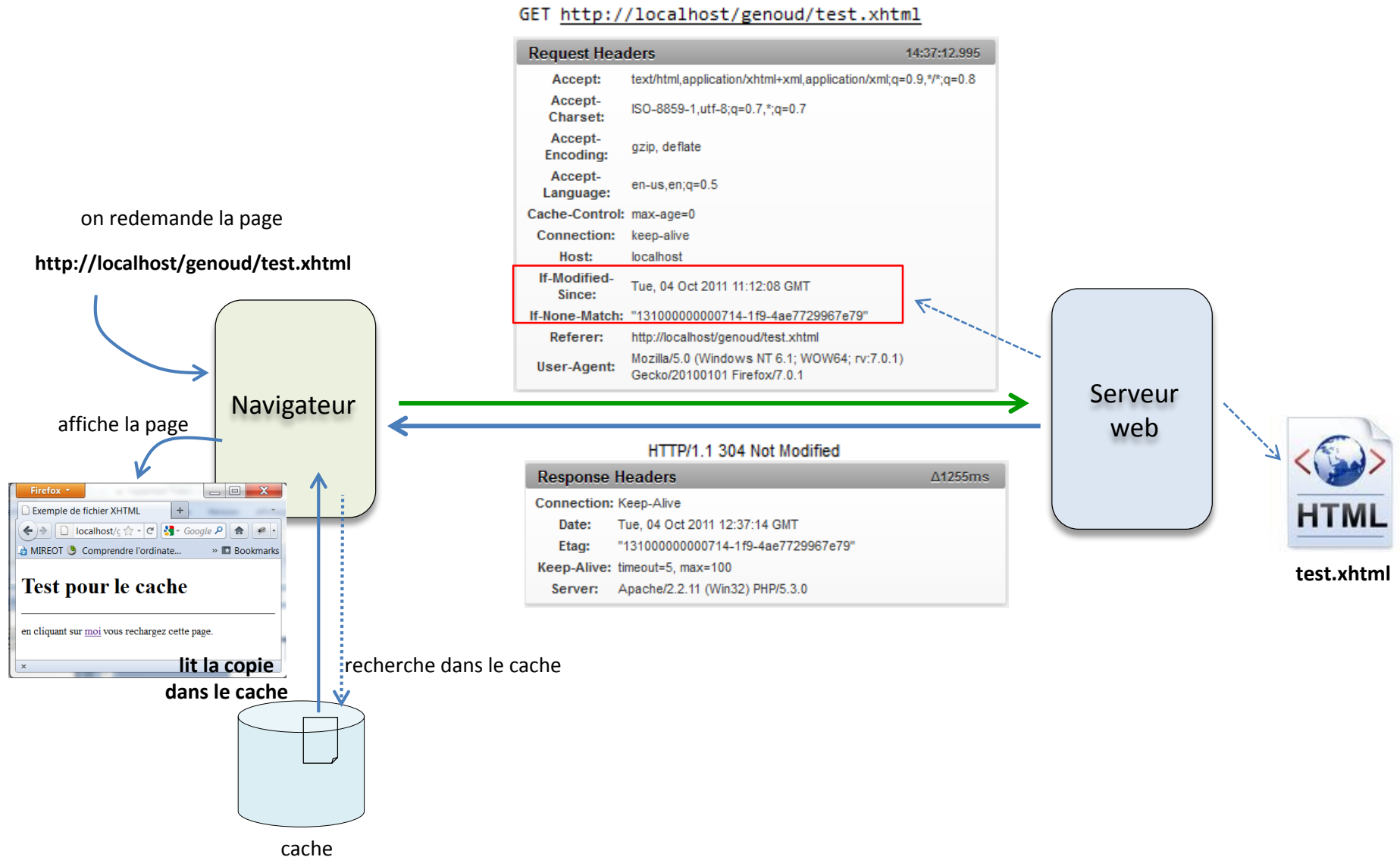
HTTP

Exemple de requêtes utilisation du cache



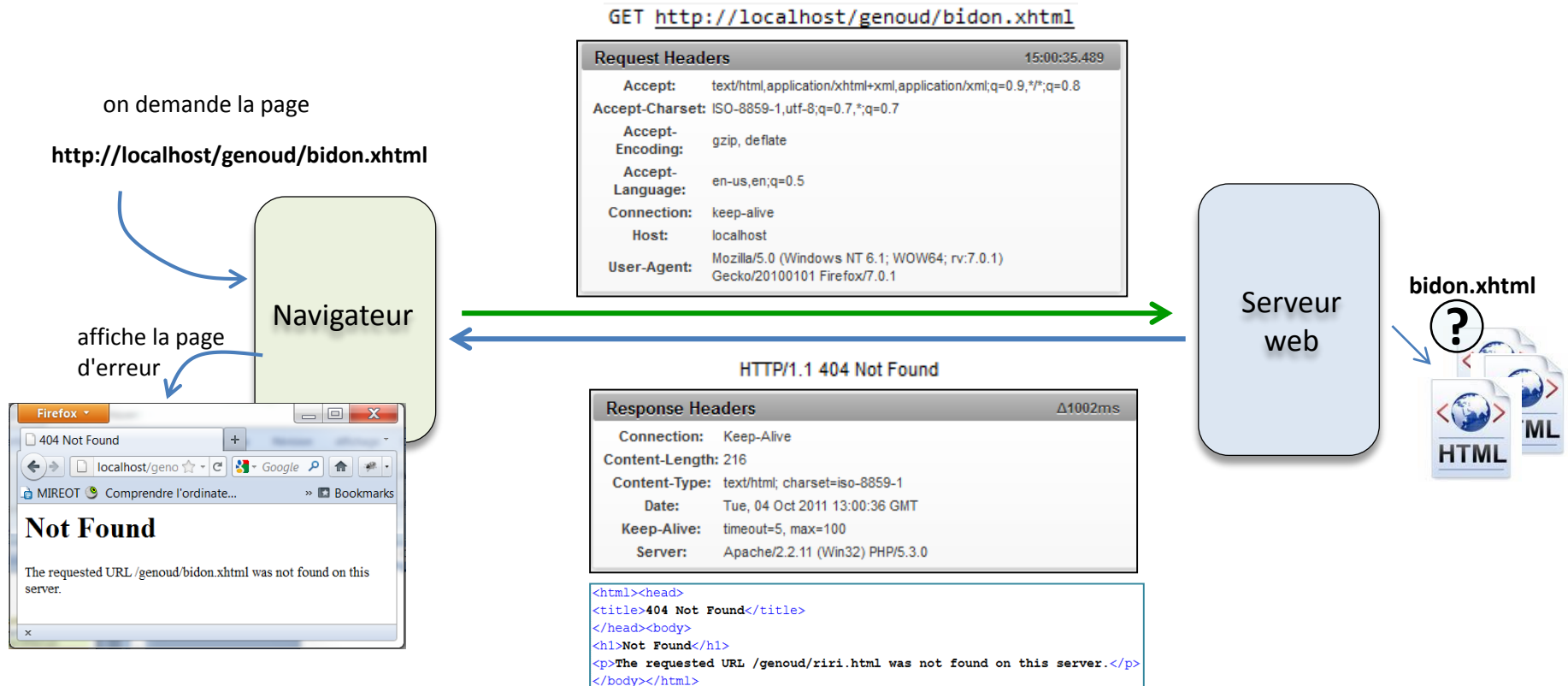
HTTP

Exemple de requêtes utilisation du cache



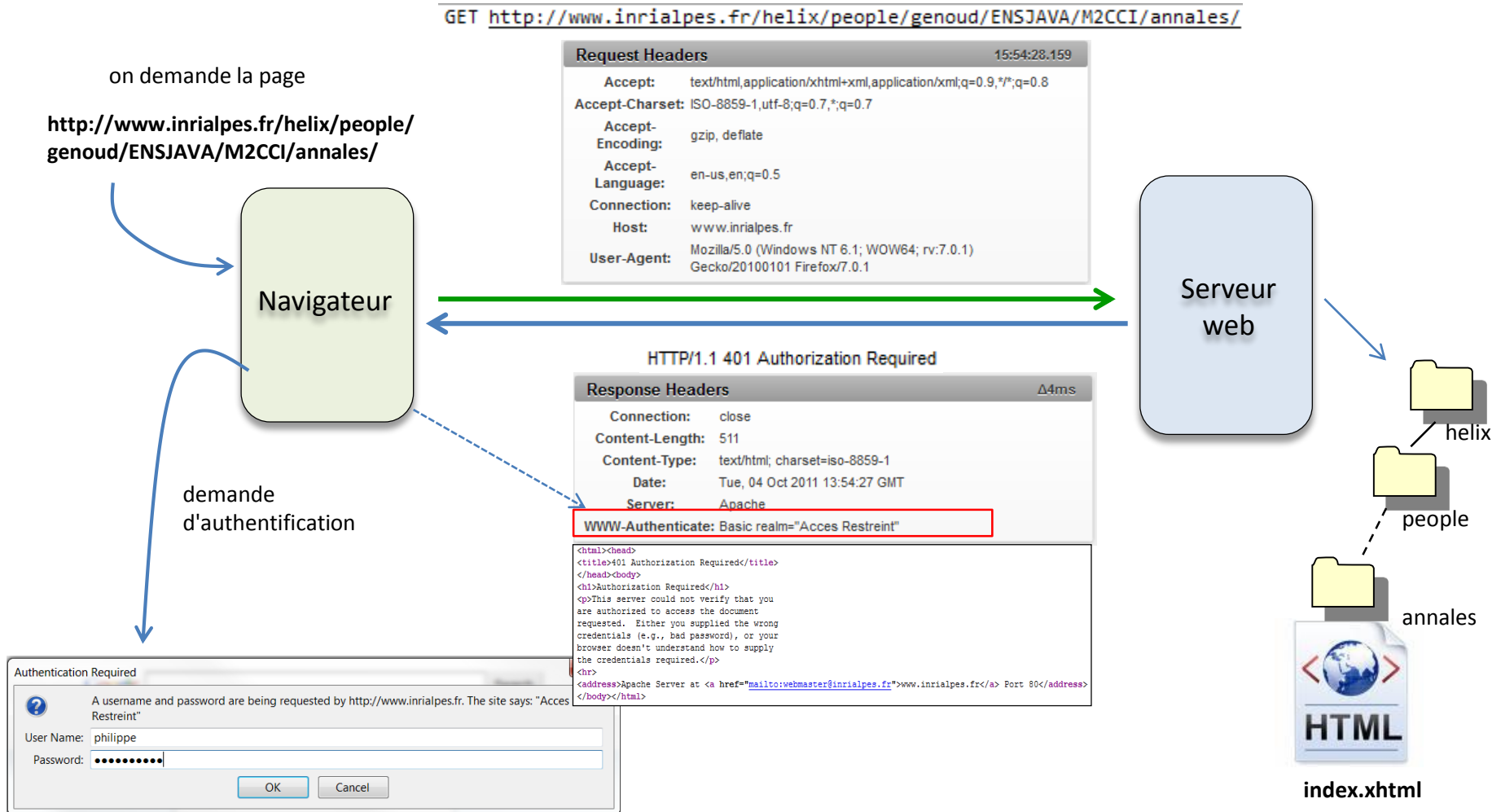
HTTP

Exemple de requêtes réponse d'erreur : 404 Not found



HTTP

Exemple de requêtes réponse d'erreur : 401 Unauthorized



HTTP

Exemple de requêtes réponse d'erreur : 401 Unauthorized

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

GET <http://www.inrialpes.fr/helix/.../genoud/ENSJAVA/M2CCI/annales/>

Request Headers 10:23:40.833

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
Authorization: Basic cGhpbGlwcGU6M21hdGVsb2F1ZA==
Connection: keep-alive
Host: www.inrialpes.fr
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:7.0.1) Gecko/20100101 Firefox/7.0.1

informations codées (Base 64)

Navigateur

affiche la page

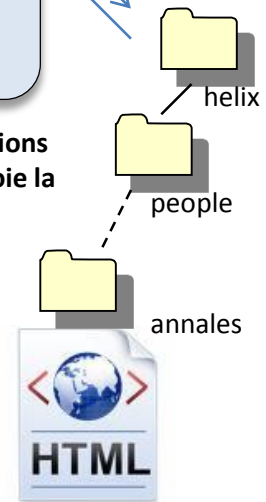
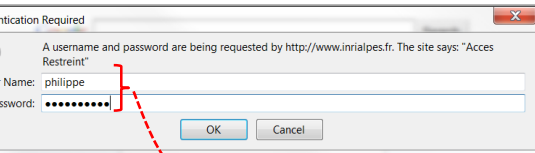
Serveur web

HTTP/1.1 200 OK

Response Headers Δ83ms

Accept-Ranges: bytes
Connection: close
Content-Length: 7238
Content-Type: text/html
Date: Wed, 05 Oct 2011 08:23:40 GMT
Etag: "8f8d7f-1c46-42b7dd80"
Last-Modified: Fri, 18 Mar 2011 15:16:22 GMT
Server: Apache

le serveur vérifie les autorisations
et si elles sont correctes renvoie la page.



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**

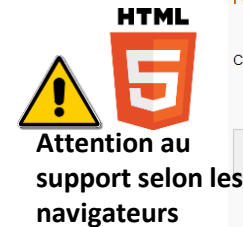
formulaire HTML

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>`

HTML 5 propose de nouveaux types:

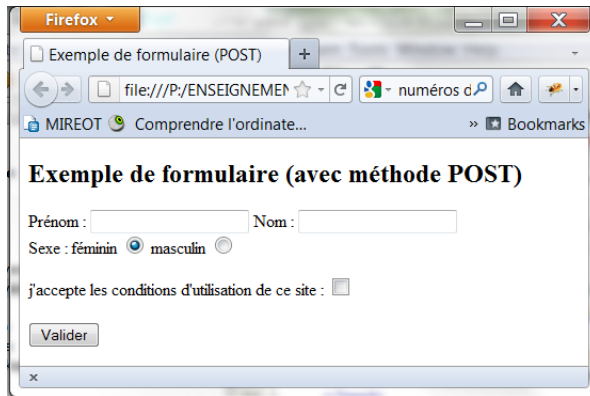
- zone de saisie de date
- zone de saisie de couleur
- zone de saisie d'adresse mail
- sliders ...



pour en savoir plus

<http://www.coreservlets.com/html5-tutorial/input-types.html>

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



Exemple de formulaire (avec méthode POST)

Prénom : Nom :

Sexe : féminin ☒ masculin ☐

j'accepte les conditions d'utilisation de ce site : ☐

```
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>
```

HTTP

formulaire HTML envoi de données au serveur (POST)

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

POST <http://localhost:8084/ExempleFormulaire/validationFormulaire>

Exemple de formulaire (avec méthode POST)

Prénom : Amélie

Sexe : féminin ☒ masculin ☐

Nom : LE POULAIN

j'accepte les conditions d'utilisation de ce site : ☒

Valider

Les variables définies dans le formulaire ont été associées avec les valeurs entrées par l'utilisateur et passées au serveur dans le format URL-Encoded

Navigateur

affiche la page

JSP Page

Bonjour Mme Amélie LE POULAIN

Vous avez accepté les conditions du site.

Request Headers		23:38:10.536
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/FormulaireAvecPost.xhtml	
User-Agent:	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:7.0.1) Gecko/20100101 Firefox/7.0.1	
<hr/>		
Content-Type:	application/x-www-form-urlencoded	
Content-Length:	56	
<hr/>		
prenom=Am%C3%A9lie&nom=LE+POULAIN&sexe=feminin&accept=ON		

données envoyées avec le corps de la requête

le serveur traite la requête POST et ses en-têtes puis passe le corps de la requête au programme spécifié par l'URL qui le traite.

Serveur web

[HTTP/1.1 200 OK 14ms]

Response Headers Δ20ms

Content-Length: 443

Content-Type: text/html; charset=UTF-8

Date: Tue, 04 Oct 2011 21:38:10 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>
```



HTTP

formulaire HTML envoi de données au serveur (GET)

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

Exemple de formulaire (avec méthode GET)

Prénom : Amélie

Sexe : féminin ☒ masculin ☐

Nom : LE POULAIN

j'accepte les conditions d'utilisation de ce site : ☐

Valider

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

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/FormulaireAvecGet.xhtml

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:7.0.1) Gecko/20100101 Firefox/7.0.1

Les données sont envoyées dans l'URL

affiche la page

Navigateur

[HTTP/1.1 200 OK 14ms]

Serveur web



JSP Page

Bonjour Mme Amélie LE POULAIN

Vous avez accepté les conditions du site.

Response Headers Δ21ms

Content-Length: 443

Content-Type: text/html; charset=UTF-8

Date: Tue, 04 Oct 2011 21:55:04 GMT

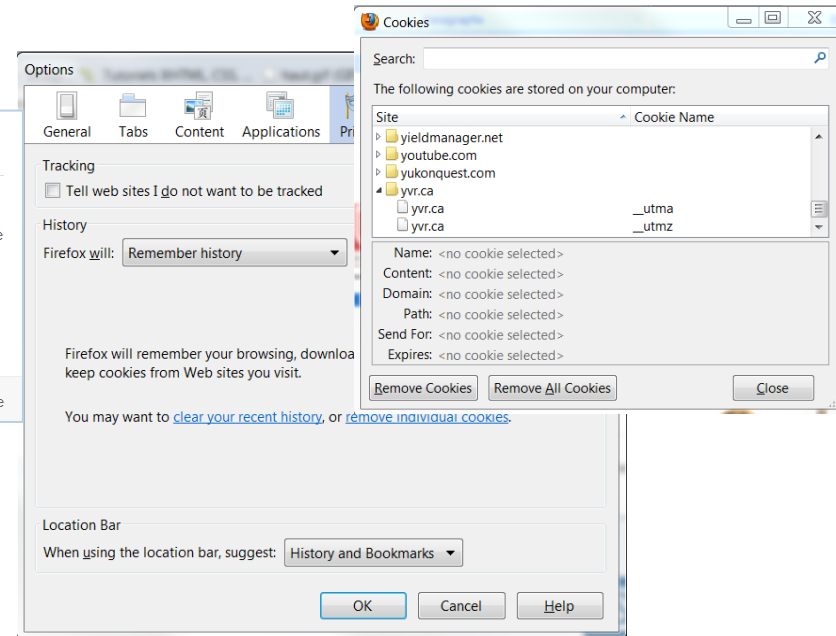
Server: Apache-Coyote/1.1

```
<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>
```

about:cache

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

- les données sont ajoutées à la fin de l'URL
 - ne peuvent contenir espaces, saut de ligne
 - format spécial : **URL-Encoded**
- 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 hexadécimal
 - ex : '=' → %3D (voir http://www.w3schools.com/tags/ref_urlencode.asp)
 - fonctions dans les langages de programmation pour encoder/décoder les URL :
`encodeURIComponent()` **JavaScript**, `rawurlencode()` **PHP**, `Server.URLEncode()` **ASP...**



• Qu'est-ce qu'un cookie ?



ne pas confondre **un** cookie
et **une** cookie !!

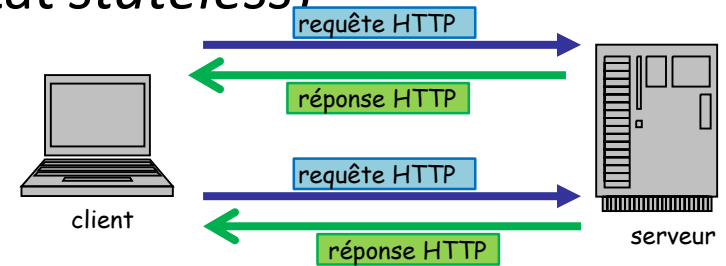


Cookie : Témoin de connexion

- Un petit (4Ko) fichier texte déposé sur le disque dur du client
- Permet au serveur de reconnaître l'utilisateur lorsqu'il reviendra ensuite sur le site.

- HTTP protocole sans mémoire (sans état *stateless*)

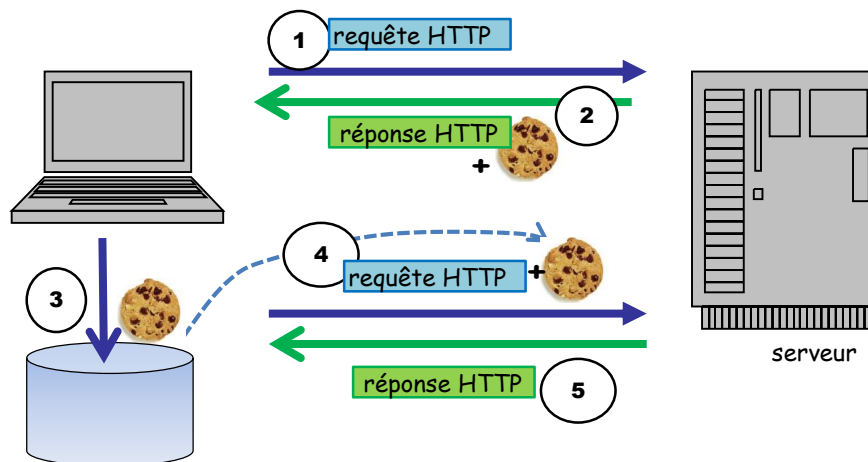
- indépendance de chaque requête
- pas de contrôle de l'ordre de navigation



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



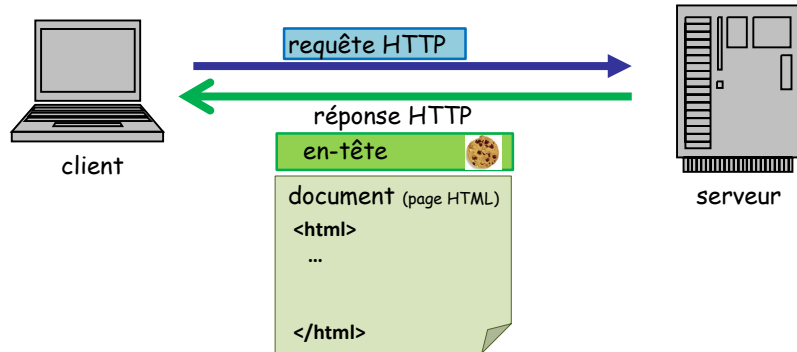
1. le client émet une requête HTTP
2. le serveur renvoie sa réponse à laquelle il associe un cookie
3. le client stocke localement le cookie
4. lors de requêtes suivantes vers le serveur le client transmet le cookie avec la requête
5. le serveur peut adapter sa réponse en fonction du client et des traitements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

- 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

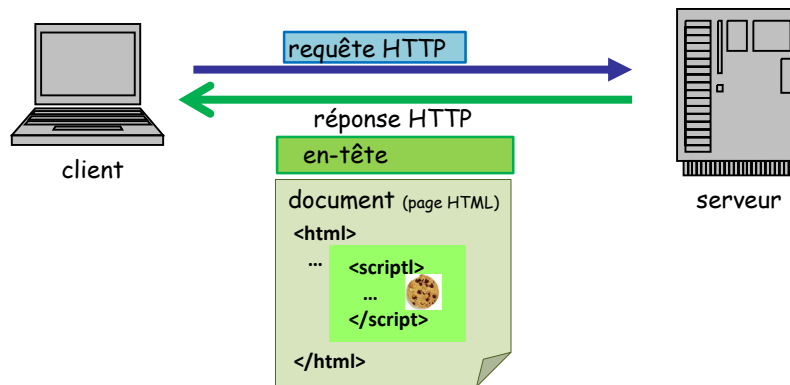
HTTP

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

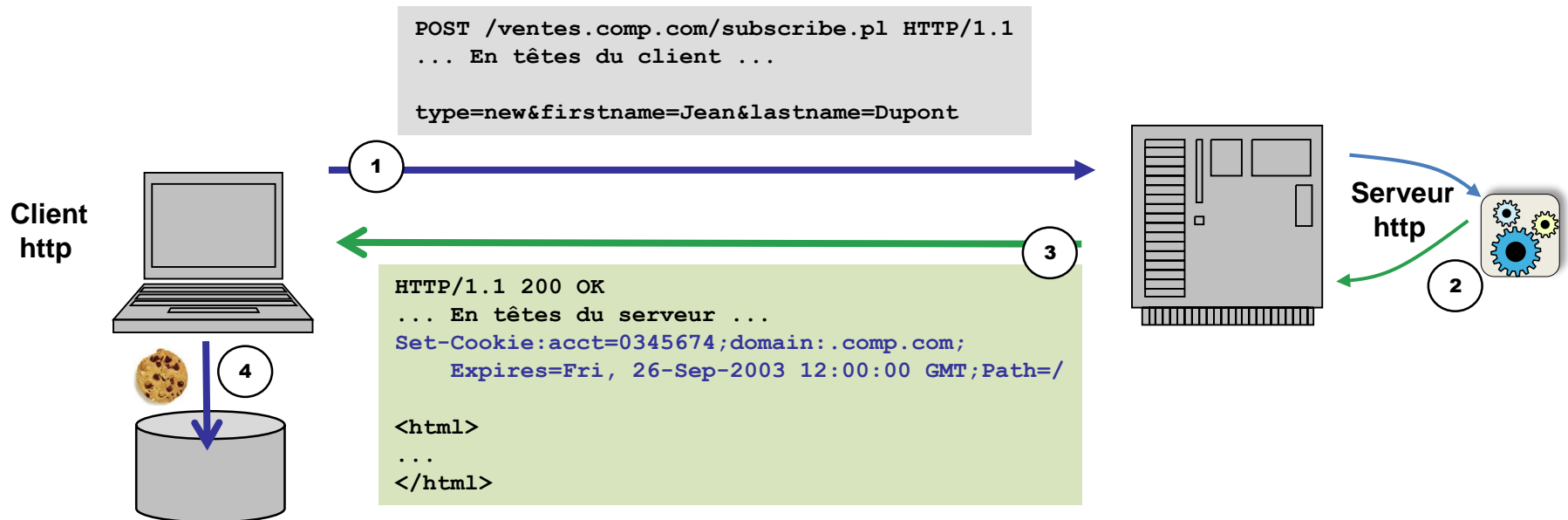


2ème méthode : les instructions de création de cookie (écrites dans un langage de programmation ex: javascript) sont encapsulées dans une page HTML

HTTP

Cookies dépôt de cookies

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



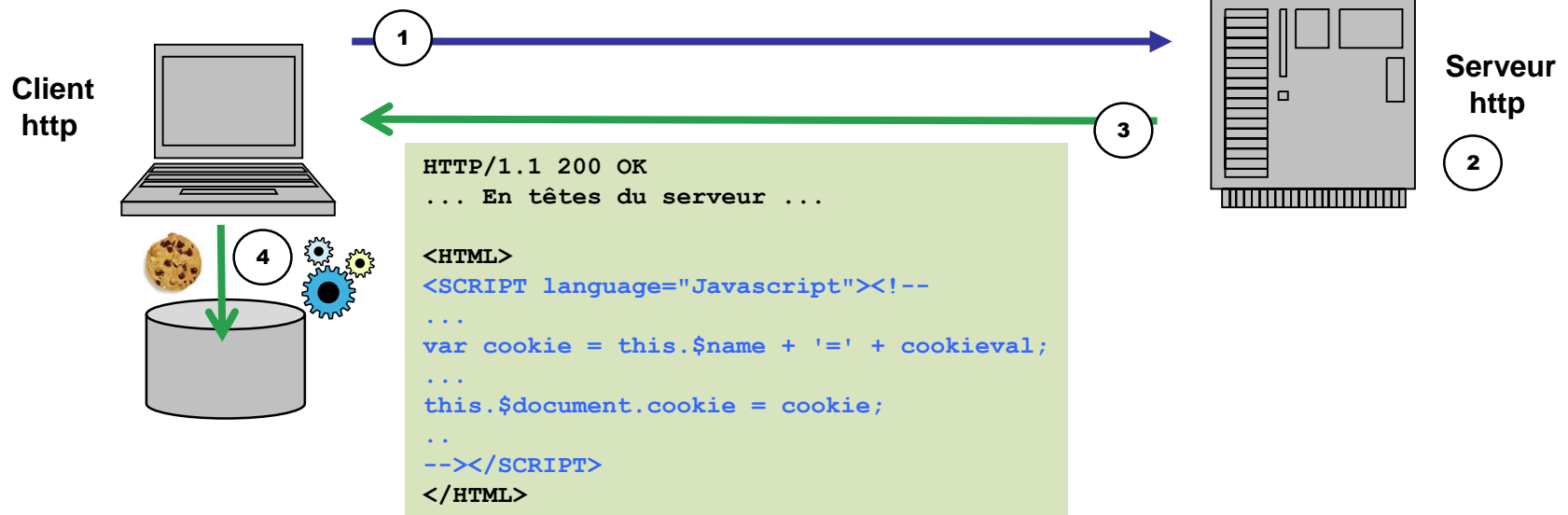
1. Le client émet une requête
2. Un programme du serveur traite cette requête et souhaite stocker des informations d'états chez le client
3. Le programme du serveur génère un en-tête **Set-Cookie** dans la réponse HTTP envoyée au client
4. Le programme client (navigateur) analyse la réponse et stocke le cookie dans un fichier sur le disque dur du client

HTTP

Cookies dépôt de cookies

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

```
POST /monsite.amoi.org HTTP/1.1  
... En têtes du client ...
```



- 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

HTTP

Cookies attributs des cookies

- attributs d'un cookie



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

The screenshot shows a web browser window displaying the Moodle platform interface. The address bar shows the URL `imag-moodle.e.ujf-grenoble.fr`. The page content includes a navigation menu, site news, and a calendar. Below the browser window, a table displays the cookies stored in the browser's session storage.

Name	Value	Domain	Path	Expires	Size	HTTP	Secure
MoodleSession	cf23e79be243049b4dfc9a9f72aebda6	imag-moodle.e.ujf-grenoble.fr	/	Session	45		
MOODLEID_	%25E4%25C9%2510C%25A6y%25A0%2506	imag-moodle.e.ujf-grenoble.fr	/	Tue, 29 Nov 2011 14:19:...	41		

- 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.

- 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 scripts clients comme javascript)

```
HTTP/1.1 200 OK
Date: Tue, 07 Aug 2010 21:36:13 GMT
Server: Apache-AdvancedExtranetServer/1.3.19
Set-Cookie: Id=Toto; path=/
Set-Cookie: NbVisites=12; path=/~reignier/Cookies
Connection: close
Content-Type: text/html
```

<html>

...

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

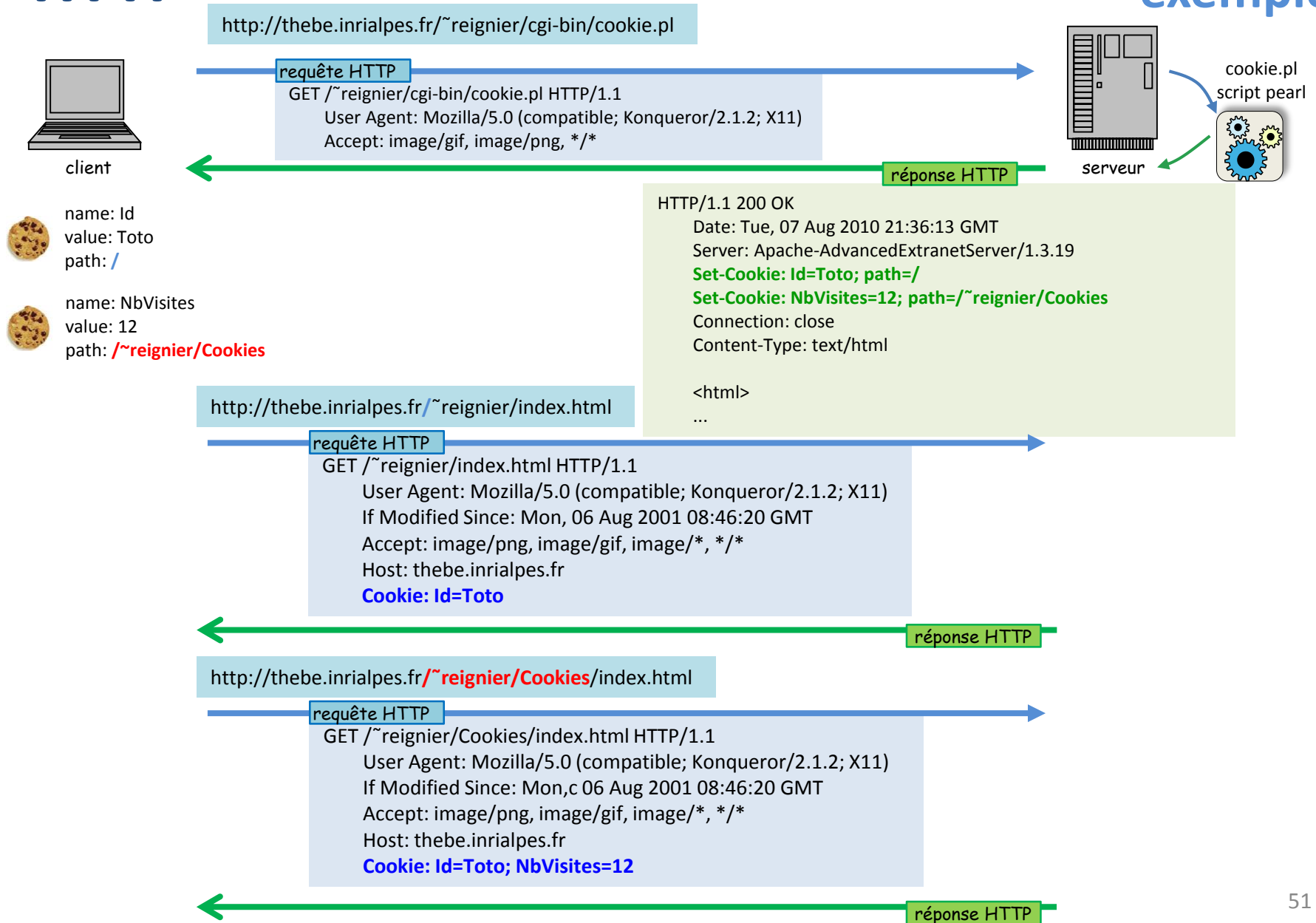
- lorsque le client établit une requête pour accéder à une URL
 1. recherche parmi les cookies mémorisés ceux s'appliquant au serveur (attribut **domain**) et à l'URL (attribut **path**) et n'ayant pas expirés
 - le serveur appartient au même domaine que celui spécifié par l'attribut **domain**
 - si 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;*

```
GET /~reignier/Cookies/index.html HTTP/1.1
User Agent: Mozilla/5.0 (compatible; Konqueror/2.1.2; X11)
If Modified Since: Mon,c 06 Aug 2001 08:46:20 GMT
Accept: image/png, image/gif, image/*, */*
Host: thebe.inrialpes.fr
Cookie: Id=Toto; NbVisites=12
```

HTTP

Cookies exemples



Alternatives au cookies

- des solutions "propriétaires"
 - 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 HTML 5.

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

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

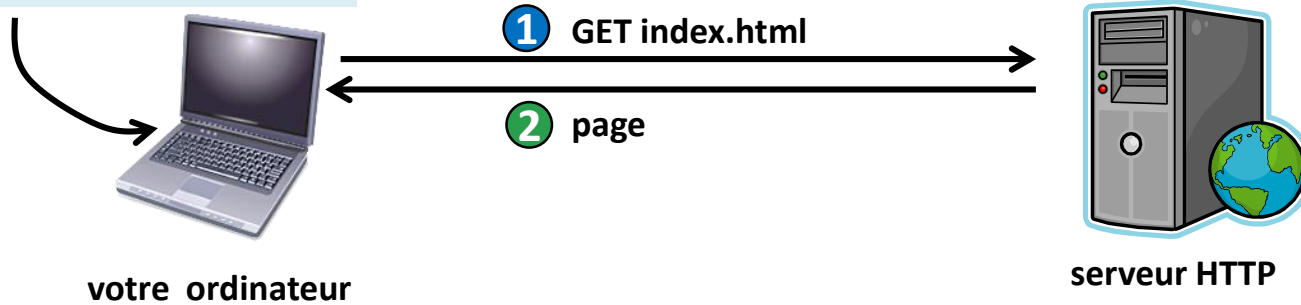
<http://www.alsacreations.com/article/lire/1402-web-storage-localstorage-sessionstorage.html>

→ applications *offline* : web déconnecté

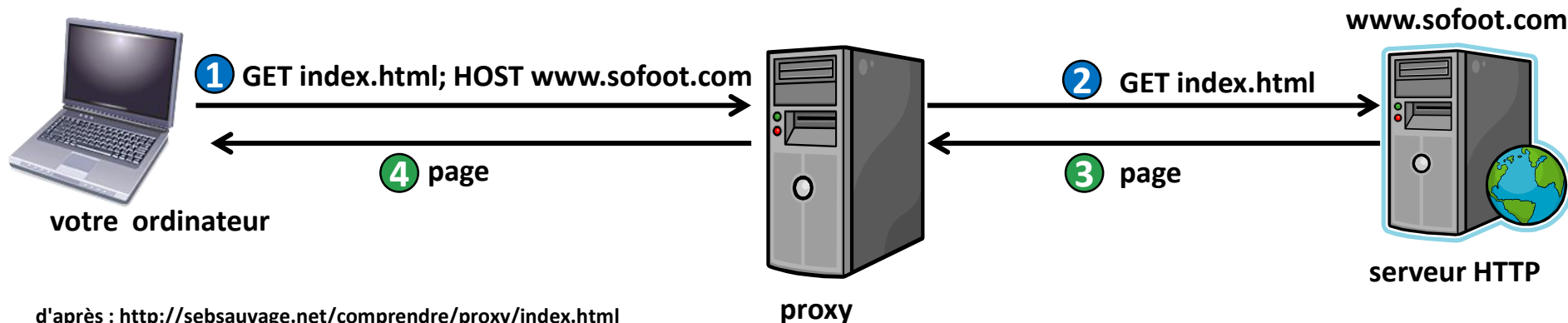
HTTP

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

`http://www.sofoot.com/index.html`



- 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

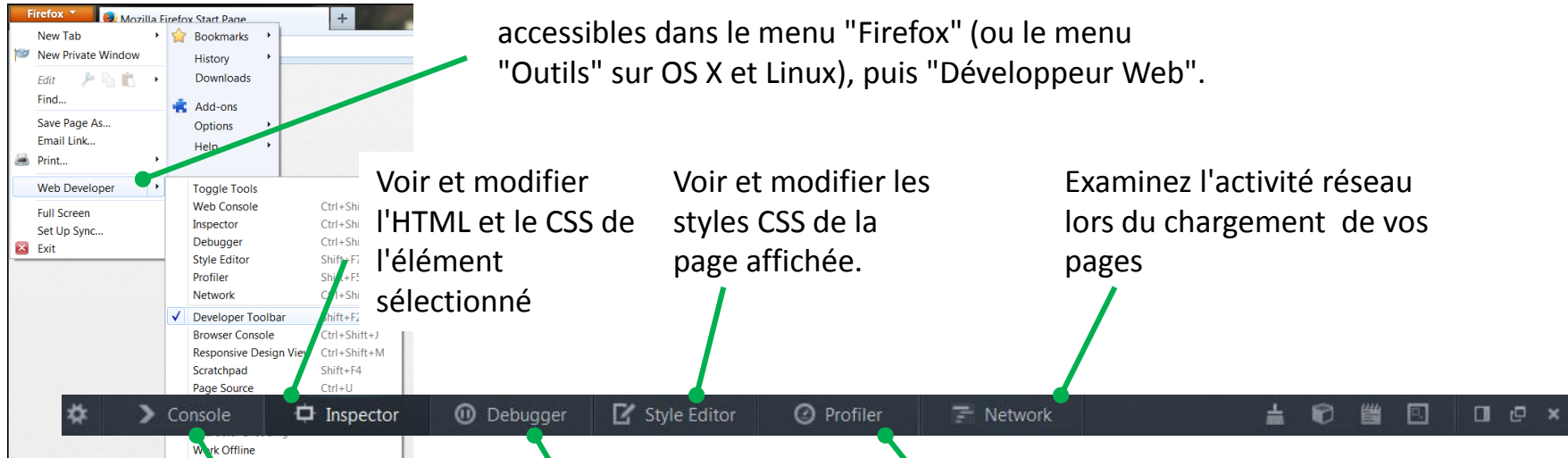
- intérêts d'un serveur proxy
 - accélération de la navigation (proxy-cache)
 - mise en cache des pages les plus demandées
 - sécurité du réseau local
 - autorise votre ordinateur à se connecter à l'extérieur mais interdit aux ordinateurs d'internet de se connecter sur le votre.
 - filtrage
 - interdit 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

Firefox Web Developer Tools

- Firefox intègre des outils pour l'aide au développement de sites web.



The screenshot shows the Firefox Web Developer Tools interface. The 'Firefox' menu is open, showing options like 'Web Developer'. The 'Web Developer' submenu is also open, showing options like 'Toggle Tools', 'Web Console', 'Inspector', 'Debugger', 'Style Editor', 'Profiler', 'Network', 'Developer Toolbar', 'Browser Console', 'Responsive Design View', 'Scratchpad', and 'Page Source'. The 'Developer Toolbar' is highlighted. The 'Web Developer' toolbar is visible at the bottom, showing icons for 'Console', 'Inspector', 'Debugger', 'Style Editor', 'Profiler', and 'Network'. Green lines connect text annotations to specific parts of the interface.

accessibles dans le menu "Firefox" (ou le menu "Outils" sur OS X et Linux), puis "Développeur Web".

Voir et modifier l'HTML et le CSS de l'élément sélectionné

Voir et modifier les styles CSS de la page affichée.

Examinez l'activité réseau lors du chargement de vos pages

Voir des informations, des messages d'erreurs ou d'avertissements émis par le navigateur ou la page web. Permet aussi d'examiner et de manipuler le JavaScript de la page

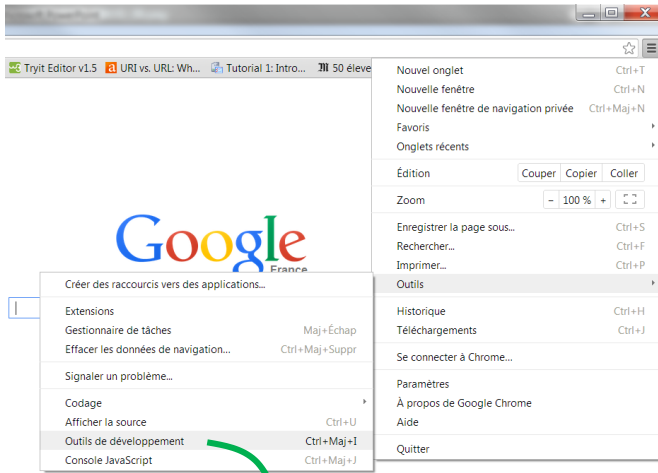
Parcourez votre code JavaScript s'exécutant dans le navigateur et observez les variables pour chasser les bugs

Utilisez le profileur pour savoir à quels endroits votre code JavaScript passe le plus de temps.

<https://developer.mozilla.org/en-US/docs/Tools>

<https://developer.mozilla.org/fr/docs/Outils?redirectlocale=fr&redirectslug=Tools>

Chrome Web Developer Tools



- Chrome et les autres navigateurs web intègrent des outils similaires à ceux de Firefox

<https://developers.google.com/chrome-developer-tools>

<https://devtoolsecrets.com/>

