GRENOBLE INP UGA Ensimag

Chapitre AP_www

Retour sur WWW

APplications

Contenu du chapitre AP_www

- Les bases historiques et techniques de WWW
  - Hypermédia réparti, fusion de la PAO et des réseaux
  - URL, HTTP, HTML
- HTTP
- HTML: données dynamiques
  - Formulaires
- Documents Web dynamiques
  - CGI & appliquettes
- Cookies, proxies, caches
- Sécurité

WWW : une application tentaculaire sur l'Internet

- Hypermédia réparti
  - documents multimédia qui incluent des objets images, son, vidéo
  - liens entre des objets répartis
- Un maillage d'objets/documents au-dessus du maillage des « hosts »

- Devenu une interface privilégiée pour les applications informatiques, accès à un SI depuis un PC
- Mais a perdu ce rôle pour les accès par mobile

WWW: fusion de deux lignées

Accès par réseau aux documents

↓

`ftp`

`archie, gopher`

Mise en forme de documents (PAO)

↓

Outils de PAO

`SGML`

`HTTP` + **`HTML`** = WWW

CERN (Genève) 1990: Tim Berners-Lee & Robert Cailliau

Les bases techniques du WWW

- URL - désignation de documents (*adresse*)
- HTTP - *protocole* d'accès aux documents
- HTML - structure et présentation des documents (*objets textuels*)

- CGI, PHP, Javascript, applets - extension de fonctionnalités, génération de documents à la volée
- Objets multimédia: textes, images, vidéo, son
  - et codages associés
- Navigateurs et afficheurs
  - Client et extension

WWW: les sigles

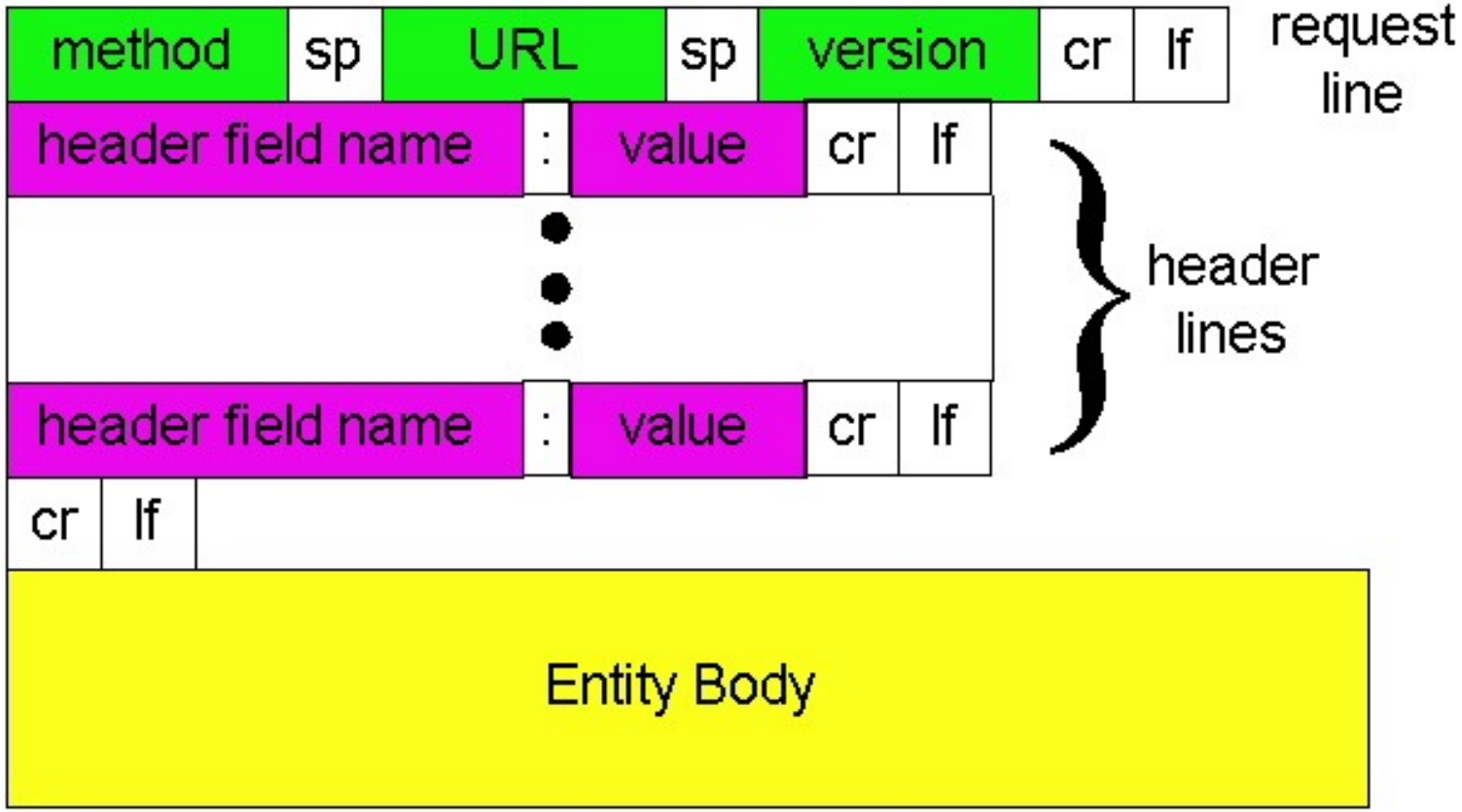
- WWW = *World-Wide Web*
- URL = *Uniform Resource Locator*
- HTTP = *HyperText Transfer Protocol*
- HTML = *HyperText Markup Language*
- CSS = *Cascading Style Sheet*
- SSL = *Secure Sockets Layer* – TLS= *Transport Layer Security*
- CGI = *Common Gateway Interface*
- PHP = *HyperText PreProcessor*
- XML = *eXtensible Markup Language*
- Ajax = *Asynchronous Javascript And XML*

Déjà vus: URL (AR2-adresses), HTTP

Retour sur HTTP, CGI, SSL/TLS: ce cours et le TP Web

Cours 2A: XML, Ajax, Javascript & JSP (cours CAWEB)

HTTP: les PDUs de requêtes



method=GET, HEAD ou POST
(+OPTIONS, PUT, DELETE, TRACE, CONNECT)
sp=espace, cr lf = fin de ligne

HTTP: exemple de PDU de requête

Nom du serveur demandé(*)

```
GET / HTTP/1.1
Host: web.ensimag.fr
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.5;...
Accept: text/html,application/xhtml+xml,application/xml;...
Accept-Language: fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

Préférences de langue

Connexion persistante

(*) Car un même serveur (ex logiciel Apache à l'adresse IP 195.220.30.42) peut héberger plusieurs hôtes « virtuels », comme web.ensimag.fr, intranet.ensimag.fr etc

HTTP: exemple de PDU de réponse

"Status"
code de retour

en-tête

```
HTTP/1.1 200 OK
Date: Fri, 23 Sep 2011 09:37:56 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Mon, 01 Mar 2010 15:20:22 GMT
ETag: "554735-578-480bec9a4b980 »
Accept-Ranges: bytes
Content-Length: 1400
Connection: close
Content-Type: text/html
```

données
HTML
demandées
par la requête

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01">
<html>
<head>
<title>Extranet ensimag</title>
...
```

HTTP: les codes de retour

- Codes les plus fréquents

200 OK

- requête correcte, le contenu demandé suit

301 Moved Permanently

- l'objet a été déplacé, sa nouvelle URL est donnée en-dessous dans le champ "`Location:`"

400 Bad Request

- requête non comprise par ce serveur

401 Authorization Required

- le client doit s'authentifier (les requêtes pour cette URL doivent contenir un champ "`Authorization:`")

404 Not Found

- URL désigne un objet inexistant/inconnu

505 HTTP Version Not Supported

HTTP/1.1: connexions persistantes

- En HTTP/1.0 d'origine: chaque URL fait l'objet d'une ouverture de connexion, GET+Réponse, fermeture de connexion
  - NB: les navigateurs peuvent lancer plusieurs connexions en parallèle pour le chargement des images
- En HTTP/1.1, les connexions peuvent être persistantes (Keep-Alive)
  - après la Réponse, on garde la connexion pour charger les images par exemple, on ne la ferme qu'après la dernière image de la page
  - la persistance et sa durée dépendent du bon vouloir du serveur

Les langages pour décrire les pages

- HTML 1.0 (~1991)
  - issu de SGML (hypertexte)
  - mélange structure (paragraphes...) et présentation (polices...) dans un même fichier
- HTML 2.0 (1995)
  - bi-directionnel (formulaires: données client->serveur)
  - séparation des règles de présentation: fichiers CSS
- HTML 4.0 (1997): frames, objets
- Apparition XML (1996), XHTML, DOM (1998)
- HTML5 (2008): intégration d'API pour app. Web
  - Dont stockage rémanent de données dans navigateur

Contenu du chapitre AP_www

- Les bases historiques et techniques de WWW
  - Hypermédia réparti, fusion de la PAO et des réseaux
  - URL, HTTP, HTML
- HTTP
- HTML: données dynamiques
  - Formulaires
- Documents Web dynamiques
  - CGI & appliquettes
- Cookies, proxies, caches
- Sécurité

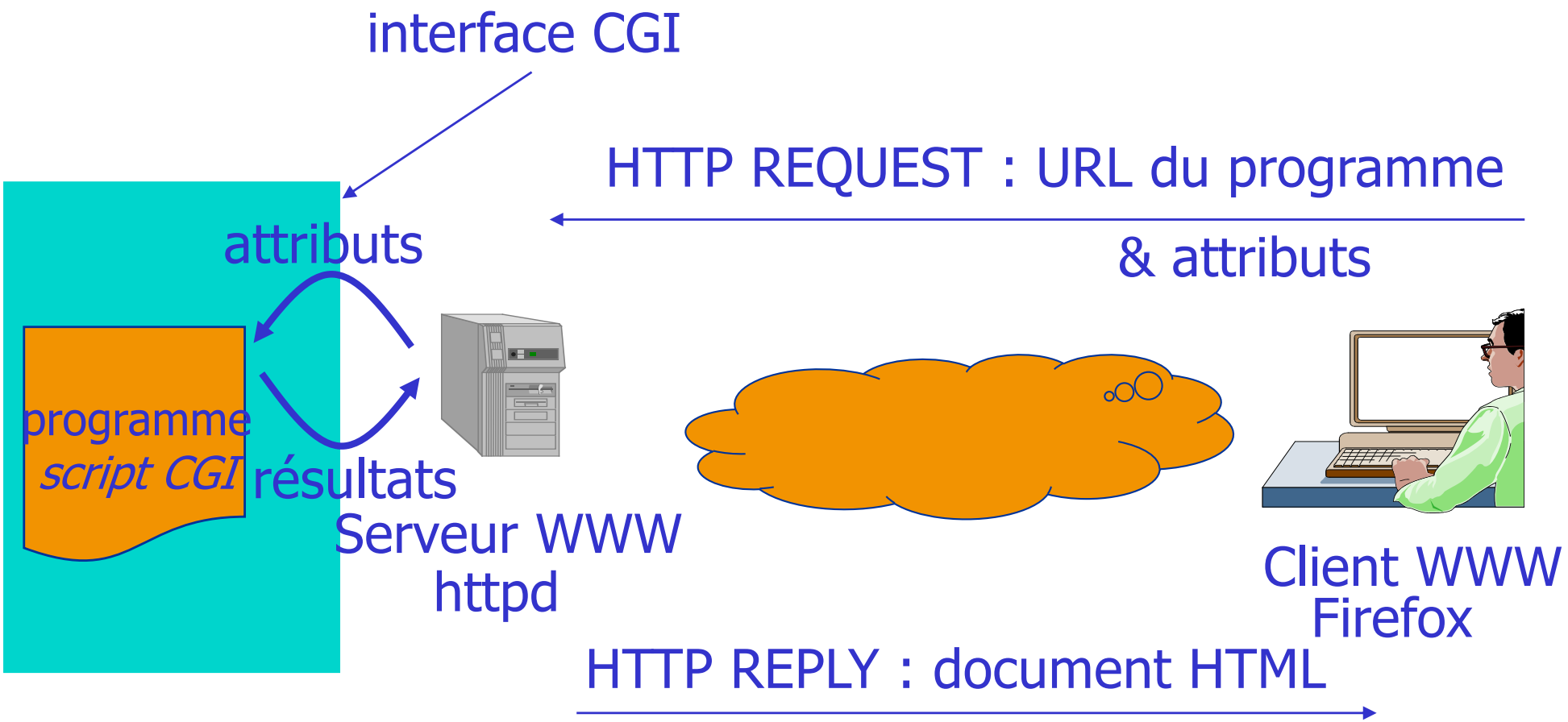
Documents Web dynamiques: 2 formes

- Génération de pages à la volée par le serveur
  - Dans ce cas, URL -> un programme de génération d'HTML (script CGI)
  - Document PHP: HTML + commandes au pré-processeur
    (PHP: Open Source, JSP/servlet: Sun, ASP: M/S)

- Génération de contenu côté client
  - (Applets Java: caduc)
  - Scripts JavaScript

Cours 2A Conception Applis Web: JSP, Javascript et programmation Ajax et de servlets

CGI


interface CGI
HTTP REQUEST : URL du programme
& attributs
attributs
programme
script CGI
résultats
Serveur WWW
httpd
Client WWW
Firefox
HTTP REPLY : document HTML

CGI

- Invocation de programmes qui s'exécutent sur le serveur
- Document formulaire (le cas échéant) saisi sur le client
- Requête POST (*)
  - URL désigne un programme
  - formulaire codé comme des attributs
    - paires nom-valeur, séparés par « & »
- httpd exécute le programme en lui passant les attributs
- Le programme engendre un document HTML et le passe à httpd (le processus serveur Web)
- httpd renvoie ce document HTML dans sa réponse HTTP

(*) Si pas d'effet de bord, on peut aussi utiliser une requête GET avec les paires nom-valeur passés en paramètre (après le « ? »)

Formulaire

Demonstration POST

Nouvel onglet

file:///Users/groz/Cours/Reseau_1A/Demos_www 110 %

Demonstration POST

Please help us to improve the World Wide Web by filling in the following questionnaire:

Your organization? Grenoble INP

Commercial? ☐ How many users? 900

Which browsers do you use?

1. Firefox ☑
2. Internet Explorer ☐
3. Chrome ☑
4. Safari ☑
5. Opera ☐
6. Others

A contact point for your site: roland.groz@imag.fr

Many thanks on behalf of the WWW central support team.

Envoyer Effacer

Source HTML du formulaire

```
<form method=POST action="http://localhost/cgi-
 bin/postecho.sh">

Please help us to improve the World Wide Web by filling in
 the following questionnaire:

<p>Your organization? <input name="org" type=text size="48">
 </p>
<p>Commercial? <input name="commerce" type=checkbox>

How many users? <input name="users" type=int> </p>

<p>Which browsers do you use? </p>
```

Formulaires

- `<form>` formulaire
  - attributs : `action, method, enctype`
- `<input>` champ d'entrée
  - attributs : `name, type, value, checked, size, maxlength`
- `<select>` liste de sélection (menu déroulant)
  - attributs : `name, multiple, size`
  - `<option>` une sélection (d'un `select`)
    - attribut : `selected`
- `<textarea>` zone de texte à remplir
  - attributs : `name, rows, cols`

Type de `input`

- `text` ligne de texte
- `checkbox` choix multiple
  - valeur dans l'attribut `name`
- `radio` un choix parmi plusieurs
- `image` image cliquable, renvoie les coordonnées du clic
- `submit` émission -> provoque l'envoi d'un POST
- `reset` valeurs initiales

Exemple formulaire (suite)

```
<ol>
 <li>Firefox <input name="browsers" type=checkbox
  value="firefox"> </li>
 <li>Internet Explorer <input name="browsers" type=checkbox
  value="IE"> </li>
 <li>Chrome <input name="browsers" type=checkbox
  value="Chrome"> </li>
 <li>Safari <input name="browsers" type=checkbox
  value="Safari"> </li>
 <li>Opera <input name="browsers" type=checkbox
  value="Opera"> </li>
 <li>Others <input name="others" size=40> </li>
</ol>
A contact point for your site: <input name="contact"
  size="42">
<p>Many thanks on behalf of the WWW central support team.</p>
<p><input type=submit> <input type=reset></p>
</form>
```

Formulaire
Envoi du POST (requête HTTP)


```
POST /cgi-bin/postecho.sh HTTP/1.1
Host: localhost
...
Content-type: application/x-www-form-urlencoded
Content-length: 113

org=Grenoble+INP&users=900&browsers=firefox&browsers
  =Chrome&browsers=Safari&others=&contact=roland.groz%40
  imag.fr
```

Nota Bene:

`urlencoded: %nn` caractère spécial de code hexadécimal `nn` en ASCII, `+` est une abréviation de `%20` (espace `=32 =0x20`), `%40=@`

Contenu du script côté serveur

- Fichier postecho.sh

```
#!/bin/sh
echo "Content-type: text/html"
echo
echo "<HTML>"
echo "<HEAD><TITLE>Contenu du POST</title</HEAD>"
echo "<H1>Query Results</H1>"
echo "You submitted the following name/value pairs:<p>"
cat
echo
echo "</HTML>"
```

`cat` sans argument recopie l'entrée standard (`stdin`) – qui ici sera le corps de la requête HTTP POST – vers la sortie standard qui sera le corps de la réponse HTTP.

Demonstration POST | Nouvel onglet

file:///Users/groz/Cours/Reseau_1A/Demos_ww 110 %

Demonstration POST

Please help us to improve the World Wide Web by filling in the following questionnaire:

Your organization? Grenoble INP

Commercial? ☐ How many users? 900

Which browsers do you use?

1. Firefox ☑
2. Internet Explorer ☐
3. Chrome ☑
4. Safari ☑
5. Opera ☐
6. Others

A contact point for your site: roland.groz@imag.fr

Many thanks on behalf of the WWW central support team.

Envoyer Effacer

1. Saisie

2. Envoi

HTTP POST

Serveur

```
echo "<HTML>"
echo "<HEAD><TITLE>Contenu du
POST</title</HEAD>"
echo "<H1>Query Results</H1>"
echo "You submitted the following name/value
pairs:<p>"
cat
```

3. Exécution du script

Contenu du POST | Nouvel onglet

localhost/cgi-bin/postecho.sh

Query Results

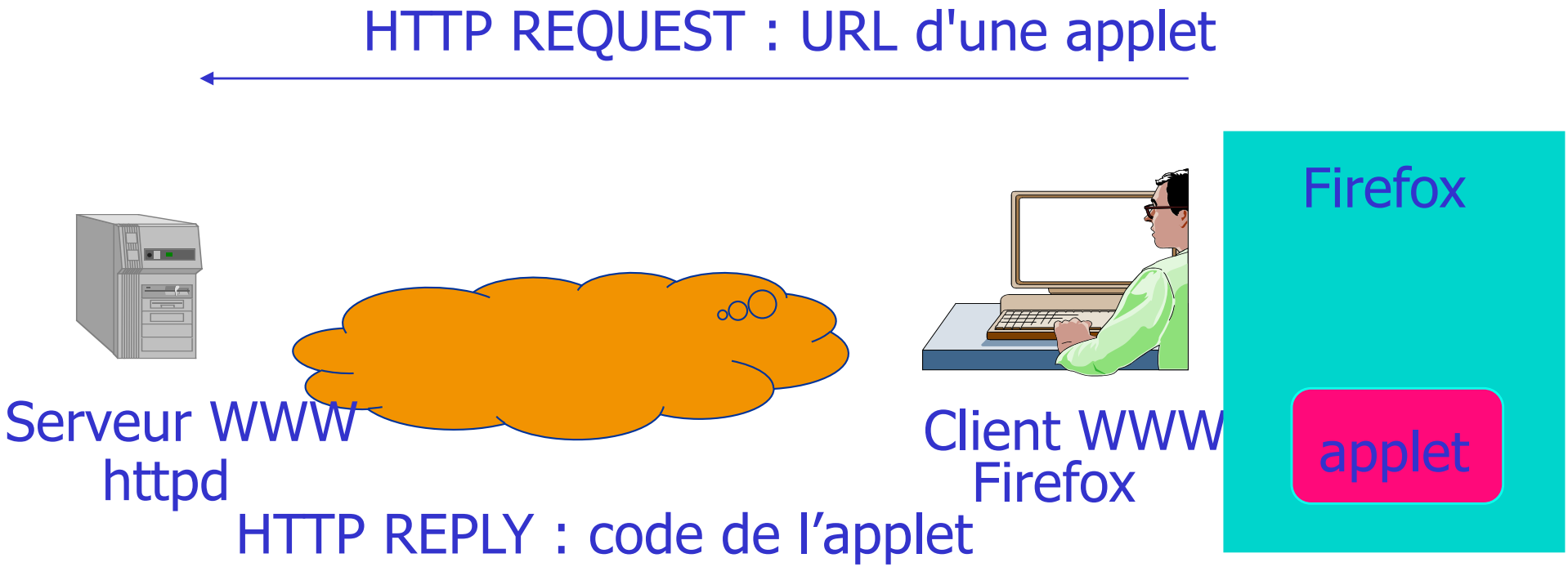
You submitted the following name/value pairs:

org=Grenoble+INP&users=900&browsers=firefox&
browsers=Chrome&browsers=Safari&others=&
contact=roland.groz%40imag.fr

4. Renvoi par HTTP du résultat (page HTML)

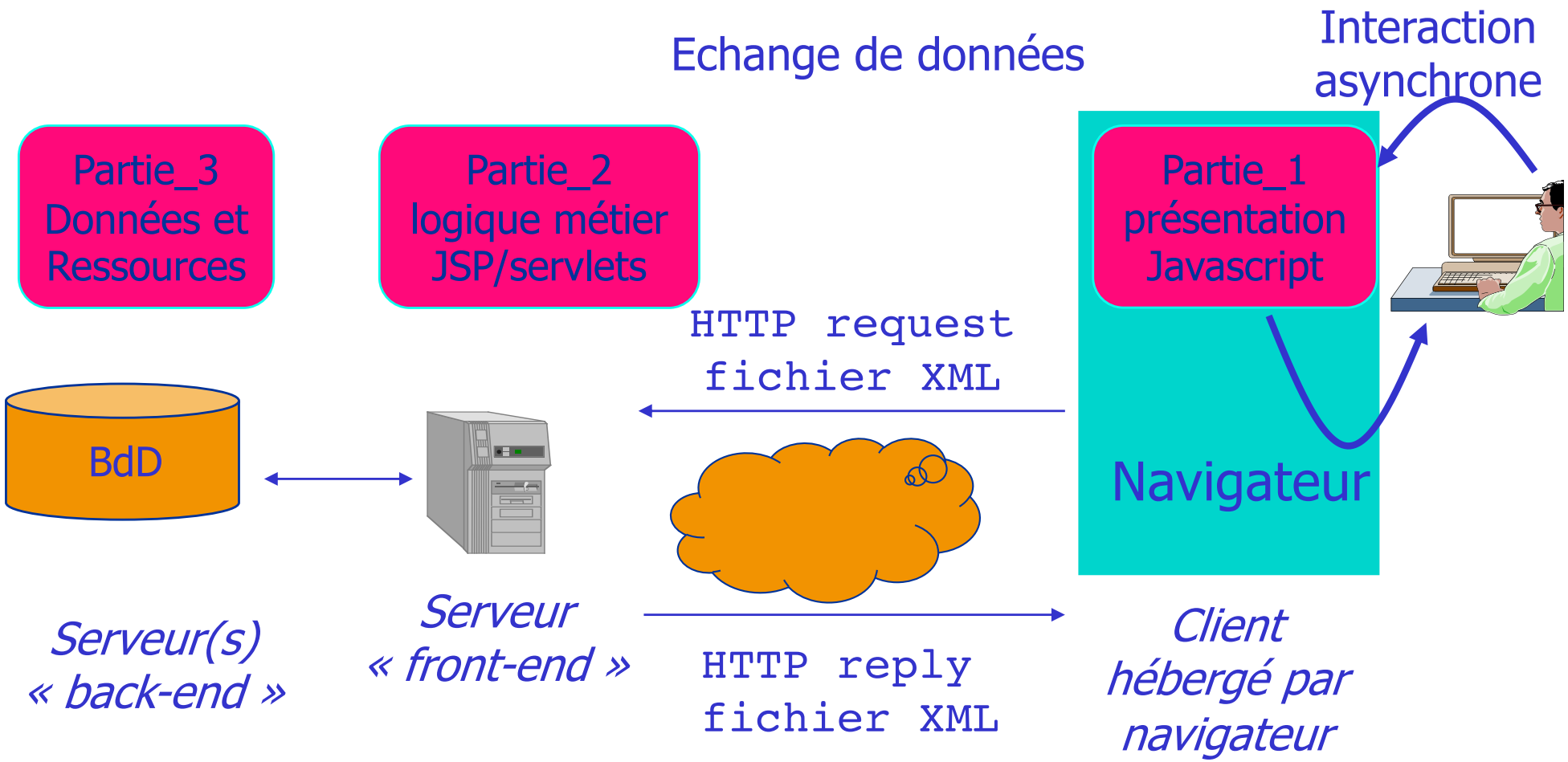
5. Affichage du résultat

Applets (appliquettes)
2ème forme de pages dynamiques



- Au départ, applet en langage Java
- Mais trop dangereux d'exécuter du code Java téléchargé
- Passage à un langage de script directement exécuté (et donc contrôlé) par le navigateur: Javascript

Applications Web « riches », Ajax



- Application répartie selon architecture 3-tiers (3 étages)
- HTTP ne sert plus qu'à
  - -charger l'applet initiale (partie1)
  - -échanger des données (XML, JSON...) entre le client et le serveur

Navigateur et multimédia

- Affichage « de base » par le navigateur
  - HTML
  - IMG: GIF, JPEG
- Applications auxiliaires (processus séparé)
  - Identifiées par le type MIME
  - Ex: application/pdf -> appel de acroread, application/msword -> appel de Word
  - cf. préférences du navigateur (ou fichier mailcap)
- Plug-in (module d'extension – du navigateur)
  - Code d'extension exécuté dans le navigateur
  - Un plug-in est une forme d'appliquette rémanente

Contenu du chapitre AP_www

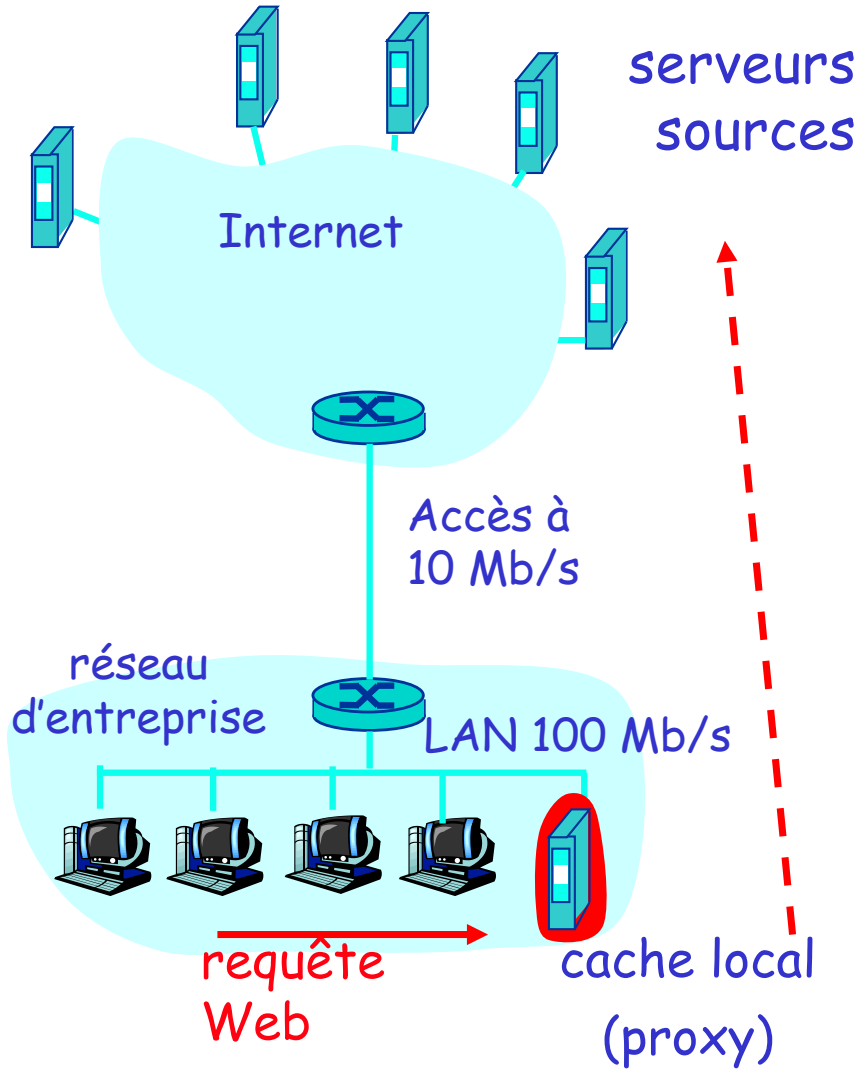
- Les bases historiques et techniques de WWW
  - Hypermédia réparti, fusion de la PAO et des réseaux
  - URL, HTTP, HTML
- HTTP
- HTML: données dynamiques
  - Formulaires
- Documents Web dynamiques
  - CGI & appliquettes
- Cookies, proxies, caches
- Sécurité

Données rémanentes invisibles

- Caches
  - Par utilisateur: le navigateur mémorise les pages consultées (~/.mozilla/…/cache/)
  - Par réseau local/par FAI: les requêtes Web passent par un processus mandataire (proxy)
  - Utilisation des champs HTTP: `Last-Modified` et `If-Modified-Since`
- Cookies
  - Petite donnée stockée par le navigateur à la demande d'un site
  - Rajoutée à toute requête ultérieure vers ce site
- Stockage local par code côté client (HTML5)

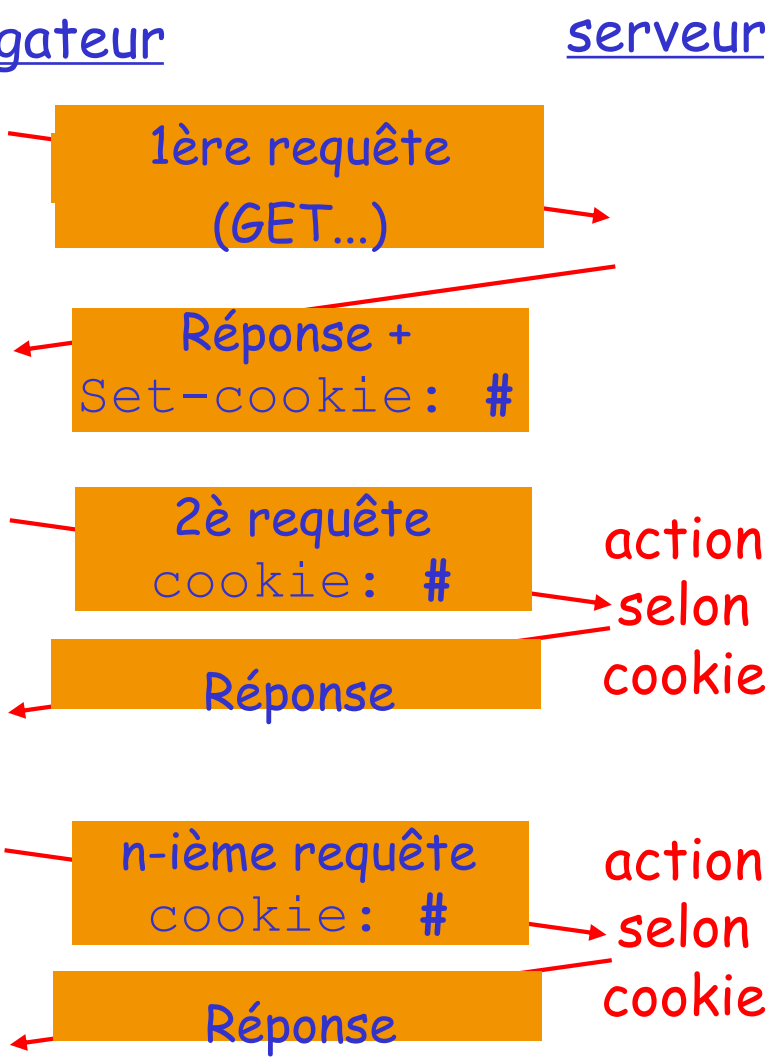
Caches de réseau (proxy)

- Navigateurs configurés pour envoyer les requêtes http au proxy
  - Si page déjà stockée, le proxy renvoie cette page
  - Sinon le proxy contacte le serveur source pour l'URL
- Avantages du cache
  - gain de temps (accès local / accès distant)
  - réduction du trafic redondant vers l'Internet



Cookies

- L'attribut Set-cookie permet d' initialiser ou de réinitialiser une valeur (#) stockée par le navigateur
- Le serveur compare la valeur reçue à ses propres informations
- Cookie: associé à un domaine+chemin
- Intérêt:
  - authentification
  - mémorisation des préférences, de choix antérieurs (No de commande etc.)



"#": en fait une longue chaîne de caractères

Cookies: informations stockées

- Format général (max 4Ko/cookie)

```
Set-Cookie: NAME=VALUE; expires=DATE; path=PATH;
 domain=DOMAIN_NAME; secure
```

- Exemple

```
Set-Cookie: CUSTOMER=groz; path=/; expires=09-Nov-
 2010
Set-Cookie: SHIPPING=Colissimo; path=/catalogue
Cookie: CUSTOMER=groz; SHIPPING=Colissimo
```

- HTTP: au départ protocole *sans état* (req-rép); les cookies permettent d'enrichir le dialogue

- N.B. HTML5 a introduit stockage dans le navigateur jusqu'à 5Mo/domaine (mais géré par code client, ex. JavaScript, pas de changement du protocole)

Sécurité *(cf cours sécurité et TP web)*

- Site contenant des données sensibles
  - Accès protégé par mot de passe: cf TP web
  - Utilisation du champ `Authorization` de HTTP
- Confiance à accorder à un site
  - sécurisation du DNS
  - authentification du serveur par le client: TLS (certificats)
- Chiffrement des échanges : TLS
- TLS: couche intermédiaire entre HTTP et TCP
  - cf cours sécu_cnx (session à distance, ssh) et sécurité

Bilan chapitre AP_www: notions essentielles

- Bases techniques du Web:
  - HTTP: type et format des PDU
  - URL: forme complète, paramètres
  - HTML: fonctionnement des formulaires
- Mécanismes de génération de pages dynamiques
  - Côté serveur: CGI
  - Côté client: applets, Javascript
- Enrichissement du protocole
  - Cookies
  - Proxies